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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи підтримки прийняття управлінських рішень”

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

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

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

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

Іванченку Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи підтримки прийняття управлінських рішень*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи підтримки прийняття управлінських рішень*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Іванченко О.О. Дослідження та програмна реалізація системи підтримки прийняття управлінських рішень. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи підтримки прийняття управлінських рішень.

Метою розробки є дослідження та програмна реалізація системи підтримки прийняття управлінських рішень.

Об'єктом дослідження є процес підтримки прийняття управлінських рішень.

Предметом дослідження є методи підтримки прийняття управлінських рішень.

Методи дослідження базуються на методах теорії підтримки прийняття управлінських рішень, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи підтримки прийняття управлінських рішень.

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'ютерні науки, підтримка прийняття управлінських рішень

ABSTRACT

Ivanchenko O.O. Research and software implementation of a management decision support system. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the management decision support system.

The purpose of the development is research and software implementation of a management decision support system.

The object of research is the process of supporting management decision-making.

The subject of the research is methods of supporting management decision-making.

Research methods are based on the methods of the theory of management decision-making support, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the management decision support system.

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

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

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

The program was developed in the Delphi 10.4 environment.

Keywords: computer science, management decision support

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ВД	–	вітрина даних
ЗД	–	зберігання даних
ЗСП	–	збалансована система показників
ЖЦ	–	життєвий цикл
ІАД	–	інтелектуальний аналіз даних
ІС	–	інформаційна система
ІСУП	–	інтегровані системи управління підприємством
ОПР	–	особа, яка приймає рішення
ПЗ	–	програмне забезпечення
СЗД	–	система зберігання даних
СППР	–	система підтримки прийняття рішень
СУБД	–	система управління базами даних
УР	–	управлінське рішення
OLAP	–	On-Line Analytical Processing – оперативна аналітична обробка

ВСТУП

Актуальність теми. У цей час в усьому світі менеджери сприймають системи підтримки прийняття рішень (СППР) як один з факторів, що сприяють одержанню переваг в умовах ринкової конкуренції. СППР припускають досить глибоке пророблення даних, спеціально перетворених так, щоб їх було зручно використовувати в ході процесу прийняття рішень. Невід'ємним компонентом СППР є правила прийняття рішень, які на основі агрегованих даних підказують менеджерському складу виводи. Такого роду системи створюються тільки в тому випадку, коли структура управління вже досить визначена і є підстави для узагальнення й аналізу не тільки даних, але й процесів їхньої обробки. Таким чином, СППР – це не простий розвиток системи оперативного управління, це механізм розвитку підприємства, що містить у собі деяку частину управляючої системи, велику систему зовнішніх зв'язків підприємства, а також технологічні й маркетингові процеси розвитку виробництва.

У тому або іншому ступені елементи автоматизованої підтримки прийняття рішень присутні в будь-якій інформаційній системі (ІС). Тому, чи усвідомлено ні, до завдання автоматизації процесу прийняття рішень організації приступають відразу після придбання обчислювальної техніки й установки програмного забезпечення. У міру розвитку підприємства, упорядкування його структури й налагодження міжкорпоративних зв'язків, проблема розробки й впровадження СППР стає особливо актуальною.

Якість СППР у першу чергу залежить від:

– даних, на підставі яких приймаються рішення;

– використовуваних аналітичних методів і моделей обробки й аналізу даних;

– адекватності використовуваних інструментальних засобів завданням прийняття рішень.

Однак існують фактори, що стримують побудову сучасних СППР управління для промислових підприємств, що дозволяють автоматизувати

підготовку даних для прийняття управлінських рішень (невірогідність даних, низька продуктивність при аналітичних запитах, неможливість перетворення різнорідних даних у єдину інформацію). Цим обумовлена актуальність дослідження питань створення СППР управління для промислових підприємств на основі вдосконалених технологій накопичення й зберігання даних, що усуває виділені фактори за допомогою інтеграції сховищ даних (СЗД, Data Warehouse), оперативної аналітичної обробки (OLAP, On-Line Analytical Processing) і інтелектуального аналізу даних (ІАД).

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи підтримки прийняття управлінських рішень.

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

– Огляд існуючих систем підтримки прийняття управлінських рішень.

– Дослідження системи підтримки прийняття управлінських рішень.

– Програмна реалізація системи підтримки прийняття управлінських рішень.

Об'єктом дослідження є процес підтримки прийняття управлінських рішень.

Предметом дослідження є методи підтримки прийняття управлінських рішень.

Методи дослідження базуються на методах теорії підтримки прийняття управлінських рішень, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод підтримки прийняття управлінських рішень.

– Розроблено вітчизняний продукт підтримки прийняття управлінських рішень, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі підтримки прийняття управлінських рішень.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи підтримки прийняття управлінських рішень, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система підтримки прийняття рішень (СППР) – це комп’ютерна програма, яка використовується для покращення можливостей компанії приймати рішення. Він аналізує великі обсяги даних і надає організації найкращі можливі варіанти.

Системи підтримки прийняття рішень об’єднують дані та знання з різних областей і джерел, щоб надати користувачам інформацію, окрім звичайних звітів і зведень. Це допоможе людям приймати зважені рішення.

Типова інформація, яку може збирати та представляти програма підтримки прийняття рішень, включає наступне:

– порівняльні показники продажів між одним тижнем і наступним;

– прогнозовані цифри доходу, засновані на припущеннях про продажі нових продуктів;

– наслідки різних рішень.

Система підтримки прийняття рішень – це інформаційна програма на відміну від операційної програми. Інформаційні програми надають користувачам релевантну інформацію на основі різноманітних джерел даних для підтримки більш обґрунтованого прийняття рішень. Операційні програми, навпаки, записують деталі бізнес-операцій, включаючи дані, необхідні для підтримки прийняття рішень бізнесом.

1.2 Область застосування

Типовий СППР складається з трьох різних частин: бази даних знань, програмного забезпечення та інтерфейсу користувача.

База знань. База знань є невід'ємною частиною бази даних системи підтримки прийняття рішень, яка містить інформацію як із внутрішніх, так і зовнішніх джерел. Це бібліотека інформації, пов'язана з певними предметами, і є частиною СППР, яка зберігає інформацію, яка використовується механізмом міркування системи для визначення курсу дій.

Програмна система. Програмна система складається з модельних систем управління. Модель – це симуляція системи реального світу з метою розуміння того, як система працює та як її можна покращити. Організації використовують моделі, щоб передбачити, як зміняться результати з різними коригуваннями системи.

Наприклад, моделі можуть бути корисними для розуміння систем, які є надто складними, надто дорогими або надто небезпечними для повного вивчення в реальному житті. Це ідея комп'ютерного моделювання, яке використовується для наукових досліджень, інженерних випробувань, прогнозування погоди та багатьох інших застосувань.

Моделі також можна використовувати для представлення та дослідження систем, яких ще не існує, як-от запропонована нова технологія, запланована фабрика чи ланцюжок постачання підприємства. Компанії також використовують моделі для прогнозування результатів різних змін у системі, таких як політика, ризики та правила, щоб допомогти прийняти бізнес-рішення.

Інтерфейс користувача. Інтерфейс користувача забезпечує легку навігацію системою. Основна мета інтерфейсу користувача системи підтримки прийняття рішень полягає в тому, щоб користувач міг легко маніпулювати даними, які в ньому зберігаються.

Компанії можуть використовувати інтерфейс для оцінки ефективності транзакцій СППР для кінцевих користувачів. Інтерфейси СППР включають прості вікна, складні інтерфейси на основі меню та інтерфейси командного рядка.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи підтримки прийняття управлінських рішень, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Комп'ютеризовані клінічні системи підтримки прийняття рішень, або ККСППР, являють собою зміну парадигми сучасної охорони здоров'я. ККСППР використовуються, щоб допомогти клініцистам у їхніх складних процесах прийняття рішень. З моменту першого використання в 1980-х роках ККСППР швидко розвивався. Нині вони зазвичай використовуються за допомогою електронних медичних записів та інших комп'ютеризованих клінічних робочих процесів, чому сприяло все більш широке впровадження електронних медичних записів із розширеними можливостями. Незважаючи на ці досягнення, залишається невідомим вплив ККСППР на постачальників, які їх використовують, результати пацієнтів і витрати. За останні десятиліття було опубліковано багато прикладів успішних історій ККСППР, але помітні невдачі також показали нам, що ККСППР не без ризиків. У цій статті ми надаємо найсучасніший огляд використання систем підтримки клінічних рішень у медицині, включаючи різні типи, поточні випадки використання з доведеною ефективністю, типові підводні камені та потенційну шкоду. Ми закінчуємо рекомендаціями, заснованими на фактичних даних, щодо мінімізації ризиків у проектуванні, впровадженні, оцінці та обслуговуванні ККСППР.

Вступ: що таке клінічна система підтримки прийняття рішень?

Система підтримки клінічних рішень (ККСППР) призначена для покращення надання медичної допомоги шляхом посилення медичних рішень за допомогою цільових клінічних знань, інформації про пацієнтів та іншої інформації про здоров'я. Традиційний ККСППР складається з програмного

забезпечення, призначеного для прямої допомоги у прийнятті клінічних рішень, у якому характеристики окремого пацієнта зіставляються з комп'ютеризованою базою клінічних знань, а оцінки чи рекомендації для конкретного пацієнта потім надаються клініцисту для рішення. Сьогодні ККСППР в основному використовуються на місці надання медичної допомоги, щоб клініцист міг поєднати свої знання з інформацією чи пропозиціями, які надає ККСППР. Однак все частіше розробляються ККСППР із можливістю використання даних і спостережень, які інакше неможливо отримати або інтерпретувати людьми.

Комп'ютерні ККСППР можна простежити до 1970-х років. У той час вони мали погану системну інтеграцію, вимагали багато часу та часто обмежувалися академічними заняттями. Виникали також етичні та правові питання щодо використання комп'ютерів у медицині, автономії лікаря та того, хто буде винен у використанні рекомендацій системи з недосконалою «пояснюваністю». Зараз ККСППР часто використовує веб-додатки або інтеграцію з електронними медичними записами (EHR) і комп'ютеризованими системами введення замовлення постачальника (CPOE). Ними можна керувати за допомогою настільного комп'ютера, планшета, смартфона, а також інших пристроїв, таких як біометричний моніторинг і переносна медична технологія. Ці пристрої можуть або не можуть створювати вихідні дані безпосередньо на пристрої або бути пов'язані з базами даних EHR.

ККСППР були класифіковані та підрозділені на різні категорії та типи, включаючи час втручання та те, чи мають вони активну чи пасивну доставку. ККСППР часто класифікують як засновані на знаннях або не засновані на знаннях. У системах, заснованих на знаннях, створюються правила (вирази IF-THEN), при цьому система отримує дані для оцінки правила та створює дію або вихід; Правила можна розробити, використовуючи докази, засновані на літературі, на практиці або націлені пацієнтом. ККСППР, які не ґрунтуються на знаннях, усе ще потребують джерела даних, але рішення використовує штучний інтелект (ШІ), машинне навчання (ML) або статистичне розпізнавання образів, а

не програмується на дотримання експертних медичних знань. ККСППР, що не базується на знаннях, хоча швидко зростає як приклад використання штучного інтелекту в медицині, рясніє проблемами, включаючи проблеми з розумінням логіки, яку використовує ШІ для створення рекомендацій (чорні ящики), і проблеми з доступністю даних. Вони ще не досягли широкого впровадження.

Вони складаються з

– бази: правил, запрограмованих у системі (на основі знань), алгоритму, що використовується для моделювання рішення (не на основі знань), а також доступних даних;

– механізму висновку: бере запрограмовані або визначені штучним інтелектом правила та структури даних і застосовує їх до клінічних даних пацієнта, щоб створити результат або дію, яка представлена кінцевому користувачеві (наприклад, лікарю) через;

– механізм зв'язку: веб-сайт, додаток або зовнішній інтерфейс EHR, за допомогою якого кінцевий користувач взаємодіє з системою.

Повнорозмірне зображення

ККСППР було схвалено актами уряду США про охорону здоров'я та медичну допомогу, що фінансово стимулює впровадження CDS в EHR. У 2018 році приблизно 41% лікарень США з EHR також мали ККСППР, а в 2022 році 40,2% лікарень США мали вдосконалені можливості CDS (HIMSS Stage 6). В інших місцях показники впровадження EMR були багатообіцяючими: приблизно 62% практиків у Канаді у 2018 році. Канада отримала значну підтримку з боку уряду, а також Infoway, некомерційної корпорації. Англія також є світовим лідером у сфері IT-інвестицій в охорону здоров'я, інвестувавши до 20 мільярдів євро ще в 2010 році. Декільком країнам також вдалося запровадити національні записи про стан здоров'я, принаймні для даних пацієнтів, зокрема Данії, Естонії, Австралії., та інші.

Обсяг функцій, наданих ККСППР, величезний, включаючи діагностику, системи сигналізації, лікування захворювань, призначення (Rx), контроль ліків та

багато іншого. Вони можуть проявлятися у вигляді комп'ютеризованих сповіщень і нагадувань, комп'ютеризованих інструкцій, наборів замовлень, звітів про дані пацієнтів, шаблонів документації та інструментів клінічного робочого процесу. Кожна функція ККСППР буде детально обговорюватися в цьому огляді з потенційними й реалізованими перевагами цих функцій, а також небажаними негативними наслідками та стратегіями уникнення шкоди від ККСППР.

Функції та переваги ККСППР

Безпека пацієнтів

Стратегії зменшення помилок при лікуванні зазвичай використовують ККСППР. Помилки, пов'язані з взаємодією між лікарськими засобами (DDI), вважаються поширеними і їм можна запобігти, при цьому до 65% пацієнтів стаціонарного лікування піддаються впливу однієї чи кількох потенційно шкідливих комбінацій. Систем CPOE тепер розроблено з програмним забезпеченням безпеки ліків, яке має гарантії для дозування, дублювання терапії та перевірки DDI. Типи сповіщень, створювані цими системами, є одними з найбільш поширених видів підтримки прийняття рішень. Однак дослідження виявили високий рівень варіативності між тим, як відображаються сповіщення про DDI (наприклад, пасивні чи активні/руйнівні), які пріоритети та в алгоритмах, що використовуються для ідентифікації DDI. Системи часто мають різний рівень нерелевантних сповіщень, представлених, і немає стандарту щодо того, як найкраще реалізувати які сповіщення для постачальників. Офіс національного координатора з інформаційних технологій охорони здоров'я США розробив список «високопріоритетних» DDI для CDS, який досяг різних рівнів схвалення та розгортання в ККСППР в інших країнах, включаючи Великобританію, Бельгію та Корею.

Повнорозмірний стіл

Інші системи, націлені на безпеку пацієнтів, включають електронні системи видачі ліків (EDDS) і системи введення ліків за допомогою штрих-коду в пунктах надання медичної допомоги (BPOC). Вони часто впроваджуються разом,

щоб створити «замкнутий цикл», де кожен крок процесу (призначення, транскрибування, відпуск, адміністрування) комп'ютеризований і відбувається в межах підключеної системи. Під час введення ліки автоматично ідентифікуються за допомогою радіочастотної ідентифікації (RFID) або штрих-кодів і перевіряються з інформацією про пацієнта та рецептами. Це є ще однією ціллю для ККСППР, і потенційна перевага полягає в запобіганні помилок при введенні ліків, що виникають біля ліжка (на відміну від подальших дій). Прийняття відносно низьке, частково через високі технологічні вимоги та витрати. Проте; дослідження показують високу ефективність цих систем у зменшенні помилок. Мохоні та ін. показали, що багато з цих систем можна поєднувати з CPOE та ККСППР одночасно, зі зниженим рівнем помилок при призначенні для виявлення алергії на ліки, надмірного дозування та неповного або нечіткого порядку. Як і у більшості ККСППР, помилки можуть виникнути, якщо провайдери пропускають або навмисно обходять технологію.

ККСППР також покращує безпеку пацієнтів завдяки системам нагадувань про інші медичні події, а не лише про ті, які пов'язані з прийомом ліків. Серед численних прикладів ККСППР для вимірювання рівня глюкози в крові у відділенні інтенсивної терапії вдалося зменшити кількість випадків гіпоглікемії. Цей ККСППР автоматично запропонував медсестрам провести вимірювання рівня глюкози відповідно до місцевого протоколу моніторингу рівня глюкози, який вказував, як часто слід проводити вимірювання відповідно до конкретних демографічних даних пацієнта та попередніх рівнів/тенденцій глюкози.

Загалом, ККСППР, націлена на безпеку пацієнтів через CPOE та інші системи, загалом була успішною у зменшенні помилок у призначенні та дозуванні, протипоказань завдяки автоматичним попередженням, моніторингу лікарських подій тощо. Безпеку пацієнтів можна вважати другорядною метою (або вимогою) майже всіх типів ККСППР, незалежно від основної мети їх впровадження.

Клінічний менеджмент

Дослідження показали, що ККСППР може покращити дотримання клінічних рекомендацій. Це важливо, оскільки було показано, що традиційні клінічні рекомендації та шляхи лікування важко реалізувати на практиці за низької прихильності клініциста. Припущення про те, що практики будуть читати, сприймати та застосовувати нові рекомендації, не справдилося. Однак правила, неявно закодовані в настановах, можуть бути буквально закодовані в ККСППР. Такий ККСППР може приймати різноманітні форми, від стандартизованих наборів замовлень для цільового випадку, попереджень про певний протокол для пацієнтів, яких він стосується, нагадувань про тестування тощо. Крім того, ККСППР може допомогти в веденні пацієнтів за протоколами дослідження/лікування, відстеження та розміщення замовлень, подальше спостереження за направленнями, а також забезпечення профілактичного догляду.

ККСППР також може сповістити клініцистів про те, щоб вони звернулися до пацієнтів, які не дотримувалися планів лікування або мають пройти подальше спостереження, і допомогти визначити пацієнтів, які підходять для дослідження на основі конкретних критеріїв. Система ККСППР, розроблена та впроваджена в клініці Клівленда, забезпечує сповіщення лікарів на місці надання медичної допомоги, коли історія пацієнта відповідає критеріям клінічного випробування. Сповіщення пропонує користувачеві заповнити форму, яка встановлює відповідність вимогам і згоду на контакт, пересилає карту пацієнта координатору дослідження та друкує інформаційний аркуш пацієнта для клінічного випробування.

Стримування витрат

ККСППР може бути економічно ефективним для систем охорони здоров’я через клінічні втручання, 38 скорочення тривалості перебування пацієнтів у стаціонарі, інтегровані в CPOE системи, що пропонують дешевші альтернативи лікам, 39 або зменшення дублювання тестів. Правило CPOE було впроваджено в педіатричному відділенні інтенсивної терапії серцево-судинних захворювань

(ВІТ), яке обмежувало планування аналізу крові, хімії та панелей коагуляції 24-годинним інтервалом. Це зменшило використання лабораторних ресурсів із прогнозованою економією коштів у розмірі 717 538 доларів США на рік без збільшення тривалості перебування (LOS) або смертності.

ККСППР може повідомити користувача про дешевші альтернативи лікам або умови, які покриватимуть страхові компанії. У Німеччині багатьох стаціонарних пацієнтів переводять на ліки за лікарняними формулярами. Виявивши, що 1 з 5 замін були неправильними, лікарня Гейдельберга розробила алгоритм заміни препарату та інтегрувала його в існуючу систему CPOE. ККСППР може автоматично перемикати 91,6% із 202 консультацій щодо лікування без помилок, підвищуючи безпеку, зменшуючи робоче навантаження та знижуючи витрати для постачальників.

Адміністративні функції

ККСППР забезпечує підтримку клінічного та діагностичного кодування, замовлення процедур і тестів, а також сортування пацієнтів. Розроблені алгоритми можуть запропонувати уточнений список діагностичних кодів, щоб допомогти лікарям вибрати найбільш підходящий(-і). ККСППР було розроблено для усунення неточності кодування госпіталізації відділень невідкладної допомоги ICD-9 (ICD – це Міжнародна статистична класифікація хвороб, стандартизовані коди, які використовуються для представлення хвороб і діагнозів). Інструмент використовував анатомографічний інтерфейс (візуальне, інтерактивне представлення людського тіла), пов’язаний із кодами ICD, щоб допомогти лікарям швидкої допомоги точно знаходити діагностичні коди вступу швидше.

ККСППР може безпосередньо покращити якість клінічної документації. Акушерський ККСППР мав покращену систему підказок, що значно покращило документування показань до індукції пологів і оціненої ваги плода порівняно з контрольною лікарнею. Точність документації важлива, оскільки вона може безпосередньо допомогти клінічним протоколам. Наприклад, було запроваджено

ККСППР, щоб переконатися, що пацієнти були належним чином вакциновані після спленектомії, щоб боротися з підвищеним ризиком інфекцій (включаючи пневмококову, *Haemophilus influenzae*, менінгококову тощо), які пов'язані з видаленням селезінки. Однак автори виявили, що у 71% пацієнтів із терміном «спленектомія» в EHR це не було задокументовано в списку проблем (що і викликає попередження ККСППР). Потім був розроблений додатковий ККСППР для покращення документації списку проблем спленектомії та покращення користі оригінального ККСППР вакцинації.

Підтримка діагностики

ККСППР для клінічної діагностики відомі як системи підтримки діагностичних рішень (DCППР). Ці системи традиційно передбачають комп'ютеризовану «консультацію» або етап фільтрації, за допомогою якого їм можуть надаватися дані/вибір користувача, а потім виводитися список можливих або ймовірних діагнозів. На жаль, DCППР не має такого великого впливу, як інші типи ККСППР (поки що) через негативне сприйняття та упередження лікарів, низьку точність (часто через прогалини в доступності даних) і погану системну інтеграцію, що вимагає ручного введення даних. Останній покращується завдяки кращій інтеграції EHR та стандартизованому словниковому запасу, як-от клінічні терміни Snomed.

Хорошим прикладом ефективного DCППР є той, який був створений Kunhimangalam et al. для діагностики периферичної нейропатії за допомогою нечіткої логіки. За допомогою 24 полів введення, які включають симптоми та результати діагностичних тестів, вони досягли 93% точності в порівнянні з експертами у визначенні моторних, сенсорних, змішаних нейропатій або нормальних випадків. Хоча це має велике значення, особливо в країнах із меншим доступом до визнаних клінічних експертів, також є бажання мати системи, які могли б доповнювати спеціалізовану діагностику. DXplain – це електронний довідковий DCППР, який забезпечує вірогідний діагноз на основі клінічних проявів. У рандомізованому контрольному дослідженні, в якому взяли участь 87

лікарів сімейної медицини, ті, хто був рандомізований для використання системи, показали значно вищу точність (84% проти 74%) у підтвердженому діагностичному тесті, який включав 30 клінічних випадків.

Враховуючи відомі випадки діагностичних помилок, особливо в первинній медичній допомозі, є багато надії на те, що ККСППР та ІТ-рішення покращать діагностику. Зараз ми бачимо, як діагностичні системи розробляються з використанням методів, не заснованих на знаннях, таких як машинне навчання, що може прокласти шлях для більш точної діагностики. Сортувальна та діагностична система Babylon у Великій Британії є хорошим прикладом потенціалу, а також роботи, яку ще потрібно виконати, перш ніж ці системи будуть готові до запуску.

Підтримка діагностики: зображення

ККСППР для візуалізації на основі знань зазвичай використовується для впорядкування зображень, де ККСППР може допомогти радіологам у виборі найбільш відповідного тесту для виконання, надаючи нагадування про найкращі практичні рекомендації або попереджаючи про протипоказання до контрасту, наприклад. Показано, що інтервенційна CDS для замовлення зображень у медичному центрі Вірджинії Мейсон істотно знижує рівень використання МРТ поперекового відділу для лікування болю в попереку, МРТ голови для лікування головного болю та КТ синусів для лікування синуситу. CDS вимагає від постачальників відповіді на низку запитань перед замовленням зображень (POC), щоб перевірити відповідність. Важливо, що якщо зображення було відхилено, система запропонувала альтернативу. Іншим комерціалізованим прикладом є RadWise®, який направляє клініцистів до найрелевантнішого порядку візуалізації, аналізуючи симптоми пацієнта та зіставляючи їх із великою базою даних діагнозів, а також надаючи відповідні рекомендації щодо використання на місці надання допомоги.

Існує великий інтерес до CDS, що не базується на знаннях, для вдосконаленої візуалізації та точної радіології («радіоміка»). Оскільки зображення

становлять все більшу кількість медичних даних, але вимагають значної ручної інтерпретації, постачальникам потрібні технології, які допоможуть їм у вилученні, візуалізації та інтерпретації. Технології штучного інтелекту виявилися здатними надавати інформацію про дані більше, ніж люди. Для цього ці технології використовують розширені алгоритми розпізнавання пікселів і класифікації зображень, найвідоміший з яких: глибоке навчання (DL). IBM Watson Health, DeepMind, Google та інші компанії знаходяться в авангарді, розробляючи продукти для виявлення пухлин, інтерпретації медичних зображень, діагностики діабетичної ретинопатії, діагностики хвороби Альцгеймера за допомогою мультимодального вивчення функцій, і незліченну кількість інших. IBM Watson «Eyes of Watson» зміг поєднати розпізнавання зображень сканування мозку з розпізнаванням тексту описів випадків, щоб забезпечити комплексну підтримку прийняття рішень (або те, що IBM описує як «когнітивний помічник»).

Кілька проектів змогли продемонструвати продуктивність, яка, безперечно, «на одному рівні» з експертами-людьми. Наприклад, команда Google навчила глибоку згорточну нейронну мережу (CNN) виявляти діабетичну ретинопатію (пошкодження кровоносних судин ока) на основі набору даних із 130000 зображень сітківки з дуже високою чутливістю та специфічністю. Продуктивність алгоритмів була на одному рівні з сертифікованими офтальмологами США. Інше дослідження, нещодавно опубліковане Стенфордською групою, продемонструвало CNN для виявлення аритмій на електрокардіограмі, що перевищувало точність (F1 і чутливість із відповідною специфічністю) середнього кардіолога для всіх класів ритму. З нинішнім темпом прогресу деякі експерти неоднозначно припускають, що через 15–20 років більшість інтерпретації діагностичних зображень здійснюватиметься (або принаймні попередньо оброблятиметься) комп’ютерами. Однак на даний момент ми повинні розглядати ці ранні системи як доповнення або розширення наявного набору інструментів клініциста.

Діагностичне забезпечення: лабораторне та патологоанатомічне

Ще одна підгрупа діагностики, де ККСППР може бути корисним, – це лабораторне тестування та інтерпретація. Попередження та нагадування про відхилення від норми лабораторних результатів є простими та повсюдними в системах EHR. ККСППР також може розширити корисність лабораторних тестів, щоб уникнути більш ризикованої або інвазивної діагностики. У тестуванні на гепатит В і С біопсія печінки вважається золотим стандартом для діагностики, тоді як неінвазивні лабораторні тести недостатньо точні, щоб бути прийнятими. Однак, розробляються моделі штучного інтелекту, які поєднують численні тести (сироваткові маркери, візуалізацію та генні тести), щоб забезпечити набагато більшу точність. Існує також застосування для ККСППР як інструменту інтерпретації, де контрольні діапазони тесту є високоперсоніфікованими, наприклад, вік, стать або підтипи захворювання.

Звіти про патологію мають вирішальне значення для багатьох інших медичних спеціальностей. Деякі ККСППР можна використовувати для автоматичної класифікації пухлини. Це було зроблено для класифікації пухлини сечового міхура та оцінки рецидиву з точністю до 93%. Те ж саме було зроблено для класифікації та класифікації пухлин головного мозку. Є багато інших прикладів, включаючи комп'ютеризований аналіз ЕКГ, автоматизовану інтерпретацію газів артеріальної крові, звіти електрофорезу білка та ККСППР для підрахунку клітин крові.

Підтримка прийняття рішень, орієнтована на пацієнта

З появою «Особистої медичної картки» (PHR) ми бачимо інтегровану функціональність CDS, подібну до EHR, з пацієнтом як кінцевим користувачем або «менеджером» даних. Це чудовий крок до лікування, орієнтованого на пацієнта, а PHR з підтримкою CDS є ідеальним інструментом для впровадження спільного прийняття рішень між пацієнтом і постачальником, зокрема тому, що ККСППР може усунути «брак інформації» як бар'єр для участі пацієнта в власний догляд. PHR часто розробляються як розширення комерційного програмного

забезпечення EHR або як автономні веб-або мобільні програми. При підключенні до EHR PHR можуть мати двосторонній зв'язок, за допомогою якого інформація, введена безпосередньо пацієнтом, може бути доступною для їхніх постачальників, а також інформація в EHR може бути передана в PHR для перегляду пацієнтами.

Один із найперших PHR, «Шлюз пацієнтів», був просто інформаційною панеллю для пацієнтів, щоб переглядати ліки та лабораторії та спілкуватися зі своїми лікарями. Це розширилося, і тепер деякі системи дозволяють пацієнтам змінювати власні записи про медичне обслуговування, впливаючи також на дані EHR. Іншим прикладом є MyHealthAtVanderbilt університету Вандербільта, PHR, повністю інтегрована в інституційну EHR. На додаток до доставки освітніх матеріалів для пацієнтів, націлених на захворювання, вони включили Flu Tool для пацієнтів із грипоподібними симптомами, щоб визначити рівень медичної допомоги, який їм потрібен, а потім допомогти їм шукати лікування. 79 Відстеження симптомів є корисною та загальною функцією PHR, але різноманітність зібраних даних практично безмежна, від алергії до страхового покриття до інформації про рецепти та ліки. Крім того, PHR та інші додатки для моніторингу пацієнтів можуть бути розроблені для збору інформації з пристроїв для здоров'я та інших переносних пристроїв, щоб створити практичну інформацію для постачальників. Чудовим прикладом є лікування діабету. Багато систем уже використовуються, але одна, зокрема, вперше розроблена Стенфордською школою медицини, використовує переносний монітор глюкози, який передає дані на пристрій Apple (HealthKit). Apple зробила HealthKit сумісним із Epic EHR та Epic PHR, «MyChart». Це успішно дозволяє постачальникам відстежувати тенденції рівня глюкози у своїх пацієнтів між візитами та зв'язуватися з ними через MyChart для подальшого спостереження або термінових рекомендацій. Пілотне дослідження продемонструвало покращення робочого процесу постачальників, спілкування з пацієнтами та, зрештою, якість медичної допомоги. Різноманітні інші галузі медицини розгортають подібні системи для

моніторингу, які поєднують PHR/EHR, портативні технології та ККСППР, включаючи, але не обмежуючись серцевою недостатністю (кардіологія), гіпертонією, апное сну, паліативним доглядом/доглядом за літніми людьми тощо.

Варто зазначити, що оскільки PHR стали більш просунутими з можливостями ККСППР, також все більше уваги приділялося дизайну цих систем, щоб обслуговувати спільне прийняття рішень між пацієнтом і постачальником, а також бути інтерактивними інструментами, щоб зробити пацієнтів більш обізнаними/залученими до власний догляд. PHR, які служать лише сховищем інформації про здоров'я, тепер розглядаються як недоречні, особливо самі пацієнти.

Підводні камені ККСППР

Фрагментовані робочі процеси

ККСППР може порушити робочий процес клініциста, особливо у випадку автономних систем. Багато ранніх ККСППР були розроблені як системи, які вимагали від провайдера документувати або отримувати інформацію за межами свого типового робочого середовища. ККСППР також порушує робочий процес, якщо розроблено без урахування обробки інформації та поведінки людини. У відповідь на це ККСППР було розроблено з використанням методу «роздумів вголос», щоб змоделювати робочий процес практиків і створити систему з кращою зручністю використання.

Порушений робочий процес може призвести до збільшення когнітивних зусиль, більше часу, потрібного для виконання завдань, і менше часу віч-на-віч з пацієнтами. Навіть якщо ККСППР добре інтегровано в існуючі інформаційні системи, може бути розрив між взаємодією обличчям до обличчя та взаємодією з комп'ютерною робочою станцією. Дослідження показали, що практики з більш глибоким досвідом мають меншу ймовірність використовувати ККСППР, а більше ймовірно, що вони замінять його.

Попередження про втому та невідповідні сповіщення

Дослідження виявили, що до 95% сповіщень ККСППР є несуттєвими, і часто лікарі не погоджуються з попередженнями або не довіряють їм. Іншим разом їх просто не читають. Якщо лікарям надають надмірні/неважливі попередження, вони можуть страждати від попереджувальної втоми.

Тривожні сповіщення мають бути обмежені більш загрозливими для життя або непрямими протипоказаннями, такими як серйозна алергія. Однак; навіть попередження про алергію можуть бути неправильними, і клініцисти часто перевірятимуть себе, особливо якщо джерелом є інше місце/лікарня/практикуючий лікар. Сповіщення про ліки також можуть стосуватися конкретної спеціальності, але не мають значення, якщо їх вирвати з контексту. Наприклад, попередження про заборону використання антибіотиків широкого спектру дії, таких як ванкоміцин, може бути недоречним у відділенні інтенсивної терапії. 85 Попередження щодо дублікатів ліків може бути недоречним у клініках із запальним захворюванням кишечника, де той самий клас ліків можна застосовувати різними шляхами для посилення ефекту.

Вплив на навички користувача

До СРОЕ та ККСППР постачальники медичних послуг, фармацевти та медсестри покладалися виключно на повторну перевірку замовлень. ККСППР може створити враження, що перевірка точності замовлення є непотрібною або автоматичною. Це важливий міф, який потрібно розвіяти.

Також важливо враховувати потенційний довгостроковий вплив ККСППР на користувачів. З часом ККСППР може надавати тренувальний ефект, так що сама ККСППР більше не потрібна. Придуманий «ефект переносу», швидше за все, з ККСППР, які мають навчальний характер. І навпаки, провайдери можуть занадто сильно покладатися або довіряти ККСППР для виконання конкретного завдання. 89 Це можна порівняти з використанням калькулятора для математичних операцій протягом тривалого періоду часу, а потім зниження розумових навичок математики. Це потенційно проблематично, оскільки

користувач має менше незалежності та буде менше оснащений для виконання цього завдання, якщо він перейде на середовище без ККСППР.

ККСППР може залежати від комп'ютерної грамотності

Відсутність технологічних навичок може стати перешкодою під час роботи з ККСППР. Це може відрізнятися залежно від деталей дизайну ККСППР, але деякі з них виявилися надто складними, що надто покладаються на навички користувача. Системи повинні прагнути залишатися якомога ближчими до основної функціональності існуючої системи. Незважаючи на це, усі нові системи мають період навчання, тому базові оцінки технологічної компетентності користувачів можуть бути доречними. Потім може бути забезпечено подальше навчання, щоб полегшити повне використання можливостей ККСППР, або більш чіткі вказівки, включені в самі рекомендації ККСППР. Ця інформація може бути реалізована як інформаційні кнопки, щоб не заважати роботі.

Обслуговування системи та контенту

Технічне обслуговування ККСППР є важливою, але часто нехтуваною частиною життєвого циклу ККСППР. Це включає технічне обслуговування систем, додатків і баз даних, що забезпечують роботу ККСППР. Іншим викликом є підтримка бази знань та її правил, які мають йти в ногу зі швидко мінливою природою медичної практики та клінічних настанов. Навіть найпередовіші заклади охорони здоров'я повідомляють про труднощі з підтримкою своїх систем в актуальному стані, оскільки знання неминуче змінюються. Набори порядку та алгоритмічні правила ККСППР були визначені як особливо складні.

Вплив на роботу через низьку якість даних і неправильний вміст

EHR та ККСППР покладаються на дані із зовнішніх динамічних систем, і це може створити нові недоліки. Наприклад, деякі модулі ККСППР можуть заохочувати замовлення, навіть якщо в лікарні бракує відповідних матеріалів. У дослідженні Ash et al. ряд експертів зазначили, що в їхній лікарні тести Hemoccult або запаси пневмококової вакцини швидко закінчуються, але це не повідомляється ККСППР.

Списки ліків і проблем можуть бути проблематичними, якщо їх не оновлювати або використовувати належним чином. На одному сайті список ліків може бути списком дозволів, що означає, що пацієнти можуть або не можуть їх приймати (і, отже, їх все одно потрібно запитувати особисто). 85 Інші списки ліків створюються лише на основі замовлень CPOE, тому все ще вимагається ручне підтвердження того, що пацієнти приймають ліки. Ідеальними є системи, які полегшують їх розрізнення. Це також основна сфера, де PHR можуть знайти рішення, збираючи дані про прихильність до лікування безпосередньо від пацієнтів.

У погано спроектованих системах користувачі можуть розробити обхідні шляхи, які компрометують дані, наприклад введення загальних або неправильних даних. База знань ККСППР залежить від централізованого великого сховища клінічних даних. Якість даних може впливати на якість підтримки прийняття рішень. Якщо збір або введення даних у систему є нестандартизованим, дані фактично пошкоджені. Ви можете розробити систему для використання на місці надання медичної допомоги, але якщо її застосувати до реальних середовищ і даних, вона не використовуватиметься належним чином. Важливість використання інформаційних стандартів, таких як ICD, SNOMED та інших, важко недооцінити.

Відсутність транспортабельності та сумісності

Незважаючи на постійний розвиток протягом більшої частини трьох десятиліть, ККСППР (і навіть EHR загалом) страждають від проблем сумісності. Багато ККСППР існують як громіздкі автономні системи або існують у системі, яка не може ефективно спілкуватися з іншими системами.

Чому так важко досягти транспортабельності? Крім складнощів програмування, які можуть ускладнити інтеграцію, різноманітність джерел клінічних даних є проблемою. 96 Існує небажання або передбачуваний ризик, пов’язаний із транспортуванням конфіденційної інформації про пацієнта. Позитивно те, що стандарти сумісності постійно розробляються та

вдосконалюються, такі як Health Level 7 (HL7) і Fast Healthcare Interoperability Resources (FHIR). Вони вже використовуються в комерційних постачальниках EHR. Декілька державних установ, медичних організацій та органів інформатики активно підтримують, а деякі навіть вимагають використання цих стандартів сумісності в системах охорони здоров'я.

Хмара також пропонує потенційне рішення для сумісності (та інших проблем EHR, таких як синхронізація даних, оновлення програмного забезпечення тощо). Хмарні EHR мають відкриту архітектуру, нові стандарти та гнучкіші можливості підключення до інших систем. Також поширена помилкова думка, що дані, які зберігаються в хмарі, більш вразливі. Це не обов'язково правда. Для зберігання даних у центрах зберігання високого рівня з розширеним шифруванням та іншими засобами захисту потрібні веб-електронні медичні картки. Вони повинні відповідати національним стандартам безпеки даних, зокрема Закону про перенесення та підзвітність медичного страхування (HIPAA) у США, Закону про захист персональної інформації та електронних документів (PIPEDA) у Канаді або Директиві про захист даних і Загальному регламенті захисту даних (GDPR) у Європа, щоб назвати декілька. Вони можуть бути такими ж безпечними (або такими ж вразливими), як і традиційна серверна архітектура. Насправді часто менше людей мають доступ до незашифрованих даних у хмарних центрах зберігання, ніж до записів на сервері.

Фінансові труднощі

До 74% тих, хто має ККСППР, сказали, що фінансова життєздатність залишається важкою. Початкові витрати на встановлення та інтеграцію нових систем можуть бути значними. Поточні витрати можуть і надалі залишатися проблемою на невизначений термін, оскільки новий персонал потребує навчання користуванню системою, а також необхідні оновлення системи, щоб не відставати від поточних знань.

Результати аналізу вартості впровадження ККСППР неоднозначні, суперечливі та рідкісні. Те, чи буде втручання економічно ефективним, залежить

від широкого спектру факторів, у тому числі специфічних для середовища, як політичних, так і технологічних. Оцінка витрат і вигод сама по собі може бути обмеженою через такі проблеми, як відсутність стандартизованих показників. 107 Це новий напрямок досліджень, і для покращення нашого розуміння фінансових наслідків ККСППР потрібно зробити багато роботи.

Висновок

Доведено, що ККСППР допомагає постачальникам медичних послуг у прийнятті різноманітних рішень і завдань з догляду за пацієнтами, і сьогодні вони активно та повсюдно підтримують надання якісної медичної допомоги. Деякі застосування ККСППР мають більше доказів, особливо ті, що базуються на СРОЕ. Підтримка ККСППР продовжує зростати в епоху електронних медичних записів, і попереду ще багато досягнень, включаючи взаємодію, швидкість і легкість розгортання та доступність. У той же час ми повинні залишатися пильними щодо можливих збоїв у ККСППР, які варіюються від простого непрацювання та марного витрачання ресурсів до втоми постачальників і погіршення якості обслуговування пацієнтів. Під час створення, впровадження та підтримки ККСППР необхідно вживати додаткових заходів обережності та сумлінного проектування. Частину цих міркувань було розглянуто в цьому огляді, але на практиці знадобиться подальший перегляд, особливо оскільки ККСППР продовжує розвиватися у складності завдяки прогресу в штучному інтелекті, сумісності та нових джерелах даних.

2.2 Обгрунтування вибору засобів для побудови системи та мови програмування

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

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

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

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи підтримки прийняття управлінських рішень.

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

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

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

Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Інтелектуальна система підтримки прийняття рішень (ІСППР)

Користувачі також можуть використовувати штучний інтелект (ШІ) у системах підтримки прийняття рішень. Інтелектуальний інтелект, що називається інтелектуальними системами підтримки прийняття рішень (ІСППР), видобуває та обробляє великі обсяги даних, щоб отримати розуміння та надати рекомендації для кращого прийняття рішень. Це робиться шляхом аналізу багатьох джерел даних і виявлення закономірностей, тенденцій і асоціацій, щоб імітувати здатність людини приймати рішення.

Розроблений, щоб діяти подібно до людини-консультанта, ІСППР збирає та аналізує дані для підтримки осіб, які приймають рішення, шляхом виявлення та усунення проблем, а також надання та оцінки можливих рішень. Компонент штучного інтелекту СППР максимально точно імітує людські можливості, водночас ефективніше обробляючи та аналізуючи інформацію як комп’ютерна система.

ІСППР може включати розширені можливості, такі як база знань, машинне навчання, аналіз даних та інтерфейс користувача. Приклади реалізації ІСППР включають гнучкі або інтелектуальні виробничі системи, інтелектуальні системи підтримки прийняття маркетингових рішень і медичні діагностичні системи.

Різні типи систем підтримки прийняття рішень можуть покращити можливості компанії приймати рішення в різних сферах.

Типи систем підтримки прийняття рішень

Системи підтримки прийняття рішень можна розбити на категорії, кожну з яких базується на їх первинних джерелах інформації.

СППР на основі даних

СППР на основі даних – це комп'ютерна програма, яка приймає рішення на основі даних із внутрішніх або зовнішніх баз даних. Як правило, СППР на основі даних використовує методи інтелектуального аналізу даних, щоб розрізняти тенденції та закономірності, що дає змогу передбачати майбутні події. Компанії часто використовують СППР на основі даних, щоб допомогти приймати рішення щодо запасів, продажів та інших бізнес-процесів. Деякі з них використовуються для прийняття рішень у державному секторі, наприклад, для прогнозування ймовірності злочинної поведінки в майбутньому.

СППР на основі моделі

Побудовані на базовій моделі прийняття рішень, керовані моделлю системи підтримки прийняття рішень налаштовуються відповідно до попередньо визначеного набору вимог користувача, щоб допомогти проаналізувати різні сценарії, які відповідають цим вимогам. Наприклад, керована моделлю СППР може допомогти в плануванні або розробці фінансових звітів.

Комунікаційні та групові СППР

Система підтримки групового прийняття рішень, що керується спілкуванням, використовує різноманітні інструменти спілкування, такі як електронна пошта, обмін миттєвими повідомленнями чи голосовий чат, щоб дозволити кільком особам працювати над одним завданням. Метою цього типу СППР є посилення співпраці між користувачами та системою та підвищення загальної ефективності та результативності системи.

СППР на основі знань

У цьому типі системи підтримки прийняття рішень дані, які керують системою, знаходяться в базі знань, яка постійно оновлюється та підтримується системою управління знаннями. СППР, що керується знаннями, надає користувачам інформацію, яка відповідає бізнес-процесам і знанням компанії.

СППР на основі документів

СППР, керована документами, – це тип системи керування інформацією,

яка використовує документи для отримання даних. СППР, керовані документами, дозволяють користувачам здійснювати пошук на веб-сторінках чи базах даних або знаходити певні пошукові терміни. Приклади документів, доступ до яких здійснюється за допомогою СППР на основі документів, включають політики та процедури, протоколи зустрічей і корпоративні записи. База знань є невід'ємною частиною системи підтримки прийняття рішень на основі знань.

Приклади систем підтримки прийняття рішень

Організації використовують системи підтримки прийняття рішень у кількох різних контекстах, зокрема:

– **GPS-маршрутизація.** Планування маршруту GPS є прикладом типового СППР. Він порівнює різні маршрути, враховуючи такі фактори, як відстань, час у дорозі та вартість. Навігаційна система GPS також дозволяє користувачам вибирати альтернативні маршрути, відображаючи їх на карті та надаючи покрокові інструкції.

– **Інформаційні панелі ERP.** Інформаційні панелі ERP (планування ресурсів підприємства) можуть використовувати систему підтримки прийняття рішень для візуалізації змін у виробничих і бізнес-процесах, моніторингу поточної ефективності бізнесу щодо поставлених цілей і визначення областей для вдосконалення. Інформаційні панелі ERP дозволяють власникам бізнесу переглядати найважливіші цифри та показники своєї компанії.

– **Система підтримки прийняття клінічних рішень.** Система підтримки клінічних рішень (ККСППР) – це програмне забезпечення, яке використовує розширені алгоритми прийняття рішень, щоб допомогти лікарям приймати найкращі медичні рішення. Медичні працівники часто використовують їх для інтерпретації записів пацієнтів і результатів аналізів, а також для розрахунку найкращого плану лікування. ККСППР у сфері охорони здоров'я може допомогти постачальникам визначати аномалії під час певних тестів, а також спостерігати за пацієнтами після певних процедур, щоб визначити, чи є у них будь-які побічні реакції.

3.2 Розробка структурної схеми

Управлінське рішення (УР) – це результат аналізу, прогнозування, оптимізації, економічного обґрунтування й вибору альтернативи з безлічі варіантів для досягнення конкретної мети менеджменту. Якість УР – це ступінь відповідності параметрів обраної альтернативи рішення певній системі характеристик, що задовольняє його розроблювачів і споживачів і можливість, що забезпечує, ефективної реалізації. До таких характеристик відносять: наукову обґрунтованість, своєчасність, несуперечність, адаптивність, реальність. Роль СППР не у тому, щоб замінити керівника, а у тому, щоб підвищити його ефективність.

У вітчизняній літературі спостерігається термінологічна плутанина, що виникла у зв'язку зі зміною розуміння терміна "система підтримки прийняття рішень". У сучасній англомовній літературі еквівалент цього поняття – Decision Support System (СППР). Раніше в період, приблизно, з початку 70-х до початку 90-х років публікувалися оригінальні й перевідні статті, у яких застосовувався інший англійський еквівалент – " Decision-Making Support System (DMSS)". Головне розходження в наступному: раніше під підтримкою прийняття рішень розумівся *інструментарій вироблення рекомендацій* для особи, що приймає рішення (ОПР), зараз те ж поняття означає *інструментарій підготовки даних* для ОПР.

У сучасній СППР повинні бути реалізовані й перший і другий інструментарій, причому вони повинні бути взаємозалежні. Тобто інструментарій підготовки даних повинен бути реалізований з обліком того, що ці дані, можливо, будуть потім використані інструментарієм вироблення рекомендацій. Об'єднання двох інструментаріїв у рамках однієї системи дає більше зручний спосіб взаємодії користувача зі СППР. У процедурі прийняття рішень менеджер може одночасно використовувати інформацію, отриману різними шляхами: будь то звичайні або

більше складні агреговані запити, отримані за допомогою OLAP-засобів, або «нові» знання, отримані в ході застосування методів і технологій ІАД.

OLAP (Online Analytical Processing – оперативна аналітична обробка)

Поряд з іншими засобами аналізу даних під час бурхливого розвитку комп'ютерних засобів з'явилась технологія багатомірного аналізу даних OLAP (Online Analytical Processing – оперативна аналітична обробка). Це спосіб управління і представлення даних у простий і зрозумілий для кінцевого користувача спосіб.

Призначення систем класу OLAP – забезпечити користувача гнучким інтуїтивно зрозумілим і простим доступом до даних. Наявність такого доступу дає змогу відмовитися від використання наперед визначених звітів, робить користувачів самодостатніми, незалежними від адміністраторів БД і програмістів. В основі концепції OLAP дані представлені у вигляді багатовимірного куба, причому користувач має змогу швидко згорнути або розгорнути дані забудь-яким виміром. Ця технологія не замінює, а доповнює традиційні реляційні бази даних з первинною інформацією[1].

OLAP на відміну від інших способів автоматизації бізнес-діяльності дає можливість отримати користувачу «на виході» не готове чітко структуроване рішення, що видається після включення раніше налаштованого майстра обробки форм, а своєрідний матеріал для творчої оцінки існуючої ситуації. Тому сфера застосування OLAP-аналізу звичайно обмежується менеджерським складом підприємств різних розмірів, якому доводиться часто займатися тактичними і стратегічними завданнями на зразок аналізу ключових показників діяльності та сценаріїв розвитку, маркетингового та фінансово-економічного аналізу груп товарів або послуг, а також є довгостроковим прогнозуванням роботи підприємства чи його підрозділів [5].

Всі OLAP-продукти характеризуються загальними принципами побудови[3]:

1. В якості зовнішнього інтерфейсу вони надають керовану динамічну таблицю. На вхід динамічної таблиці подається багатовимірний масив. Масив складається з даних двох типів: вимірювань і фактів. Вимірювання стають колонками і рядками динамічної таблиці. У них відображаються члени вимірювань. На перетині колонок і рядків розміщені факти.

2. Колонки та рядки є основними інструментами управління таблицею. З їх допомогою користувач може маніпулювати вихідними даними: міняти місцями рядки і колонки, встановлювати фільтри з вимірювань, деталізувати інформацію або навпаки узагальнювати її. При цьому проміжні і остаточні підсумки за фактами автоматично перераховуються. Виконання цих операцій забезпечується OLAP-машиною (або машиною OLAP-обчислень). Самі маніпуляції з даними носять назву OLAP-операцій.

3. Ще однією важливою стороною OLAP-аналізу є графічне відображення даних. Графік синхронізований з динамічною таблицею. Після виконання будь-OLAP-операції дані перераховуються, а графік перемальовується.

Інтегровані системи управління підприємством

Елементи автоматизованої підтримки прийняття рішень присутні в інтегрованих системах управління підприємством (ІСУП). Однак для промислових підприємств впровадження ІСУП є однієї з найбільш трудомістких, дорогих і тривалих програм розвитку з величезним ризиком невдалого закінчення проекту впровадження. Широко відомі результати дослідження фірми Standish Group, які свідчать, що 84% IT-проектів кінчаються зірваними строками, перевищенням бюджету або зовсім нічим. У той же час автоматизована підтримка рішень необхідна керівникам підприємств уже зараз, і тому зроблено вивід про необхідність розробки незалежних СППР, які можуть одержувати дані з існуючих на підприємстві автоматизованих систем.

Виявимо фактори, що стримують побудову сучасних СППР, що дозволяють автоматизувати підготовку даних для прийняття управлінських рішень. Самі головні з них:

– невірогідність даних;

– низька продуктивність при нестандартних запитах;

– неможливість перетворення різнорідних даних у єдину інформацію.

У роботі ухвалене рішення, знайдене Б. Інмоном і сформульоване у вигляді концепції СЗД. По його визначенню, сховище даних – це предметно-предметно-орієнтована, інтегрована, некоректуєма, залежна від часу колекція даних, призначена для підтримки прийняття управлінських рішенні.

Розглянемо сховища даних двох типів:

– корпоративні СЗД;

– вітрини даних (ВД).

СЗД і ВД будуються по подібних принципах і використовують практично ті самі технології. ВД мають менший розмір і обслуговують деяке один напрямок або аспект діяльності підприємства.

На основі проведених досліджень зроблений вивід про те, що сучасна СППР повинна складатися з наступних компонентів:

1. Оперативні джерела даних (можуть являти собою OLTP, корпоративні БД, зовнішні джерела).

2. Засоби переносу й трансформації даних (виконують збір, очищення й узгодження даних із джерел).

3. СУБД (високошвидкісна серверна СУБД), що дозволяє підтримувати багаторівневу систему зберігання даних, що складає із СЗД і безлічі ВД.

4. Засоби доступу й аналізу даних (дозволяють одержувати деталізовані дані, агреговані показники й закономірності).

5. Допоміжні компоненти: засобу проектування/розробки й засобу адміністрування.

Питання технологій розробки ІС, і СППР зокрема, багато в чому стали вже традиційними й увійшли в класичні підручники. У даній роботі особлива увага приділяється технологіям створення СЗД/ВД, які ще не досить досліджені.

Розглянемо структурну схему системи, що зображена на рисунку 3.1. Аналітичний програмний комплекс являє собою розподілену систему, у якій можна умовно виділити три частини:

– підсистему збору оперативних даних;

– підсистему зберігання даних;

– підсистему аналізу даних.

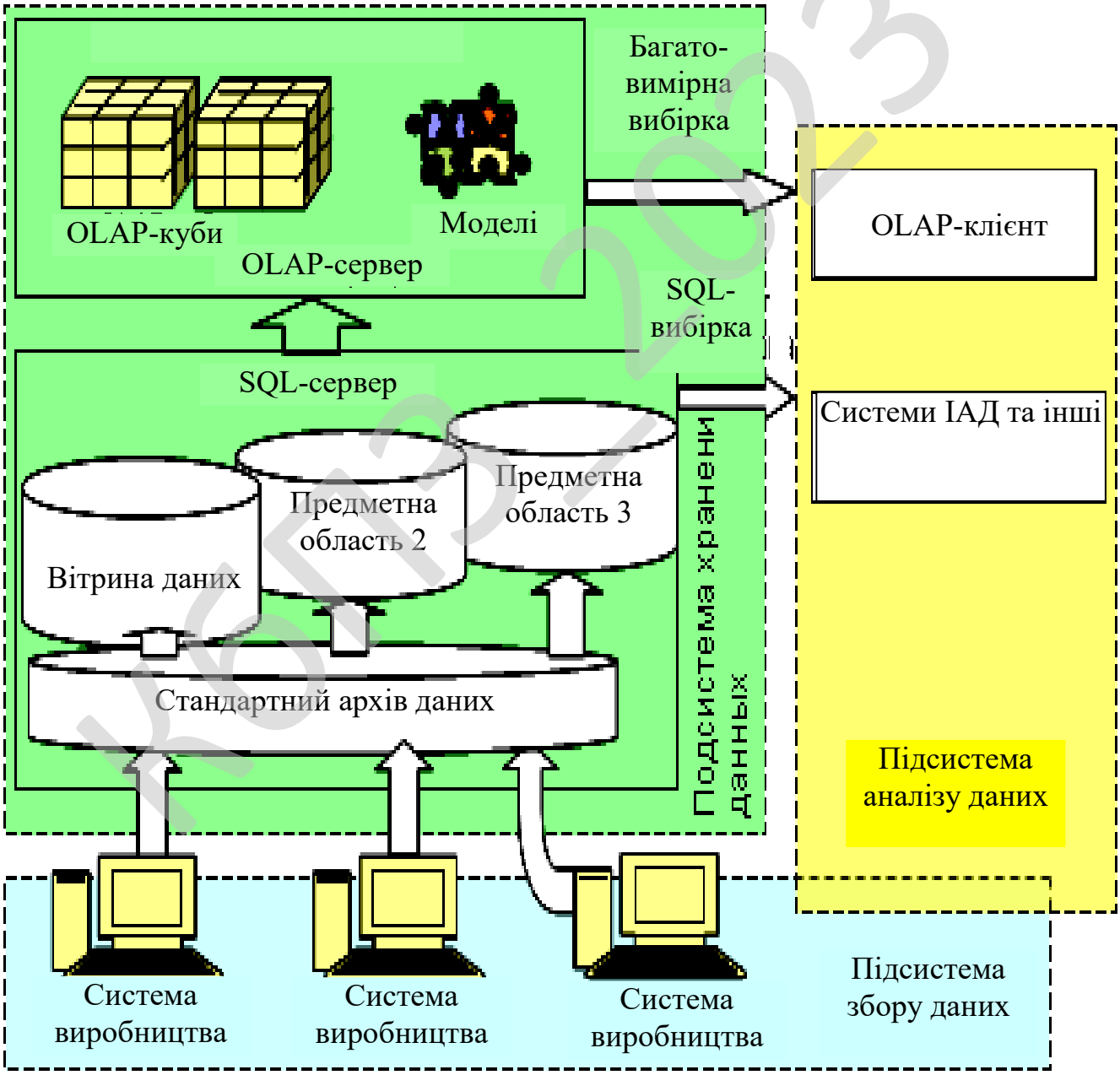

Рисунок 3.1 – Структурна схема системи

У підсистему збору даних включені будь-які джерела даних:

– оперативні системи підприємства;

– автоматизовані системи управління технологічними процесами;

– зовнішні джерела.

У зв'язку з низьким рівнем автоматизації найважливіших для оперативного управління виробництвом завдань, розробка аналітичного комплексу була почата зі створення системи обліку виробничих бізнес-процесів. Система має клієнт-серверну архітектуру. Специфіка операційних даних, що заносяться майстрами змін за допомогою системи – їх сувора періодичність. Облікові дані заносяться за підсумками зміни. Таким чином, дані мають строгу часову прив'язку, що характерно для СЗД. Наступною оперативною системою, розробленої для підприємства, стала система, що реалізує функції обліку даних по обслуговуванню, службою експлуатації заводу. У перспективі планується підключення інших оперативних БД інших підрозділів підприємства (склад, служба якості, бухгалтерія).

Підсистема зберігання даних виконує наступні функції:

– збір інформації з різних джерел, насамперед з оперативних інформаційних систем підприємства, що входять у підсистему збору даних, а також від зовнішніх джерел у сховище даних;

– інтеграція даних у логічні моделі по певних предметних областях на OLAP-сервері;

– зберігання інформації в OLAP-кубах таким чином, щоб вона була легко доступна й зрозуміла різним категоріям користувачів;

– надання даних різноманітним додаткам підсистеми аналізу даних.

У запропонованій структурі СЗД має дворівневу структуру:

– стандартний архів;

– вітрини даних (ВД).

Дані з оперативних джерел за допомогою розроблених програм завантажуються в стандартний архів даних, з якого вони вивантажуються у ВД, організовані відповідно до їхньої предметної спрямованості.

Із ВД формуються OLAP-куби, які надають зручні швидкодіючі засоби доступу, перегляду й аналізу ділової інформації. Користувач одержує природну, інтуїтивно зрозумілу модель даних, організувати їх у вигляді багатомірних кубів. Осями багатомірної системи координат служать основні атрибути аналізованого бізнес-процесу: час роботи зміни, номер зміни, характеристики встаткування й т.д. На перетинаннях осей – у термінології OLAP, називаних «вимірами» (Dimensions) – перебувають дані, що кількісно характеризують процес, – «міри» (Measures). Це можуть бути обсяги випущеної продукції, надходження, витрати й залишки матеріалів і ін.. Користувач може «розрізати» куб по різних вимірах, одержувати зведені (наприклад, по роках, кварталах, тижнях) або, навпаки, детальні (по змінах) відомості й здійснювати інші маніпуляції, необхідні в процесі аналізу.

За допомогою засобів OLAP-клієнта керівництво підприємства має можливість оперативно одержувати аналітичні звіти про випуск продукції й простої встаткування в характерні для даної інформації розрізах у різних одиницях виміру (кількість у штуках, коробках, кілограмах, відсотках і т.д.). На кнопках зі списками, що розкриваються, можна задавати умови вибору даних, а шляхом простого перетаскування полів задавати виміру. Дані можуть бути представлені в табличному й у графічному виді.

Візуалізація й агрегування даних, звичайно, полегшують прийняття управлінських рішень, однак засоби OLAP-аналізу не виробляють рекомендації ОПР. При прийнятті рішень корисне виявлення якісне нової інформації в даних, що втримуються в сховище, у вигляді схованих закономірностей, залежностей і взаємозв'язків, що забезпечується методами й засобами ІАД. Ілюстрація роботи інструментарію вироблення рекомендацій ОПР, як елемента нової технології, що реалізують її алгоритмів і програмних засобів була здійснена на конкретному завданні визначення складу ремонтних робіт методами імітаційного

моделювання, первинної статистичної обробки даних і статистичної перевірки гіпотез.

Одна з найважливіших проблем – це позапланові простої встаткування. У зв'язку з безперервним технологічним процесом, зупинка встаткування спричиняє зниження якості продукції й утворення дорогих відходів. Крім того, своєчасна заміна деяких деталей запобігає дорогому ремонту цілих вузлів. Вище сказане спричиняється актуальність завдання моделювання (прогнозування) позапланових простоїв устаткування.

Виділено основні **етапи рішення** поставленого завдання:

1. Оцінка статистичних характеристик досліджуваних вибіркових даних (середнє, мінімальне, максимальне, найбільш імовірне (модальне) час безвідмовної роботи й простоїв машин; середньоквадратичне відхилення, розкид часів і т.д.).

2. Дослідження законів розподілу часу безвідмовної роботи й часів простоїв машин по кожній технічній причині й підбор теоретичних розподілів, що адекватно описує вибіркові дані, на основі використання статистичних критеріїв згоди.

3. Імітаційне моделювання циклів роботи машин за допомогою календарного методу, коли і-цикл складається з ділянки робочого стану (інтервал часу роботи), за яким треба ділянка поломки (простою) внаслідок виникнення певної технічної причини.

4. Прогнозування моментів зупинки машини по певній технічній причині.

Впровадження розробленої системи дозволило скоротити трудомісткість стандартних операцій по обліку даних виробничих бізнес-процесів і формуванню стандартних аналітичних звітів. Середня кількість помилок, що виявляються в момент формування місячного звіту по випуску, скоротилося з п'яти до однієї на місяць.

3.3 Розробка функціональної схеми

Розглянемо функціональну модель системи, відображену на рисунку 3.2. З різних моделей життєвого циклу інформаційних систем при створенні СЗД була обрана спіральна модель. Це пов'язане з необхідністю виділення всіх необхідних даних для довільних запитів, для чого варто скласти вичерпний перелік необхідних даних і побудувати схему їхніх зв'язків. При цьому із загального масиву виділяється значима інформація й з'ясовується потреба в додаткових джерелах даних для прийняття рішень. Наша практика показала, що оптимальна тривалість ітерації становить 2-3 тижні при 4-5 ітераціях. На етапі експлуатації розвиток СЗД не припиняється. Це процес, що припиниться тільки в момент завершення життєвого циклу (ЖЦ) СЗД.

Технологічний процес побудови СЗД промислового підприємства пропонується представити як сукупність, і підпорядкованість необхідних етапів створення СЗД. Модель технологічного процесу $M_{ТП}$ може бути представлена в такий спосіб:

$$M_{ТП} = (M_{E1}, \dots, M_{E6}, \Phi_{E1}, \dots, \Phi_{E6}, P_{E1}, \dots, P_{E6}, \{O_{Eij}\}, \{R_{Eij}\}),$$

де $M_{E1}, \dots, M_{E6}$ – опису етапів технологічного процесу;

$\Phi_{E1}, \dots, \Phi_{E6}$ – вихідні фактори, що впливають на зміст і результати рішення завдань усередині етапу;

$P_{E1}, \dots, P_{E6}$ – результати рішення завдань кожного етапу;

$\{O_{Eij}\}$, $i,j \in (1,6)$ – набір відносин, що визначають підпорядкованість окремих етапів;

$\{R_{Eij}\}$, $i,j \in (1,6)$ – набір відносин, що визначають вплив вихідних факторів на результати рішення завдань окремих етапів.

Етапи, позначені в моделі як E1,...,E6, це:

1. Бізнес-аналіз процесів і даних підприємств.

2. Вибір типу OLAP-системи й відповідного ПЗ.

3. Вибір архітектури й способу побудови СЗД.

4. Проектування структури СЗД/ВД.

5. Створення сховища метаданих (репозитарія).

6. Завантаження СЗД/ВД.

Були систематизовані описи етапів побудови СЗД, узяті з літературних джерел, і власний досвід розробки. Відповідно до запропонованої моделі технологічного процесу $М_{ТП}$ для кожного з етапів, там, де можливо, були виділені:

– варіанти технічних рішень;

– фактори, що впливають на вибір;

– переваги й недоліки варіантів рішень;

– склад і форма результатів, які необхідно одержати по завершенні етапу.

Для підтримки вибору технічних рішень в умовах безлічі критеріїв і альтернатив був розроблений програмний пакет, що реалізує процедуру експертного оцінювання груповим ОПР на основі методів ранжирування й попарного порівняння.

Сам неформалізованим і складним є **перший етап** – бізнес-аналіз процесів і даних підприємств. Розробка аналітичної системи на основі СЗД без подібного аналізу заздалегідь приречена на невдачу – без ясного розуміння розроблювачами цілей бізнесу, способів їхнього досягнення, що виникають при цьому проблем і методів їхнього рішення, ресурси, необхідні для розробки СЗД, будуть витрачені зрячи.

Тому розглянуті джерела визначення інформаційних потреб, причому не тільки поточних, але й потенційно можливих. Ними є:

– інтерв'ю з керівниками й безпосередніми учасниками виділених бізнес-процесів;

– аналітичні звіти, формовані на підприємстві;

– документація по сертифікації підприємства по стандарті ICO 9000 (опис цілей і бізнес-процесів компанії, контрольних точок і характеристик, що

знімаються, бізнес-процесу в цих точках, заходів щодо забезпечення якості продукції).

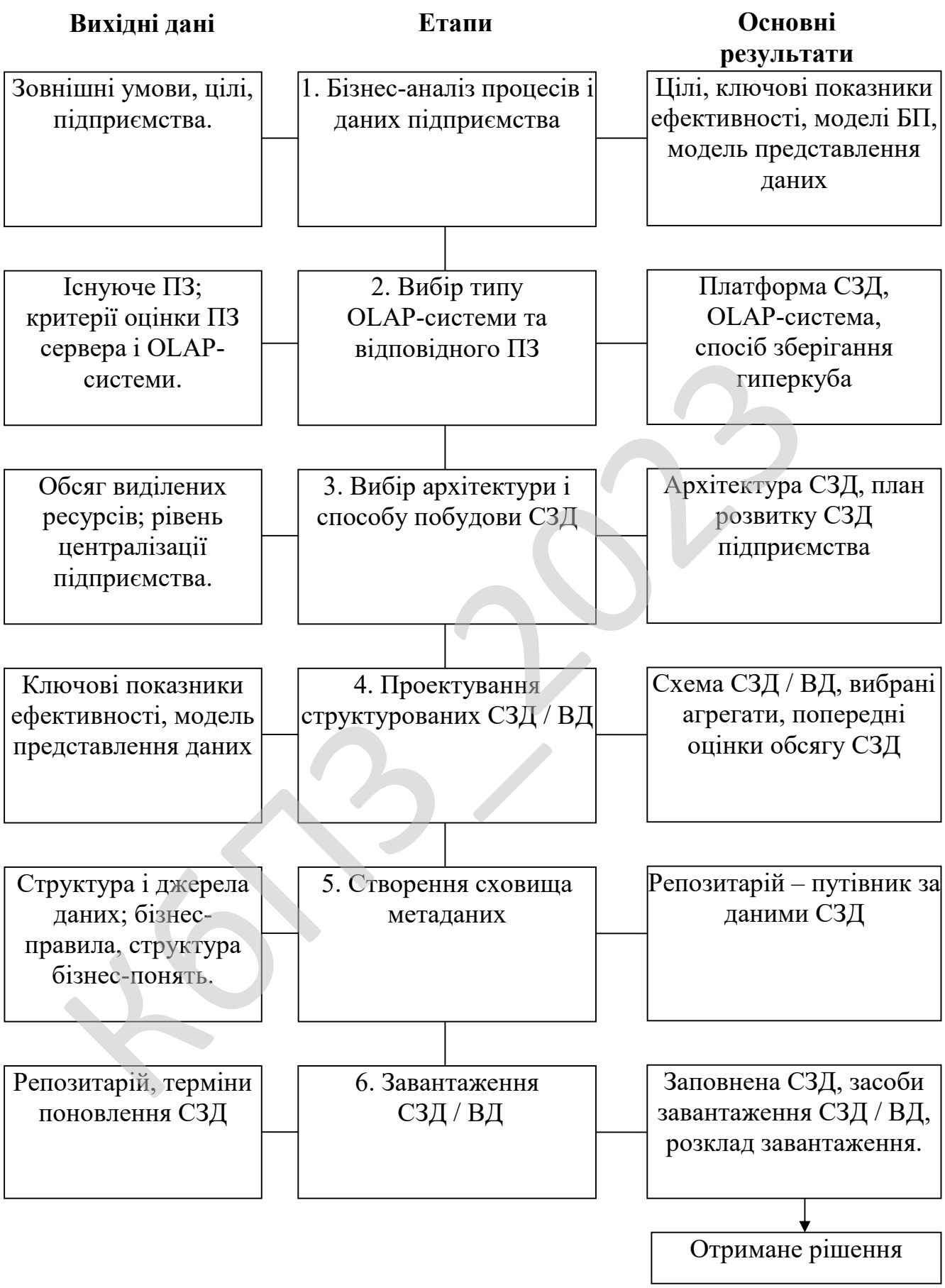

Рисунок 3.2 – Функціональна схема системи

Традиційні підходи до побудови інформаційно-аналітичних систем виходять із того, що на початку проекту складно сказати, що повинне бути поміщене в СЗД і які аналітичні завдання будуть вирішуватися кінцевими користувачами. Пропонується використовувати підхід, заснований на збалансованій системі показників (ЗСП). Підхід ЗСП дозволяє проектувати інформаційно-аналітичну систему зверху-долілиць і формувати в СЗД показники, що дають аналітикам цілісну картину розвитку підприємства по виділених напрямках.

Другий етап проекту пов'язаний з розумінням того, у якому виді й на яких апаратних і програмних платформах розміщати структуру даних СППР на основі СЗД. Запропоновано основні критерії вибору OLAP-системи:

– зручність і багатство можливостей засобів адміністрування;

– гнучкість налаштування й наочність форм демонстрації результатів OLAP-систем;

– спектр методів постобробки даних, доступність засобів інтелектуального аналізу;

– можливість обробки більших сховищ даних із прийнятною продуктивністю;

– можливість ув'язування OLAP-Інструментарію з усіма СУБД, використовуваними в організації;

– ціна.

В основі OLAP лежить поняття гіперкуба, або багатомірного куба даних. Залежно від відповіді на питання, чи існує гіперкуб як окрема фізична структура або лише як віртуальна модель даних, виділяють:

– системи MOLAP (Multidimensional OLAP);

– системи ROLAP (Relational OLAP);

– системи HOLAP (Hybrid OLAP).

Показано, що системи третього типу дозволяють сполучити компактне зберігання й підтримку дуже більших БД, забезпечувані ROLAP– системами із

простотою налаштування, і гарні часи відгуку при роботі з агрегованими даними, забезпечуваними MOLAP– системами. Саме вони й рекомендуються для побудови СППР.

На **третьому етапі** вибирається архітектура й спосіб побудови СЗД. Існує два основних способи побудови СЗД підприємства: "зверху долілиць" (*top down*) і "знизу нагору" (*bottom up*). Виявлено переваги й недоліки кожного. При підході "зверху долілиць" СЗД розробляється, проектується й будується ітераційним способом. Для підходу «зверху долілиць» характерні великі ресурси (для середньої компанії 3-4 роки при вартості 3-4 млн. дол.), високий ризик, забезпечення скоординованого середовища розробки.

При методі "знизу нагору" створюється ряд ВД, що поступово розвиваються, які формують основу результуючої системи СЗД підприємства. У порівнянні із СЗД при проектуванні ВД застосовують більше прості схеми БД, обсяги даних значно менше, процедури підтримки простіші, не потрібно такої високої деталізації, як у підході «зверху долілиць». У підсумку підхід "знизу нагору" характеризується швидким поверненням інвестицій, незначним ризиком, швидким розгортанням. На думку автора, саме цей подход, що рекомендується, реалізує величезний потенціал, властивий СЗД.

На наступних етапах відповідно до обраного підходу повинна бути заповнена архітектура ВД підприємства, поки відділи й організація не будуть готові побудувати СЗД підприємства.

На **четвертому етапі** при проектуванні структури СЗД/ВД необхідно використовувати схеми даних, що одержали назви "зірка" і "сніжинка". Відзначено, що це приводить до дублювання даних у СЗД, зниженню гнучкості структури й збільшенню часу завантаження Однак, все це – плата за ефективний і зручний доступ до даних, необхідний у СППР.

На **п'ятому етапі** пропонується розробка системи управління метаданими. У загальному випадку метадані містяться в централізовано керований репозитарій, у який включається інформація:

– про структуру дані сховища;

– про структури даних, імпортованих з різних джерел;

– про самі джерела, методи завантаження й агрегування даних;

– відомості про засоби доступу;

– бізнес-правила оцінки й подання інформації.

На **шостому етапі** при описі технології заповнення СЗД/ВД пропонується вирішувати три взаємозалежні завдання: добування даних (Data Acquisition), очищення даних (Data Cleansing) і агрегування даних (Data Consolidation). Завдання очищення й агрегування даних вирішуються шляхом реалізації програм, що запускаються в момент завантаження даних у сховище.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3.

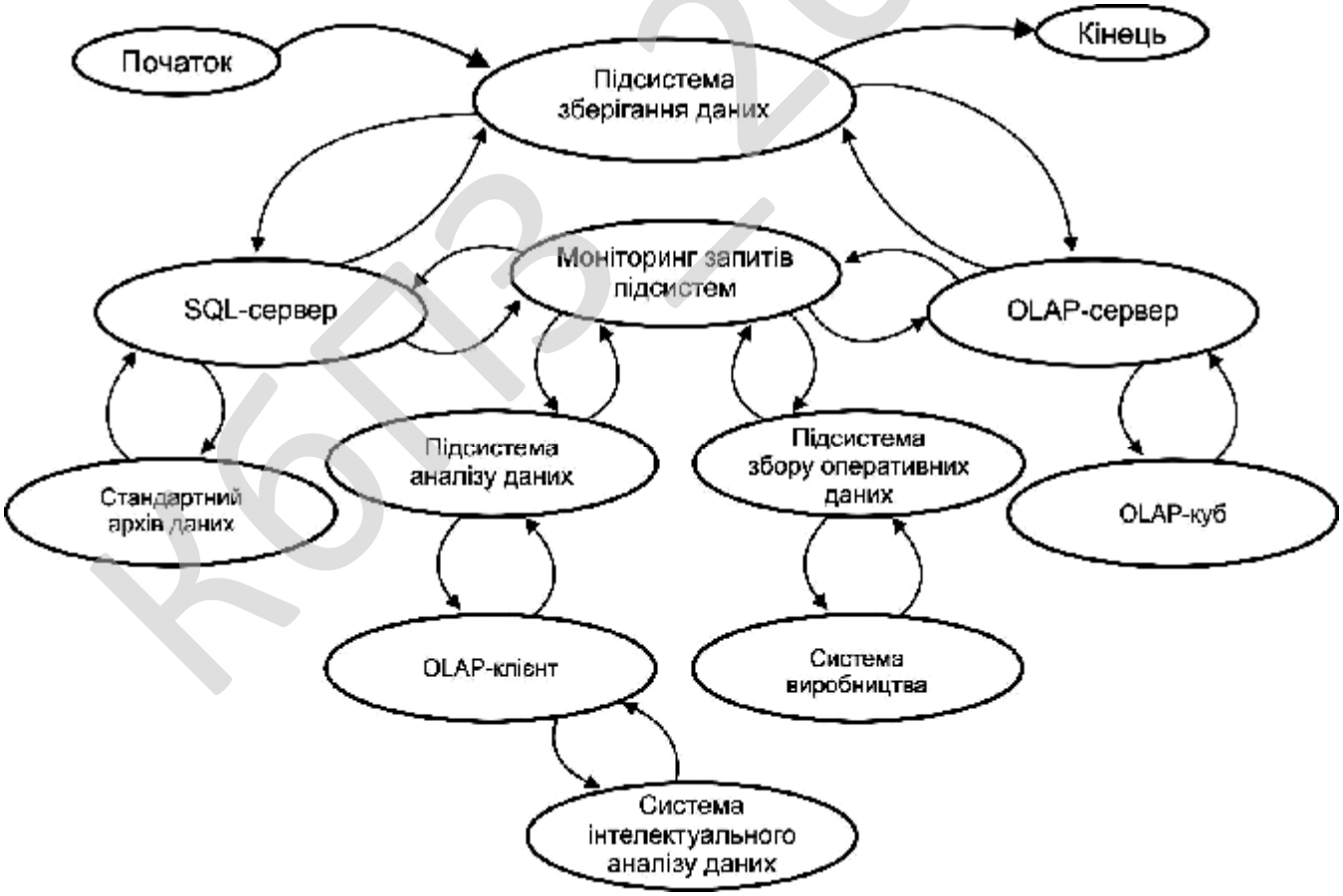

Рисунок 3.3 – Діаграма взаємодії процесів

Після початку роботи розробленого ПЗ ми потрапляємо до підсистеми зберігання даних, звідки можемо перейти до моніторингу запитів підсистем через SQL-сервер з стандартним архівом даних та через OLAP-сервер з OLAP-кубом. Далі через моніторинг запитів підсистем ми потрапляємо до підсистема збору оперативних даних з системою виробництва та до підсистеми аналізу даних, OLAP-клієнту, системи інтелектуального аналізу даних.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми спочатку відбувається ініціалізація ресурсів ПЗ. Потім здійснюється:

– Підпрограма перевірки стану серверів.

– Отримано код Er = -1 (запит).

– Отримано код Er = -2 (запит).

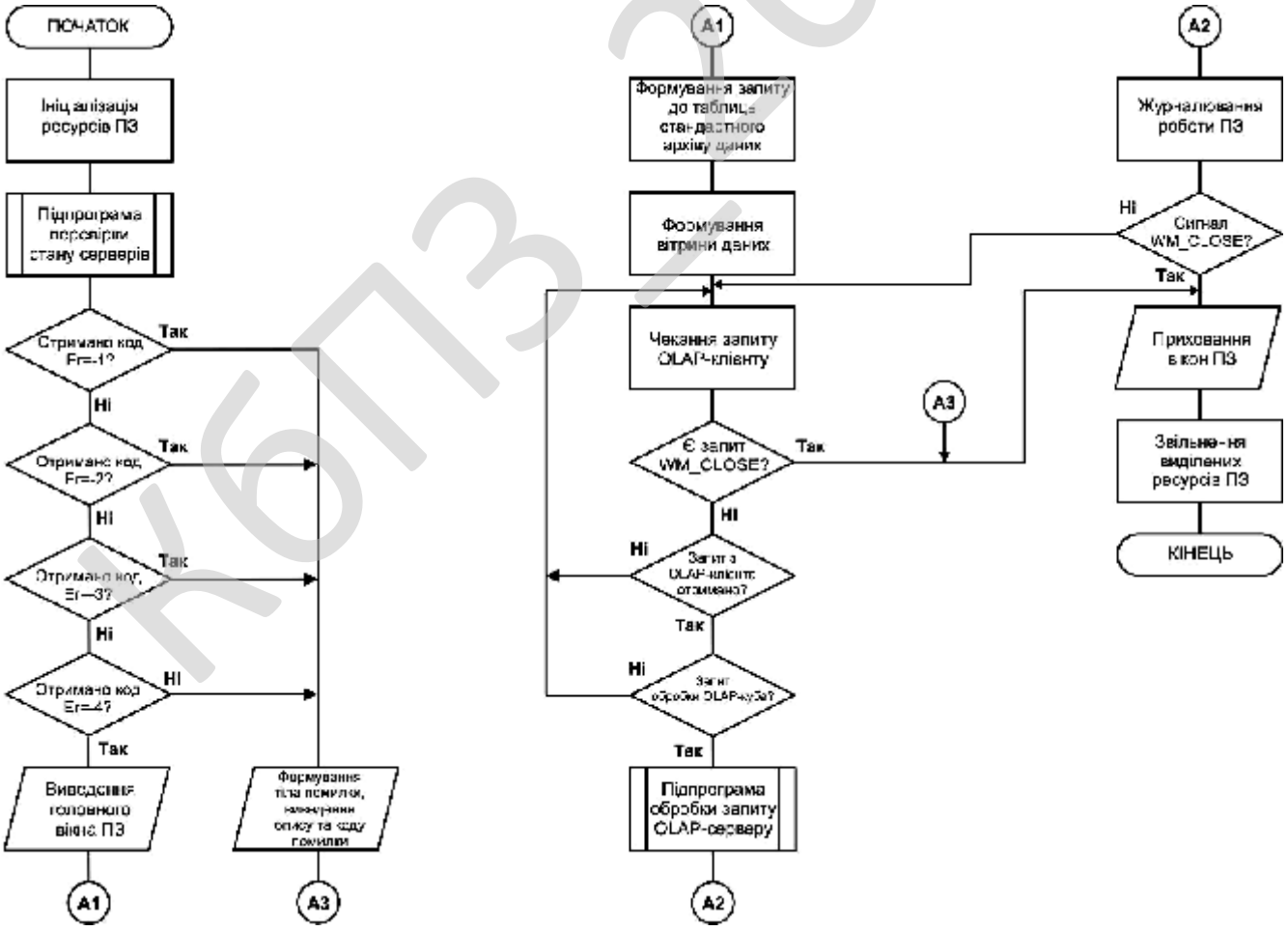

Рисунок 4.1 – Блок-схема основної програми

– Отримано код Er = -3 (запит).

– Отримано код Er = -4 (запит).

– Виведення головного вікна ПЗ.

– Формування запиту до таблиць стандартного архіву даних.

– Формування вітрини даних.

– Чекання запиту OLAP-клієнту.

– Є запит WM_CLOSE?

– Запит з OLAP-клієнта отримано?

– Запит обробки OLAP-куба?

– Підпрограма обробки запиту OLAP-серверу .

– Журналювання роботи ПЗ.

– Сигнал WM_CLOSE (запит).

– Приховання вікон ПЗ.

– Звільнення виділених ресурсів ПЗ.

На рисунку 4.2 зображено роботу підпрограми перевірки стану серверів. Де проходять наступні дії:

– Читання файлу налаштувань ПЗ.

– Файл налаштувань знайдено (запит).

– Встановлення значень по замовченню, встановлення коду повернення Er = -1 «Файл налаштувань».

– Спроба запиту до SQL-серверу.

– Запит оброблено?

– Встановлення коду повернення Er = -2 «SQL-сервер не відповідає».

– Спроба запиту до OLAP-серверу.

– Запит оброблено?

– Встановлення коду повернення Er = -3 «OLAP-сервер не відповідає».

– Спроба запиту OLAP-куба.

– Запит оброблено?

– Встановлення коду повернення Er = -4 «Помилка OLAP-куба».

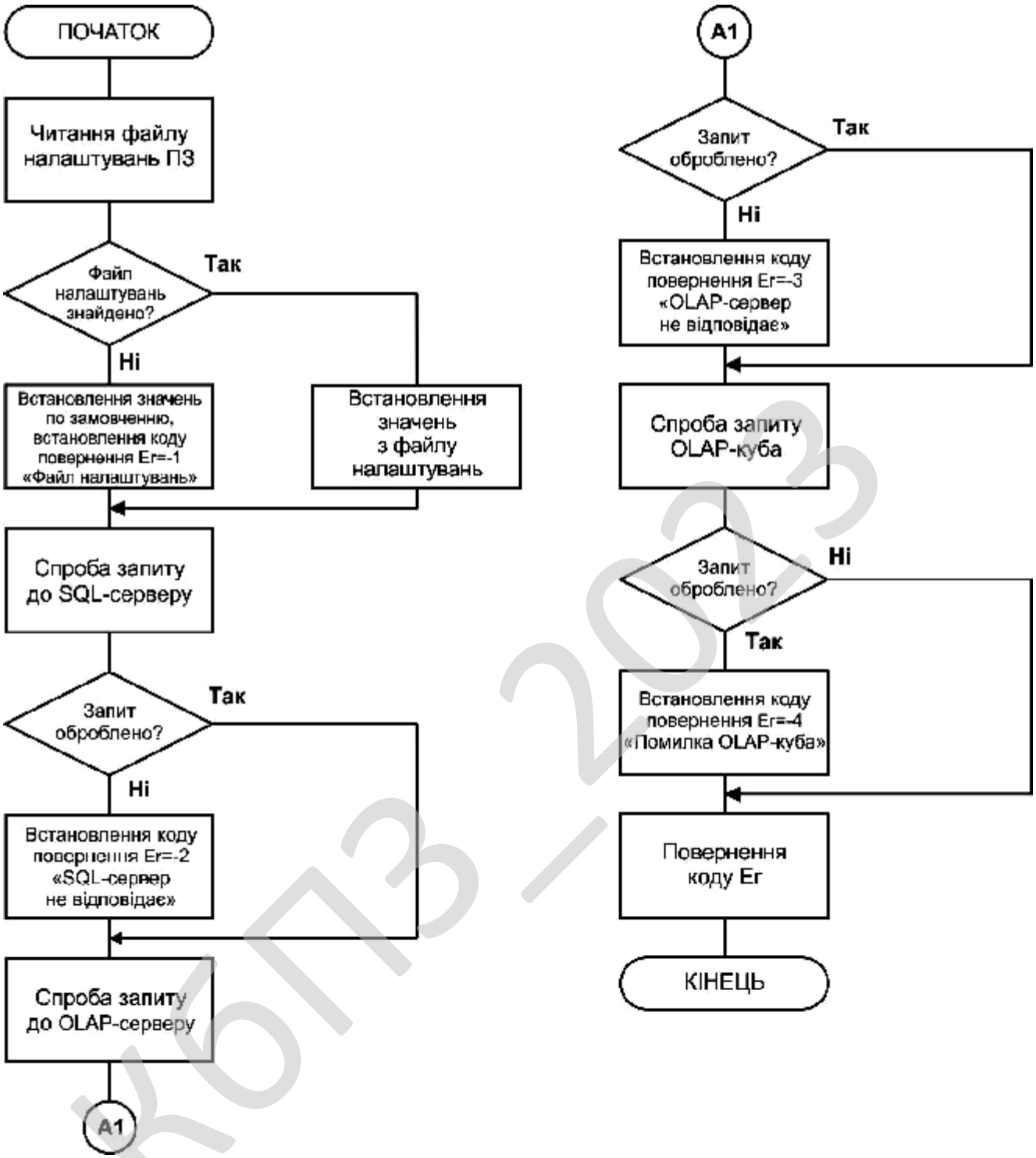

Рисунок 4.2 – Блок-схема підпрограми перевірки стану серверів

– Повернення коду Er.

На рисунку 4.3 зображено роботу підпрограми обробки запиту OLAP-серверу. Де проходять наступні дії:

– Виведення вікна моніторингу OLAP-серверу.

– Запит на з'єднання з вітриною даних.

– Вітрину даних сформовано (запит).

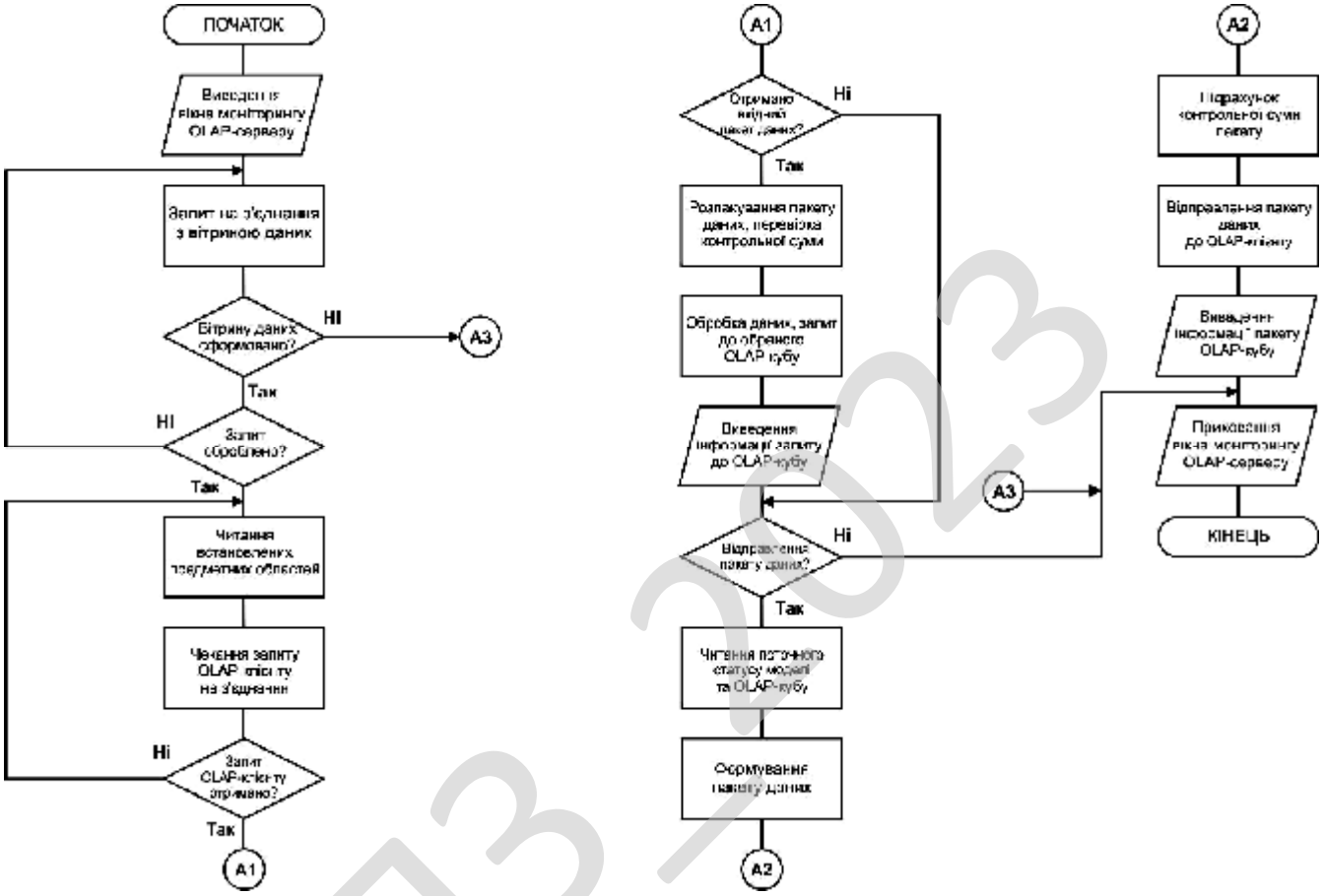

Рисунок 4.3 – Блок-схема підпрограми обробки запиту OLAP-серверу

– Запит оброблено?

– Читання встановлених предметних областей.

– Чекання запиту OLAP-клієнту на з'єднання.

– Запит OLAP-клієнту отримано?

– Отримано вхідний пакет даних?

– Розпакування пакету даних, перевірка контрольної суми.

– Обробка даних, запит до обраного OLAP-кубу.

– Виведення інформації запиту до OLAP-кубу.

– Відправлення пакету даних (запит).

– Читання поточного статусу моделі та OLAP-кубу.

– Формування пакету даних.

– Підрахунок контрольної суми пакету.

– Відправлення пакету даних до OLAP-клієнту.

– Виведення інформації пакету OLAP-кубу.

– Приховання вікна моніторингу OLAP-серверу.

Опис алгоритмів функціонування системи

Розглянемо розроблену частину коду ПЗ, за допомогою якого користувач зможе:

– переглядати метадані багатовимірної бази даних у вигляді ієрархічної структури;

– копіювати імена об'єктів багатовимірної бази даних і ключові слова MDX зі списку, заздалегідь заданого в редактор MDX- запитів;

– виконувати MDX- запит і копіювати результати в набір даних (компонент TClientDataSet) з метою представлення їх у компоненті TDBGrid.

Для цього створимо новий проект і помістимо його головну форму майбутнього програми компонента TToolBar з декількома кнопками, TTreeView, TListBox, TDBGrid, TClientDataSet, TDataSource і TMemo.

Потім встановимо значення властивості DataSource компонента DBGrid1 рівним DataSource1, а значення властивості DataSet компонента DataSource1 рівним ClientDataSet1. Компонент ListBox1 слід заповнити ключовими словами MDX, такими як CHILDREN, MEMBERS, DESCENDANTS, та ін..

Далі слід послатися в нашому додатку на бібліотеку типів ADO MD, що міститься у файлі MSADOMD. DLL, так як ADO MD не підтримується Delphi 2010 на рівні компонентів. Для цього слід вибрати пункт Project | Import Type Library з головного меню середовища розробки, а потім вибрати Microsoft ActiveX Data Objects (Multi- dimensional) 1.0 Library зі списку доступних бібліотек типів.

Зверніть увагу на те, що якщо ви вже імпортували бібліотеку типів ADOX і не перейменували клас Delphi 2010 для об'єкта ADOX Catalog, то клас Delphi 2010 TCatalog виявиться вже певним. У цьому випадку, щоб уникнути конфлікту імен можна перейменувати TCatalog в TADOMDCatalog. Бажано також переконатися, що опція Generate Component Wrapper не вибрана, так як нам потрібно створити тільки *.pas – файл для доступу до об'єктів ADO MD.

Далі можна натиснути кнопку Create Unit, що призведе до створення файлу ADOMD_TLB.PAS, що представляє собою інтерфейсний модуль до бібліотеки типів ADO MD.

Нарешті, нам потрібно включити посилання на цей файл в пропозицію Uses, так само як і посилання на модулі ComObj і ADODB.

Наступний крок – з'єднання з багатовимірної базою даних. Параметр ConnectionString, використовуваний для цієї мети, повинен посилатися на OLE DB Provider for OLAP Services (варто переконатися, що він дійсно встановлений), а також на ім'я комп'ютера та ім'я бази даних, наприклад:

```
DS: = 'Provider = MSOLAP.1;
Data Source = localhost;
Initial Catalog = FoodMart ';
```

Можна також з'єднатися з локальними кубами, створеними за допомогою Microsoft Excel і збереженими в файлах *.Cub. У цьому випадку параметр Connection String може виглядати так:

```
DS: =  ' Provider = MSOLAP.1;
Data Source = C:\Data\Cubes\NW1.cub ';
```

Код, що відповідає за з'єднання з базою даних, можна помістити в обробник події OnClick однієї з кнопок.

Цей код, а також процедура, що заповнює компонент TreeView1 іменами кубів, представлені нижче:

```
procedure TForm1.Button1Click (Sender: TObject);
begin
DS: =  ' Provider = MSOLAP.1;Data Source = localhost; ' +
     ' Initial Catalog = FoodMart ';
 FillTreeView (DS);
 end;
```

```
procedure TForm1.FillTreeView (DataSource: WideString);
 var
 I: Integer;
 begin
// Створимо новий об'єкт Catalog
 Catalog1: =  CoCatalog.Create;
 TreeView1.Items.Clear;
 RootNode: =  TreeView1.Items.Add (nil, ' Catalog ');
// З'єднаємося з багатовимірної базою даних
 Catalog1._Set_ActiveConnection (OleVariant (DataSource));
// Послідовно отримаємо імена всіх кубів на базі даних
 for I: =  0 to Catalog1.CubeDefs.Count – 1 do
 begin
    CubeDef1: =  Catalog1.CubeDefs[I] as CubeDef;
  CubeDefNode: =  TreeView1.Items.AddChild (RootNode, CubeDef1.Name);
end;
end;
```

Тут ми єднаємося з базою даних, створюємо об'єкт Catalog, переглядаємо по черзі всі елементи його колекції CubeDefs і витягаємо імена кубів (вони містяться у властивості Name об'єктів CubeDef).

Реальна обробка метаданих куба реалізована в обробнику події OnMouseDown компонента TreeView1:

```
procedure TForm1.TreeView1MouseDown (Sender: TObject; Button: TMouseButton;
                                     Shift: TShiftState; X, Y: Integer);
var
     HitTest: THitTests;
     CurrNode: TTreeNode;
     I: integer;
     NodeName: string;
    AddString: String;
begin
HitTest: =  TreeView1.GetHitTestInfoAt (X, Y);
// Якщо користувач клацнув мишею на одній з гілок TTreeView
 if (htOnItem in HitTest) then
 begin
    CurrNode: =  TreeView1.GetNodeAt (X, Y);
// Якщо у гілки можуть бути дочірні гілки, але вони ще не додані
// додамо їх
    if ((CurrNode.Count = 0) and (CurrNode.Level < 4)) then
    begin
```

```
    case CurrNode.Level of
// Ця гілка представляє куб
    1: begin
          CubeDef1: =  Catalog1.CubeDefs.Get_Item (CurrNode.Text);
// Отримуємо імена всіх розмірностей куба
          for I: =  0 to CubeDef1.Dimensions.Count - 1 do
          begin
           Dimension1: =  CubeDef1.Dimensions[I] as Dimension;
           DimNode: =  TreeView1.Items.AddChild (CurrNode,
                     Dimension1.Name);
          end;
        end;
// Ця гілка представляє розмірність
    2: begin
   CubeDef1: =  Catalog1.CubeDefs.Get_Item (CurrNode.Parent.Text);
    Dimension1: =  CubeDef1.Dimensions.Get_Item (CurrNode.Text);
// Отримуємо імена всіх рівнів ієрархії даної розмірності
for I: =  0 to Dimension1.Hierarchies[0].Levels.Count - 1 do
          begin
     Level1: =  Dimension1.Hierarchies[0].Levels[ i ] as Level;
     LevelNode: =  TreeView1.Items.AddChild (CurrNode, Level1.Name);
          end;
        end;
// Ця гілка представляє рівень ієрархії
    3: begin
       CubeDef1: =  Catalog1.CubeDefs.Get_Item (
                  CurrNode.Parent.Parent.Text);
        Dimension1: =  CubeDef1.Dimensions.Get_Item (
                  CurrNode.Parent.Text);
       Level1: =  Dimension1.Hierarchies[0].Levels.Get_Item (
                  CurrNode.Text);
// Отримуємо імена всіх членів даного рівня ієрархії
          for I: =  0 to Level1.Members.Count - 1 do
          begin
           Member1: =  Level1.Members[I] as Member;
           MemberNode: =  TreeView1.Items.AddChild (CurrNode,
                     Member1.Name);
          end;
        end;
      end;
    end
    else
```

```
// Якщо дана гілка вже має дочірні гілки (або їх не повинно бути),
// скопіюємо ім'я об'єкта в редактор MDX- запитів
     begin
// Якщо вітка не коренева
       if Currnode.Level > 0 then
       begin
        CurrNode: =  TreeView1.GetNodeAt (X, Y);
        NodeName: =  CurrNode.Text;
// Копіюємо ім'я гілки, сформатував відповідно
// з синтаксисом MDX, в редактор MDX- запитів
        if ((CurrNode.Level = 1) or (CurrNode.Parent.Parent.Text =
           ' Measures '))
        then AddString: =  '[' + NodeName + '] '
        else AddString: =  '[' + NodeName + '].';
        Memo1.SetSelTextBuf (PChar (AddString));
     end;
    end;
 end;
 end;
 procedure TForm1.ListBox1Click (Sender: TObject);
 var AddString: string;
 begin
// Додамо ключове слово MDX зі списку в редактор MDX- запитів
 AddString: =  Listbox1.Items[ Listbox1.ItemIndex ] + '';
 Memo1.SetSelTextBuf (PChar (AddString));
 end;
```

Тут ми визначаємо, що саме представляє гілку, на якій користувач клацнув мишею: куб, розмірність, рівень ієрархії, член рівня), використовуючи її властивість Level, а також з'ясовуємо, чи є вже у неї дочірні гілки.

Якщо дочірні гілки відсутні (властивість Count даної гілки дорівнює нулю), ми звертаємося до бази даних і створюємо відповідні дочірні гілки, використовуючи властивість Name відповідного об'єкта ADO MD.

Якщо ж з бази даних вже нічого завантажувати, ми копіюємо ім'я об'єкта, представленого даної гілкою, в компонент Memo1, в те місце, де знаходиться курсор.

Код для копіювання ключових слів MDX в компонент Memo1 наведено в цьому ж фрагменті коду. Таким чином, ми отримали інструмент для перегляду

метаданих куба та створення тексту MDX- запитів за допомогою клацань миші на гілках дерева об'єктів ADO MD на елементах списку ключових слів.

Наступний крок у створенні OLAP-клієнта полягає у виконанні MDX - запиту, що міститься в компоненті Memo1, і в заповненні компонента TClientDataSet його результатами. Ця функціональність реалізована у процедурі CDSFill, наведеної нижче:

```
procedure TForm1.Button2Click (Sender: TObject);
begin
 CDSFill (DS);
 end;
procedure TForm1.CDSFill (DataSource: WideString);
 var
 I, J: Integer;
 V: OleVariant;
begin
// Створимо новий об'єкт CellSet
 CellSet1: =  CoCellSet.Create;
 try
// Виконаємо MDX – запит, що міститься в компоненті Memo1,
// і відкриємо об'єкт CellSet
CellSet1.Open (Memo1.Text, DataSource);
with ClientDataSet1 do
begin
    Close;
    with FieldDefs do
    begin
// Знищимо всі визначення полів у ClientDataset
     Clear;
// Додамо нові визначення полів
// Перше поле потрібно для поранення імен рядків
     with AddFieldDef do
     begin
       Name: =  ' Rows ';
       DataType: =  ftString;
     end;
// Перебираємо колекцію Positions перший осі
     for I: =  1 to CellSet1.Axes[0].Positions.Count do
     begin
       with AddFieldDef do
```

```
        begin
// Значення поля вихідної бази даних стане ім'ям колонки
      Name: =  CellSet1.Axes[0].Positions[I - 1 ].Members[0].
               Caption +(' + IntToStr (CellSet1.Axes[0].Positions[I
               - 1 ].Ordinal) + ')';
// Імена колонок в наборах даних повинні бути унікальні, тому
// додамо унікальне число, що міститься у властивості Ordinal
// об'єкта Position, до значення поля
        DataType: =  ftFloat;
        end;
      end;
     end;
// Створюємо і відкриваємо ClientDataSet
     CreateDataSet;
     Open;
// Додаємо до нього записи
     for J: =  1 to CellSet1.Axes[ 1 ].Positions.Count do
     begin
// Додаємо запис
      Append;
// Додаємо ім'я рядка, використовуючи колекцію Position другої осі
      Fields[0].Value: =  CellSet1.Axes[ 1 ].Positions[ J - 1 ].
               Members[0].Caption;
// Перебираємо клітинки в рядку, витягуючи з них дані
      for I: =  1 to CellSet1.Axes[0].Positions.Count do
      begin
// Створюємо масив координат осередків
        V: =  VarArrayCreate ([ 0,1 ], varVariant);
        V[ 0 ]: =  I - 1;
        V[ 1 ]: =  J - 1;
// Якщо відповідна комірка в CellSet не порожня,
if CellSet1.Item[ PSafeArray (TVarData (V).VArray)].
      FormattedValue < > ''
       then
// Значення поля дорівнюватиме значенню в комірці
         Fields[I].Value: =  Cellset1.Item[ PSafeArray (TVarData
         (V).VArray)].Value
        else
         // інакше помістимо в поле нульове значення
         ClientDataSet1.Fields[I].Value: =  0;
      end;
     end;
```

```
// Закриваємо Cellset і вивільняємо ресурси
    CellSet1.Close;
    CellSet1: =  nil;
 end;
 except
    ShowMessage (' Invalid MDX Query ');
 end;
 end;
```

У цьому фрагменті коду ми створюємо об'єкт CellSet і використовуємо його метод Open. Якщо MDX – запит коректний, буде створений порожній набір даних типу TClientDataset з іменами полів, рівними властивості Caption перших елементів колекцій Members, які є властивостями колекції Positions перший осі (Axis[0]).

Зверніть увагу на те, що імена полів у наборах даних повинні бути унікальні. Однак у реальних багатовимірних базах даних властивість Caption членів колекції Members таким не є. Наприклад, в ієрархії Year/Month властивість Caption для членів колекції Members January 1999 і January 2000 дорівнюватиме одному і тому ж значенню January.

Існує багато способів уникнути дублювання імен полів, і в даному прикладі ми використовували найпростіший – додавання унікального числа, що міститься у властивості Ordinal об'єкта Member.

Після того як імена полів компонента ClientDataset1 визначені, виконується цикл перебору рядків об'єкта CellSet, і для кожного ряду ми встановлюємо значення першого поля рівним властивості Caption першого елемента колекції Members відповідного члена колекції Positions другої осі (Axis[1]), а потім поміщаємо значення з відповідних об'єктів Cell (доступних за допомогою колекції Item об'єкта CellSet) в решту поля.

У результаті виходить набір даних, заповнений двомірним перетином куба і відображений у компоненті DBGrid1.

Отже, таким чином можна переглядати OLAP – куби і виконувати MDX-запити, використовуючи ADO MD.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою MARS, який є блочно-симетричним шифром з відкритим ключем. Розмір блоку при шифруванні 128 біта, розмір ключа може варіюватися від 128 до 448 біт включно (кратні 32бітам). Творці прагнули поєднати в своєму алгоритмі швидкість кодування і стійкість шифру. В результаті вийшов один з самих криптостійкий алгоритм з алгоритмів, які брали участь в конкурсі AES.

Алгоритм унікальний тим, що використовував практично всі існуючі технології, застосовувані в криптоалгоритмах, а саме:

– Найпростіші операції (додавання, віднімання, виключаюче або).
– Підстановки з використанням таблиці замін.
– Фіксований циклічний зсув.
– Залежний від даних циклічний зсув.
– Множення за модулем 2^{32}.
– Ключове забілювання.

Використання подвійного перемішування представляє складність для криптоаналіз а, що деякі відносять до недоліків алгоритму. У той же час на даний момент не існує будь-яких ефективних атак на алгоритм, хоча деякі ключі можуть генерувати слабкі підключі.

Структура алгоритму

Автори шифру виходили з наступних припущень:

1. Вибір операцій. MARS був спроектований для використання на найсучасніших комп'ютерах того часу. Для досягнення найкращих захисних характеристик в нього були включені самі «сильні операції» підтримувані в них. Це дозволило добитися більшого відносини securityper-instruction для різних реалізації шифру.

2. Структура шифру. Двадцятирічний досвід роботи в області криптографії підштовхнув творців алгоритму до думки, що кожен раунд

шифрування грає свою роль в забезпеченні безпеки шифру. Зокрема, ми можемо бачити, що перший і останній раунди зазвичай сильно відрізняються від проміжних («центральних») раундів алгоритму в плані захисту від криптоаналітичних атак. Таким чином, при створенні MARSa використовувалася змішана структура, де перший і останній раунди шифрування істотно відрізняються від проміжних.

3. Аналіз. Швидше за все, алгоритм з гетерогенною структурою буде краще протистояти криптоаналітичних методам майбутнього, ніж алгоритм, всі раунди якого ідентичні. Розробники алгоритму MARS надали йому сильно гетерогенну структуру – раунди алгоритму дуже різняться між собою.

У шифрі MARS використовувалися такі методи шифрування:

1. Робота з 32-х бітними словами. Всі операції застосовуються до 32-бітовим словами. тобто вся початкова інформація розбивається на блоки по 32біта. (Якщо ж блок опинявся меншої довжини, то він доповнювався до 32біт)

2. Мережа Фейстеля. Творці шифру вважали, що це найкращий варіант поєднання швидкості шифрування і криптостійкості. В MARS використана мережа Фейстеля 3-го типу.

3. Симетричність алгоритму. Для стійкості шифру до різних атакам всі його раунди були зроблені повністю симетричними, тобто друга частина раунду є дзеркальне повторення першої його частини.

Структуру алгоритму MARS можна описати таким чином:

1. Попереднє накладення ключа: на 32-бітові субблоки A, B, C, D накладаються 4 фрагмента розширеного ключа k_0 … k_3 операцією складання за модулем 2^{32}.

2. Виконуються 8 раундів прямого перемішування (без участі ключа шифрування).

3. Виконуються 8 раундів прямого криптоперетворення.

4. Виконуються 8 раундів зворотного криптоперетворення. [2]

5. Виконуються 8 раундів зворотного перемішування, також без участі ключа шифрування.

6. Фінальне накладення фрагментів розширеного ключа k_{36} … k_{39} операцією віднімання за модулем 2^{32}.

Пряме перемішування

У першій фазі на кожне слово даних накладається слово ключа, а потім відбувається вісім раундів змішування згідно з мережею Фейстеля третього типу спільно з деякими додатковими змішування. У кожному раунді ми використовуємо одне слово даних (зване, вихідним словом) для модифікації трьох інших слів (звані, цільовими словами). Ми розглядаємо чотири байта вихідного слова як індексів на двох S-блоків, S_0 і S_1 , кожен, що складається з 256 32-розрядних слів, а далі проводимо операції XOR або додавання даних відповідного S-блоку в три інших слова.

Якщо чотири байти вихідного слова b_0 , b_1 , b_2, b_3 (де b_0 є першим байтом, а b_3 є старшим байтом), то ми використовуємо b_0 , b_2, як індекси в блоку S_0 і байти b_1 , b_3 , як індекси в S-блоці S_1 . Спочатку зробимо XOR S_0 до першого цільовим речі, а потім додамо S_1 до того ж слова. Ми також додаємо S_0 до другого цільовим слову і XOR блоку-S_1 до третього цільовим слову. У висновку, ми обертаємо вихідне слово на 24 біта вправо.

У наступному раунді ми обертаємо наявні у нас чотири слова: таким чином, нинішні перші цільове слово стає наступним вихідним словом, поточним другий цільове слово стає новим першим цільовим словом, третє цільове слово стає наступний другий цільовим словом, і поточне вихідне слово стає третім цільовим словом.

Більш того, після кожного з чотирьох раундів ми додаємо одне з цільових слів назад у вихідне слово. Зокрема, після першого і п'ятого раундів ми додав третю цільове слово назад у вихідне слово, а після другого і шостого раунду ми додаємо першої цільової слово назад у вихідне слово. Причиною цих додаткових операцій змішування, є ліквідація декількох простих диференціальних

криптоатаки в фазі перемішування, щоб порушити симетрію у фазі змішування та отримати швидкий потік.

Криптографічне ядро

Криптографічне ядро MARS – мережа Фейстеля 3-го типу, що містить в собі 16 раундів. У кожному раунді ми використовуємо ключову E-функцію, яка є комбінацією множень, обертань, а також звернень до S-блоків. Функція приймає на вхід слово даних, а повертає три слова, з якими згодом буде здійснена операція додавання або XOR до інших трьох слів даними. У доповненні вихідне слово обертається на 13 біт вліво.

Для забезпечення, серйозного опору до криптоатаки, три вихідних значення E-функції (O_1 , O_2, O_3) використовуються в перших восьми раундах і в останніх восьми раундах в різних порядках. У перші вісім раундів ми додаємо O_1 і O_2до першого і другого цільовим речі, відповідно, і XOR O_3 у третьому цільовим слову. За останні вісім раундів, ми додаємо O_1 і O_2 до третього і другого цільовим речі, відповідно, і XOR O_3 до першого цільовим слову.

E-функція

E-функція приймає як вхідні дані одне слово даних і використовує ще два ключових слова, виробляючи на виході три слова. У цій функції ми використовуємо три тимчасові змінні, що позначаються L, M і R (для лівої, середньої та правої).

Спочатку ми встановлюємо в R значення вихідного слова зміщеного на 13 біт вліво, а в M – сума вихідних слів і першого ключового слова. Потім ми використовуємо перші дев'ять бітів M як індекс до однієї з 512S-блоків (яке виходить суміщенням S_0 і S_1 змішуванням фази), і зберігаємо в L значення відповідного S-блоку.

Потім помножимо друге ключове слово на R, зберігши значення в R. Потім обертаємо R на 5 позицій вліво (так, 5 старших бітів стають 5 нижніми бітами R після обертання). Тоді ми робимо XOR R в L, а також переглядаємо п'ять нижніх біт R для визначення величини зсуву (від 0 до 31), і обертаємо M

вліво на цю величину. Далі ми обертаємо R ще на 5 позицій вліво і робимо XOR в L. У висновку, ми знову дивимося на 5 молодших бітів R, як на величину обертання і обертаємо L на цю величину вліво. Таким чином результат роботи E-функції – 3 слова (по порядку): L, M, R.

Зворотне перемішування

Зворотне перемішування практично збігається з прямим перемішуванням, за винятком того факту, що дані обробляються в зворотному порядку. Тобто, якби ми поєднали пряме і зворотне перемішування так, щоб їх виходи і входи були б з'єднані в зворотному порядку (D [0] прямого і D [3] зворотного, D [1] прямого і D [2] зворотного), то не побачили б результату перемішування. Як і в прямому змішування, тут ми теж використовуємо одне вихідне слово і три цільових. Розглянемо чотири перших байта вихідного слова: b_0 , b_1 , b_2, b_3 . Будемо використовувати b_0 , b_2як індекс до S-блоку – S_1 , а b_1 b_3 для S_0 . Зробимо XOR S_1 [b_0] в перше цільове слово, віднімемо S_0 [b_3] з другого слова, віднімемо S_1 [b_2] з третього цільового слів і потім проробимо XOR S_0 [b_1] також до третього цільового слова. Нарешті, ми повертаємо початкове слово на 24 позицій вліво. Для наступного раунду ми обертаємо наявні слова так, щоб нинішнє перше цільове слово стало наступним вихідним словом, поточне друге цільове слово стало першим цільовим словом, поточне третє цільове слово стало другим цільовим словом, і поточне вихідне слово стало третім цільовим словом. Крім того, перед одним з чотирьох «особливих» раундів ми віднімаємо одне з цільових слів з вихідного слова: перед четвертим і восьмим раундами ми віднімаємо першої цільової слово, перед третьому і сьомим раундами ми віднімемо третя цільова слово з вихідного.

Дешифрування

Процес декодування обернений процесу кодування. Код дешифрування схожий (але не ідентичний) на код шифрування.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

– Перегляд результатів: Цілі, ключові показники ефективності, моделі БП, модель представлення даних; Платформа СЗД, OLAP-система, спосіб зберігання гіперкуба; Архітектура СЗД, план розвитку СЗД підприємства; Схема СЗД / ВД, вибрані агрегати, попередні оцінки обсягу СЗД; Репозитарій – путівник за даними СЗД; Заповнена СЗД, засоби завантаження СЗД / ВД, розклад завантаження.

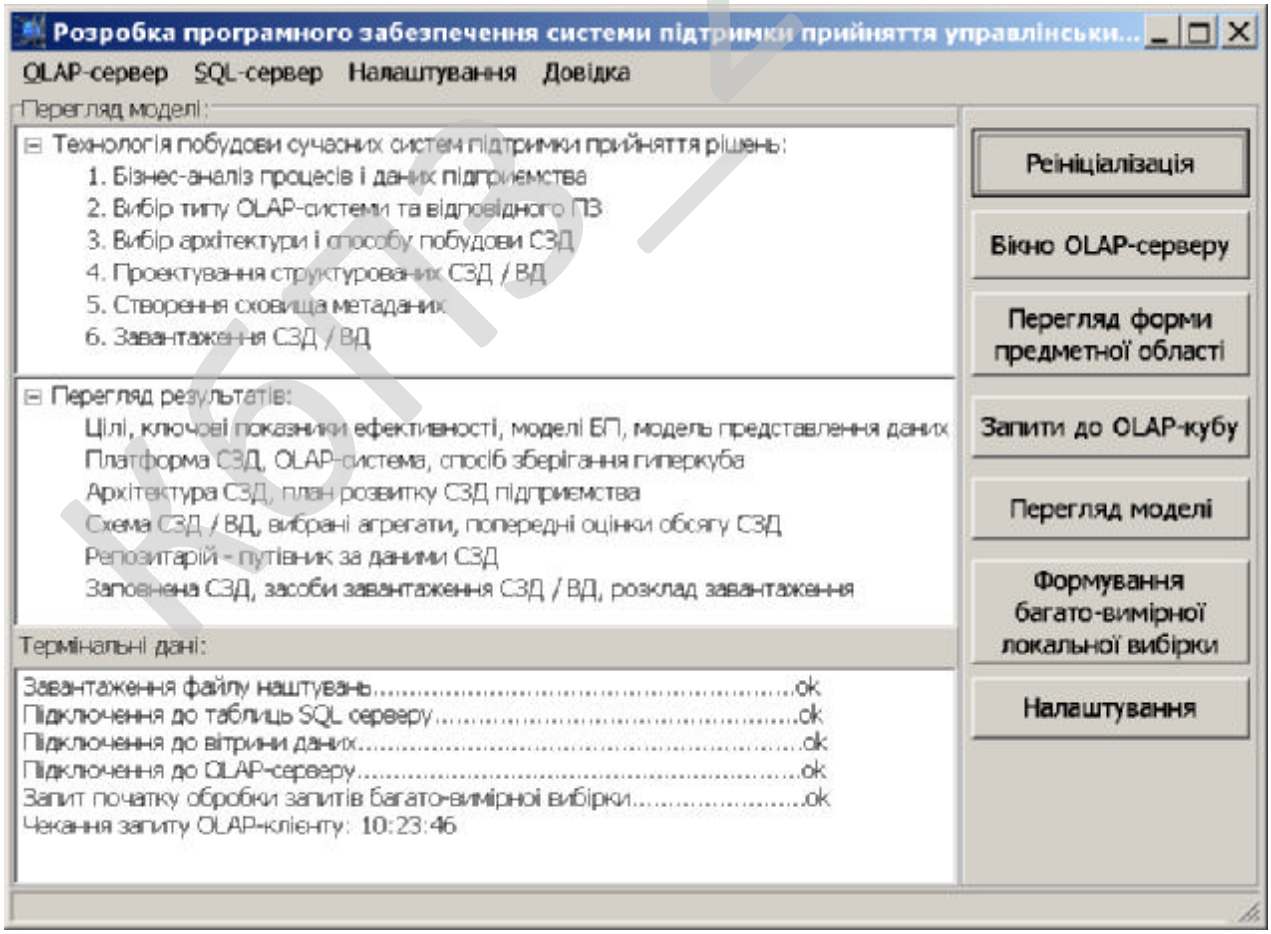

Рисунок 5.1 – Головне вікно програми

– Технологія побудови сучасних систем підтримки прийняття рішень: 1. Бізнес-аналіз процесів і даних підприємства; 2. Вибір типу OLAP-системи та відповідного ПЗ; 3. Вибір архітектури і способу побудови СЗД; 4. Проектування структурованих СЗД / ВД; 5. Створення сховища метаданих; 6. Завантаження СЗД / ВД.

– Термінальні дані (дій системи).

– Функції системи: Реініціалізація; Вікно OLAP-серверу; Перегляд форми предметної області; Запити до OLAP-кубу; Перегляд моделі; Формування багато-вимірної локальної вибірки; Налаштування.

На рисунку 5.2 зображено форму авторського права. Обрано ліцензію на програмне забезпечення – proprietary software тобто пропріетарне програмне забезпечення. Це ПЗ, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

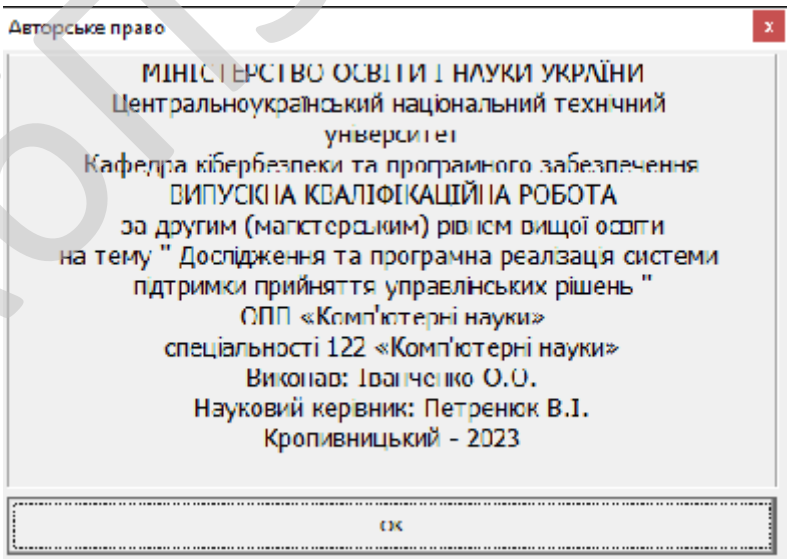

Рисунок 5.2 – Довідка розробника

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи підтримки прийняття управлінських рішень.

Метою розробки є дослідження та програмна реалізація системи підтримки прийняття управлінських рішень.

Об'єктом дослідження є процес підтримки прийняття управлінських рішень.

Предметом дослідження є методи підтримки прийняття управлінських рішень.

Методи дослідження базуються на методах теорії підтримки прийняття управлінських рішень, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод підтримки прийняття управлінських рішень.

– Розроблено вітчизняний продукт підтримки прийняття управлінських рішень, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

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

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи підтримки прийняття управлінських рішень.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	100
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	100000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

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

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

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

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

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

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 80 = 123 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	123	Ф 7.1-7.4
Впровадження	13	Д13
Всього	164	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{164 \cdot 1}{60 - 5} = 3 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	39,49

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{39,49 \cdot 3}{1,2} = 99 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 99/(60 \cdot 8) = 0{,}2 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	0,25	9500	7125
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3	8000	72000
Інженер - електронщик	0,2	7000	4200
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,5	7000	10500
Системний програміст	0,25	7000	5250
Дизайнер WEB	0,25	7000	5250
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,5	7000	10500
Всього за період розробки	$R_{сп}$ = 5,7	-	$Ф_{роб}$ = 131325

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{131325}{5,7 \cdot 60} = 384 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

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

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

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

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

$$Б_{уд} = 8 \cdot 8 \cdot 29000 = 1858000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 185800 грн.

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

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

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Компбест за 20.10.23 – джерело https://compbest.com.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i7-6700 (4 (8) ядра по 3.4 - 4. GHz), 8 MB Smart Cache	-
Системна плата	Fujitsu Esprimo P757 E90+ Tower, 2x PS/2, 4 USB 2.0, 6x USB 3.0, 1x USB Type-C, 2 DisplayPort, 1x DVI, 1x LAN (RJ-45), 4 Audio	-
Відеокарта	Вбудована Intel HD Graphics 530	-
Жорсткий диск	SSD: 240 Gb	-
Оперативна пам’ять	16 GB DDR4	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	-
Корпус	ATX Middle Tower, PSU 280(), black, (fron bezel – black+light silver; body material 0.6mm), 80mm fan (rear), 2x USB 2.0, 2 USB 3.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 3.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1858000	-	-
2. Передавальні пристрої	185800	-	-
Всього по групі	2043800	5	102190
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	100000	10	10000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	37857,75
Разом	$К_р$ = 2523008		$А_р$ = 249636,25

Примітка: вартість автомобіля Renault Scenic 2010 взята за даними електронного ресурсу, джерело https://auto.ria.com/uk/auto_renault_scenic_33598032.html, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 384 \cdot 164/100 = 629 \text{ грн.}$$

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

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

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 629 \cdot 10 \cdot 0{,}01 = 63 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

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

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(629+63) = 256 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 629 \cdot 15 \cdot 0{,}01 = 94 \text{ грн.}$$

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

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

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

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 1,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=206 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 206 \cdot 1,5 = 309 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 50):

$$З_{М2} = \sum Ц_д\,, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 24 грн./шт., DVD-R box – 39 грн./шт.

$$З_{М2} = 49 \cdot 24 + 1 \cdot 39 = 1215 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з.\,, \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

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

$$З_М = (309 + 1215 + 1702)/100 = 32 \text{ грн.}$$

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

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

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

$$О_п = 629 \cdot 15 \cdot 0,01 = 94 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 100 прим.):

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

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

$$A_м = 249636 \cdot 3/(100 \cdot 12) = 624 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	629
2. Додаткова зарплата виконавців	$З_д$	63
3. Відрахування на соціальні потреби	$С_{оц}$	256
4. Загальногосподарські витрати	$Г_{осп}$	94
5. Витрати на матеріали	$З_М$	32
6. Освоєння нових операційних систем, мов програмування	$О_п$	94
7. Амортизація основних фондів	$А_м$	624
8. Повна собівартість програмного забезпечення	$С_п$	1792
9. Плановий прибуток	$П_р$	896
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2688
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	537,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3225,6

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

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

$$C_п = 629+63+256+94+32+94+624 = 1792 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 1792 = 896 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3226
Всього капітальних витрат	–	3226

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування)	$З_р$	24156	5033
2. Витрати на електроенергію	$З_{ел}$	276	58
3. Витрати на амортизацію	$З_{ам}$	0	807
Всього витрат за рік	$І$	24432	5898

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 50 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 240 \cdot 75 \cdot 1{,}1 \cdot 1{,}22 = 24156\ грн,$$

до:

$$З_{р\ нов} = 50 \cdot 75 \cdot 1{,}1 \cdot 1{,}22 = 5033\ грн.$$

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

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

$$З_{ел\ баз} = 0{,}5 \cdot 240 \cdot 2{,}3 = 276 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}5 \cdot 50 \cdot 2{,}3 = 58 \text{ грн.}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3226	–	806,5
Всього відрахувань	-	–	3226	–	806,5

7.8 Визначення економічної ефективності програмної продукції

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

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

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

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

$$Ев = (2688\text{-}1792) \cdot 100\text{-}(0{,}05 \cdot 2043800 + 0{,}4 \cdot 199177 + 0{,}25 \cdot 37031 + $$
$$+ 0{,}1 \cdot 100000 + 0{,}2 \cdot 143000) \cdot 3/12 = 32170 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{479208}{(2688\text{-}1792)\cdot 100\cdot 12/3} = 1{,}3 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	100
2. Повна собівартість розробленої програми	Грн.	1792
3. Ціна розробленої програми	Грн.	2688
4. Плановий прибуток від реалізації розробленої програми	Грн.	896
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2523008
7. Загальний прибуток від реалізації програмної продукції	Грн.	89600
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	32170
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	17728
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,17

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (24432\text{-}5898)\text{-}0{,}25 \cdot 3226 = 17728 \text{ грн.}$$

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

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

$$T_{сп} = \frac{3226}{24432 - 5898} = 0{,}17 \text{ } року.$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1

8. ЗАХОДИ ЩОДО ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Характерною ознакою сучасного науково-технічного прогресу практично у всіх сферах діяльності людини є широке застосування комп'ютерних технологій, заснованих на використанні електронно-обчислювальних машин (ЕОМ). Сьогодення, а тим більше, майбутнє, вже важко уявити без комп'ютерів та іншої електронної техніки. Адже саме завдяки їм стала можливою швидка переробка величезних обсягів інформації, проведення необхідних розрахунків, виконання різних видів робіт, пов'язаних обробкою текстових та ілюстраційних зображень, організація оперативного отримання та передачі інформації, збереження її значних обсягів електронним способом.

Стрімке впровадження комп'ютерів не тільки в сфері управління виробництвом, в банківській систему, бізнесі, системі освіти, але також на транспорті, сфері обслуговування призвело до того, що десятки мільйонів людей у всьому світі виявились втягнутими у взаємодію людини з комп'ютером. Природно виникає запитання: настільки безпечною є ця взаємодія для людини? Адже відома аксіома про те, що будь-яка взаємодія людини та засобів праці двостороння.

Людина впливає на удосконалення засобів праці, а останні – на працюючу людину. Отже, навіть сучасні технології та техніка, до яких безперечно, залежать комп'ютерні технології та ЕОМ несуть у собі певні потенційні небезпеки. У зв'язку з цим набуває актуальності адекватна оцінка конкретних умов і характеру праці, яка сприяє обґрунтованому розробленню та впровадженню комплексу заходів і засобів, спрямованих на збереження здоров’я і працездатності людини в процесі

праці за рахунок поліпшення параметрів виробничого середовища, зменшення важкості, напруженості трудового процесу та збереження здоров'я працівників на комп'ютеризованих робочих місцях.

Законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп’ютером., зокрема «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», затверджені наказом Мінсоцполітики від 14.02.2018 № 207 [1], «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98. [2].

Загальні вимоги пожежної безпеки під час експлуатації комп’ютерної техніки визначають «Правила пожежної безпеки в Україні» (затверджені наказом МВС від 30.12.2014 № 1417) [3], комп’ютерних класів — пункт 3 розділу VIII «Правил пожежної безпеки для навчальних закладів та установ системи освіти України» (затверджені наказом МОН від 15.08.2016 № 974). [4] та інші державні стандарти, що регламентують експлуатування комп’ютерної техніки як радіоелектронної апаратури.

8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

Можна виділити наступні основні фактори, що впливають на стан здоров'я людей, які працюють за комп'ютером:

– сидяче положення на протязі тривалого періоду;

– вплив електромагнітного випромінювання монітора;

– втома очей, навантаження на зір;

– перевантаження суглобів кистей;

– стрес при втраті інформації.

У кожному з цих випадків ступінь ризику прямо пропорційний часу, що проводиться за комп’ютером і поблизу нього.В сучасних умовах взаємодія людини

з технікою значно ускладнилась, що вимагає комплексного підходу, який передбачає розгляд людини, технічних засобів праці та виробничого середовища, як взаємозв'язаних елементів єдиної системи. Все вищесказане в повній мірі відноситься й до системи «людина–комп'ютер–середовище».

Вагомий вплив на працездатність та здоров’я користувачів комп’ютерів здійснює виробниче середовище. Це середовище у виробничих приміщеннях(офісах), в основному, визначається мікрокліматом, освітленням, наявністю шкідливих речовин у повітрі, рівнем шуму, випромінювання.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам та запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини необхідно здійснити санітарно-гігієнічну характеристику умов працівника, який працює з програмним продуктом.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці користувача ПК

Розглянемо приміщення в якому працює користувач ПК з даним програмним продуктом.

Приміщення має одностороннє природне освітлення і загальне штучне освітлення. Стіни і стеля обклеєні світлими шпалерами, підлога вкрита темним ламінатом. У приміщенні відсутні сильні вібрації та шкідливі речовини. Склад повітря в нормі. У кімнаті знаходиться ПК з 4-ядерним процесором і 23-дюймовим IPS монітором, а також меблі.

Приміщення має довжину 4м, ширину 3,5м, висоту стелі 2,7м. Кількість робочих місць – одне. Площа – 14м^2, об'єм – 37,8м^3. Виходячи з цього, отримано дані, наведені в таблиці 8.1.

Таблиця 8.1–Фактичні та нормативні значення параметрів приміщення

Параметр	Норма *	Реальні параметри
Площа, S	не менше 6 м^2	14 м^2
Об'єм, V	не менше 20 м^3	37,8 м^3

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

За даними, які наведено у табл. 8.1, можна зробити висновок, що отримані показники, площа та об'єм приміщення у розрахунку на одно робоче місце користувача ПК відповідає чинним нормам і вимогам.

Щодо мікроклімату, то згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [5] роботу з ПК можна віднести до категорії легка 1а. Джерелами тепла в цьому приміщенні є люди, електроустаткування, освітлювальні прилади в темний час доби і система опалювання взимку. Оператором виділяється до 120ккал теплової енергії за годину. Оптимальні та фактичні значення параметрівмікроклімату приведені в таблиці 8.2.

Таблиця 8.2 – Значення параметрів мікроклімату

Період року	Параметр	Оптимальний*	Фактичний
Теплий	Температура	23 – 25 ^{0} С	24^0 С
	Вологість	40 – 60%	50%
	Швидкість повітря	< 0,1м/с	
Холодний	Температура	22 – 24^0 С	23^0 С
	Вологість	40 – 60%	55%
	Швидкість повітря	< 0,1м/с	

*ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»

По отриманим замірам параметрів мікроклімату можна зробити висновок, що усі показники задовольняють вимогам зазначеним для робіт категорії легка 1а і є задовільними для здоров'я людини.

Щодо освітлення , то згідно з ДБН В.2.5-28:2006 «Природне і штучне освітлення» [6] ця робота відноситься до Va розряду зорових робіт.
Передбачається використання природного, штучного і змішаного освітлення.

Природне освітлення здійснюється за допомогою вікна, площа якого складає $S' = 1,8*1,5 = 2,7м^2$ і є боковим освітленням. У світильниках місцевого і загального освітлення використовуються світлодіодні лампи потужністю 20Вт із світловим потоком лампи 900 лм. Згідно замірів рівень освітлення в даному приміщенні і на робочому місці складає в межах 350 -500 лк, що відповідає нормованому значенню

Джерелом шуму в приміщенні є комп'ютер. Вентилятори (кулери) системного блоку, процесора, відеокарти і блоку живлення є сучасними і мають низький рівень шуму. Згідно з технічною документацією шум, зумовлений кулером в блоці живлення складає 25 дБ, кулером процесора – 30 дБ, загальний -34 дБ. Враховуючи незначний рівень шуму від персонального комп'ютера і незначний рівень фонового шуму від іншого устаткування, можна стверджувати, що сумарний рівень шумового забруднення приміщення не перевищує максимально допустимий рівень коригованої звукової потужності і складає не більше 50 дБА, що відповідає рівню шуму для приміщень з комп’ютерною технікою згідно Державних санітарних правил і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

У приміщенні відсутні джерела інфрачервоного, ультрафіолетового і електромагнітного випромінювання, бо монітор ПК вироблений на основі рідкокристалічної матриці, підсвітка якої здійснюється неоновою лампою, що не має сильного електромагнітного випромінювання і сертифіковані в Україні.

Блок живлення є екранованим і не випускає вищезазначених видів випромінювання.

8.4 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці користувача ПК.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

Крім рекомендацій щодо конкретного приміщення, де було проведено дослідження умов праці, існують загальні вимоги, які зарекомендовані відповідними нормативними документами.

Правильна організація робочих місць запобігає передчасній втомлюваності користувача і сприяє збереженню здоров'я. Організація робочого місця передбачає:

- правильне розміщення робочого місця у виробничому приміщенні;
- вибір ергономічного обґрунтованого робочого положення, виробничих меблів з урахуванням характеристик людини;.
- раціональне компонування обладнання на робочих місцях;
- урахування характеру й особливостей трудової діяльності. Стосовно робочих місць користувача ВДТ, то організація робочого має забезпечуватися відповідно до ДСанПіН 3.3.2-007-98. Для запобігання перевтомленню необхідно виконувати вправи для очей та дотримуватись розпорядку роботи та відпочинку.

На робочому місці реалізовувався режим відпочинку: кожні дві години – перерва для виконання фізичних вправ для м‘язів очей.

8.5 Розрахункова частина

Система освітлення робочого місця користувача ПК має відповідати наступним вимогам (рис. 8.1).

Рисунок 8.1 – Вимоги до системи освітлення робочого місця користувача ПК

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4,4 м, висота – 3 м.

У зазначеному приміщенні працює 2 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де: *F* – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення.

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін}$.)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють

$\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: *S* – площа приміщення, *S = 13,2 м²;*

h – розрахункова висота підвісу, *h* = 3 *м. (співпадає з в*исотою стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 3 м;*

B – довжина приміщення, *B* = 4,4 *м.*

Підставимо всі значення у формулу та визначемо індекса приміщення: *i=1,1.*

Знаючи індекс приміщення, за знаходимо *n = 0,46 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним*

типом лампам) [8]. Підставимо всі значення у формулу, визначемо світловий потік: *F=14204 Лм.*

Для розрахунку дудемо використовувати **стельові** *світлодіодні панелі Призма-72 6400К*, світловий потік яких $F_л$ *= 7200* Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де: *F* – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення: *N= 14204 / 7200=1,9 шт.*

Приймаємо необхідну кількість *світлодіодних світильників* 2 шт.

Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи підтримки прийняття управлінських рішень.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів підтримки прийняття управлінських рішень.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем підтримки прийняття управлінських рішень.

– Досліджена система підтримки прийняття управлінських рішень.

– На основі отриманих результатів досліджень створена програмна реалізація системи підтримки прийняття управлінських рішень.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання підтримки прийняття управлінських рішень.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MARS.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 17728 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,17 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іванченко О.О. Дослідження та програмна реалізація системи підтримки прийняття управлінських рішень // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Aggarwal C.C. Neural Networks and Deep Learning: A Textbook 1st ed. 2018 Edition. – Springer, 2018. – 520 p
3. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 c.
4. Brink H., Richards J., Fetherolf M. Real-World Machine Learning. – Manning, 2016. – 474 p.
5. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.
6. Fenner M. Machine Learning with Python for Everyone (Addison-Wesley Data & Analytics Series) 1st Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 586 p.
7. Foreman J.W. Data Smart: Using Data Science to Transform Information into Insight 1st Edition. – Wiley, 2013. – 432 p.
8. Hurbans R. Grokking Artificial Intelligence Algorithms. – Manning, 2020. – 631 p.
9. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 p.
10. Kotu V., Deshpande B. Data Science: Concepts and Practice. – Elsevier Science, 2018. – 953 p.
11. Knowledge Base A Complete Guide – 2021 Edition // The Art of Service – Knowledge Base Publishing, 2020. – 306 p.

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

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

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

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

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

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

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

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

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

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

22. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile

Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

37. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation

Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

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

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

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

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

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

43. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

44. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-122.23.0037.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Іванченко О.О.			Дослідження та програмна реалізація системи підтримки прийняття управлінських рішень	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Петренюк В.І.				М	
Н. Контр.		Коваленко А.С.				ЦНТУ КН-22М-2	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи підтримки прийняття управлінських рішень.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи підтримки прийняття управлінських рішень.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи підтримки прийняття управлінських рішень;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

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

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 110 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

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

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Петренюк В.І.

Дослідження та програмна реалізація системи підтримки прийняття управлінських рішень

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 48

Літера: РП

Кропивницький – 2023 року

ФАЙЛ ПРОЕКТУ ПРОГРАМИ SPPR

```
program SPPR;

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Вихідний код до магістерської роботи
на тему: Дослідження та програмна реалізація системи
         підтримки прийняття управлінських рішень
Виконав: студент 2 курсу, групи КН-22М-2 Іванченко Олександр Олександрович, 2023
рік
Керівник: Петренюк В.І.
}

uses
{Використовуваемі бібліотекі, та свої допоміжні модулі}
  Forms,
  SysUtils,

  Unit1 in 'Unit1.pas' {Synaser},
  Unit2 in 'Unit2.pas' {About};
  Unit3 in 'Unit3.pas' {Unit3};
  Unit4 in 'Unit4.pas' {LybADD};
  Unit5 in 'Unit5.pas' {indat};

var
i:integer;
{$R *.res} {ресурси програми}

Begin
{виведення початкового вікна програми}
  try
      CM_SPLASH:=TCM_SPLASH.Create(Application);
      CM_SPLASH.Show;
      CM_SPLASH.Update;
      Application.Initialize;
      Application.Title:= '';

 Application.CreateForm(TSynaser,Synaser);
 Application.CreateForm(TUnit3,Unit3);
 Application.CreateForm(TLybADD,LybADD);
 Application.CreateForm(Tindat,indat);
 Application.CreateForm(TAbout,About);
 finally
     for i:=1 to 100 do
        begin
          CM_SPLASH.glProgress1.Percent:=i;
          CM_SPLASH.Update;
//Робота початкового вікна
          Sleep(20);
        end;

    CM_SPLASH.Hide;
    CM_SPLASH.free;
   end;
      Application.Run;
//Запуск програми
end.
```

ФАЙЛ ЗАВАНТАЖЕННЯ БІБЛІОТЕК LybADD

```
unit LybADD;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Вихідний код до магістерської роботи
на тему: Дослідження та програмна реалізація системи
         підтримки прийняття управлінських рішень
Виконав: студент 2 курсу, групи КН-22М-2 Іванченко Олександр Олександрович, 2023
рік
Керівник: Петренюк В.І.
}

interface

uses
  Libc, dynlibs;
type
HMODULE = Longint;
function LoadLibrary(ModuleName: PChar): HMODULE;
function FreeLibrary(Module: HMODULE): LongBool;
function GetProcAddress(Module: HMODULE; Proc: PChar): Pointer;
function GetModuleFileName(Module: HMODULE; Buffer: PChar; BufLen: Integer):
Integer;
procedure Sleep(milliseconds: Cardinal);

implementation

function LoadLibrary(ModuleName: PChar): HMODULE;
begin
  Result:= HMODULE(dynlibs.LoadLibrary(Modulename));
end;

function FreeLibrary(Module: HMODULE): LongBool;
begin
  Result:=dynlibs.UnloadLibrary(pointer(Module));
end;

function GetProcAddress(Module: HMODULE; Proc: PChar): Pointer;
begin
  Result:=dynlibs.GetProcedureAddress(pointer(Module), Proc);
end;

function GetModuleFileName(Module: HMODULE; Buffer: PChar; BufLen: Integer):
Integer;
begin
  Result:=0;
end;

procedure Sleep(milliseconds: Cardinal);
begin
  usleep(milliseconds * 1000);
end;

end.
```

ФАЙЛ SPLASH ВІКНА

```
unit SPLASH;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Вихідний код до магістерської роботи
на тему: Дослідження та програмна реалізація системи
         підтримки прийняття управлінських рішень
Виконав: студент 2 курсу, групи КН-22М-2 Іванченко Олександр Олександрович, 2023
рік
Керівник: Петренюк В.І.
}


interface

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

type
TCM_SPLASH = class(TForm)
      glProgress1: TglProgress;
      Label1: TLabel;
private
      { Private declarations }
public
      { Public declarations }
end;

var
      CM_SPLASH: TCMSPLASH;

implementation

      {$R *.dfm}

end.
```

ФАЙЛ АВТОРСЬКОГО ПРАВА

```
unit About;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Вихідний код до магістерської роботи
на тему: Дослідження та програмна реалізація системи
         підтримки прийняття управлінських рішень
Виконав: студент 2 курсу, групи КН-22М-2 Іванченко Олександр Олександрович, 2023
рік
Керівник: Петренюк В.І.
}

interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,Buttons,
ExtCtrls;
type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  AboutBox: TAboutBox;
implementation
uses IrqForm;
{$R *.dfm}
procedure TAboutBox.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action:=caNone;
end;
procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
MainForm.show;
AboutBox.hide;
end;
end.
```

ФАЙЛ ПЕРШОЇ ФОРМИ

```
unit main;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Вихідний код до магістерської роботи
на тему: Дослідження та програмна реалізація системи
         підтримки прийняття управлінських рішень
Виконав: студент 2 курсу, групи КН-22М-2 Іванченко Олександр Олександрович, 2023
рік
Керівник: Петренюк В.І.
}

interface

uses

{$IFNDEF FPC}
  Windows, Messages, ADODB, FileCtrl, TeeProcs, TeEngine, Chart,
{$ELSE}
  LMessages, Messages, LCLIntf, sqldb, odbcconn, odbcsqldyn,
{$ENDIF}
  SysUtils, DateUtils, Classes, Graphics, Controls, Forms,
{$IFDEF DELPHI_6UP}
  Variants,
{$ENDIF}
{$IFDEF FASTREPORT4}
  frxClass, frxDBSet, frxDesgn, fcxpComponents, fcxpChartComponents, fcxpChart,
{$ENDIF}
  Dialogs, ComCtrls, DB, StdCtrls, ExtCtrls, GraphUtil, Menus, ImgList, ToolWin,
  fcxTypes, fcxComponent, fcxRes, fcxDataSource, fcxCube, fcxSlice, fcxPainters,
  fcxGridPainters, fcxZone, fcxCustomGrid, fcxSliceGrid, fcxCubeGrid,
  fcxChart, fcxSliceGridToolbar, fcxCubeGridToolBar,
  fcxCustomExport, fcxExportBIFF, fcxExportODF, fcxExportXML, fcxExportHTML,
fcxExportDBF,
  fcxCustomToolbar, fcxControl, fcxStyles;

type
  { TMainForm }
  TMainForm = class(TForm)
    Pages: TPageControl;
    SliceGridSheet: TTabSheet;
    CubeGridSheet: TTabSheet;
    ColorDialog: TColorDialog;
    NotesSheet: TTabSheet;
    AxisImages: TImageList;
    CellImages: TImageList;
    FieldsImages: TImageList;
    ImageList1: TImageList;
    NewInDemo: TMemo;
    FontDialog: TFontDialog;
    fcSliceGridToolbar1: TfcxSliceGridToolbar;
    MDGrid: TfcxSliceGrid;
    CubeGrid: TfcxCubeGrid;
    MDSlice: TfcxSlice;
    MDCube: TfcxCube;
    fcDataSource1: TfcxDataSource;
    fcDBDataSet1: TfcxDBDataSet;
    fcxDirDataset: TfcxUserDataSet;
    fcxDBDSItems: TfcxDBDataSet;
    fcxDBDScustomer: TfcxDBDataSet;
    fcxDBDSemployee: TfcxDBDataSet;
    fcxDBDSparts: TfcxDBDataSet;
```

```
    Demo: TfcxDataSource;
    fcxDBDSorders: TfcxDBDataSet;
    fcxDBvendors: TfcxDBDataSet;
    fcxCalendarDataset: TfcxUserDataSet;
    fcxCalendarDataSource: TfcxDataSource;
    ChartSheet: TTabSheet;
    fcxChart1: TfcxChart;
    MainMenu1: TMainMenu;
    SliceGrid1: TMenuItem;
    Editstyles1: TMenuItem;
    PaintStyleMenu: TMenuItem;
    fcxChartToolBar1: TfcxChartToolBar;
    Splitter2: TSplitter;
    Panel2: TPanel;
    Tree: TTreeView;
    CubeDescr: TMemo;
    Help1: TMenuItem;
    About1: TMenuItem;
    Splitter1: TSplitter;
    fcxCubeGridToolbar1: TfcxCubeGridToolbar;
    fcxBIFFExport1: TfcxBIFFExport;
    fcxODSExport1: TfcxODSExport;
    fcxXMLExport1: TfcxXMLExport;
    fcxHTMLExport1: TfcxHTMLExport;
    fcxDBFExport1: TfcxDBFExport;
    Panel1: TPanel;
    Button1: TButton;
    Button2: TButton;
    Panel3: TPanel;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Panel4: TPanel;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    PrintBtn: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure TreeChange(Sender: TObject; Node: TTreeNode);
    procedure TreeCustomDrawItem(Sender: TCustomTreeView; Node: TTreeNode;
      State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure TreeDeletion(Sender: TObject; Node: TTreeNode);
    procedure ShowSplitFieldsInFieldListChange(Sender: TObject);
    procedure OnUpdateSelection(Sender: TObject);
    procedure fcxDirDatasetOpen(Sender: TObject);
    procedure fcxDirDatasetFirst(Sender: TObject);
    procedure fcxDirDatasetNext(Sender: TObject);
    function fcxDirDatasetEof(Sender: TObject): Boolean;
    procedure fcxDirDatasetClose(Sender: TObject);
    function fcxDirDatasetGetVarData(Sender: TObject; AFieldIndex: Integer;
      out AValue: Variant): Boolean;
    procedure fcxDirDatasetDataFieldAt(Sender: TObject;
      AFieldIndex: Integer; var ADataFieldPropery: TfcxDataFieldPropery);
    function fcxDirDatasetActive(Sender: TObject): Boolean;
    procedure DoGetCellImageIndex(Sender: TfcxSliceDataZone; const Cell:
TfcxMeasureCell; var ImageIndex: Integer);
    procedure DoGetCellStyle(Sender: TfcxSliceDataZone; const Cell:
TfcxMeasureCell; States: TfcxThemeCellStates; Style: TfcxCustomThemeStyle);
    procedure DoGetAxisCellStyle(Sender: TfcxSliceCustomAxisZone; const AInfo:
TfcxCellInfo; State: TfcxThemeState; Style: TfcxCustomThemeStyle);
    procedure MDGridGetDataCellText(Sender: TfcxSliceDataZone;
      const Cell: TfcxMeasureCell; var Text: String);
    procedure employeeCalcFields(DataSet: TDataSet);
    function fcxCalendarDatasetActive(Sender: TObject): Boolean;
```

```
    procedure fcxCalendarDatasetClose(Sender: TObject);
    procedure fcxCalendarDatasetDataFieldAt(Sender: TObject; AFieldIndex:
Integer;
        var ADataFieldPropery: TfcxDataFieldPropery);
    function fcxCalendarDatasetEof(Sender: TObject): Boolean;
    procedure fcxCalendarDatasetFirst(Sender: TObject);
    function fcxCalendarDatasetGetVarData(Sender: TObject; AFieldIndex: Integer;
        out AValue: Variant): Boolean;
    procedure fcxCalendarDatasetNext(Sender: TObject);
    procedure fcxCalendarDatasetOpen(Sender: TObject);
    procedure Editstyles1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure TreeKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure TreeClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure MDGridGetAxisCellImageIndex(Sender: TfcxSliceCustomAxisZone;
      const AInfo: TfcxCellInfo; var ImageIndex: Integer);
    procedure PrintBtnClick(Sender: TObject);
    procedure frxReportGetValue(const VarName: String; var Value: Variant);
  private
    procedure SetPaintStyle(const Value: TfcxPaintStyle);
    function GetPaintStyle: TfcxPaintStyle;
    procedure PaintStyleClick(Sender: TObject);
    procedure SetActiveCube(const Value: TTreeNode);
    procedure OpenExample(Node: TTreeNode; AClearCaption: boolean = True);
    procedure CreateAdoObjects;
    procedure CreateReportObjects;
  private
    FCurrentStyle: Integer;
    FUserPaint: Boolean;
    FDirectory: String;
    FSearchRec: TSearchRec;
    FEof: Boolean;
    FDate: TDate;
    FActiveCube: TTreeNode;
    FSkipAdoExamples: Boolean;
    // report objects
    {$IFDEF FASTREPORT4}
    frxReport: TfrxReport;
    frxDBDataset: TfrxDBDataset;
    fcxpSliceGridProvider: TfcxpSliceGridProvider;
    fcxpChartProvider: TfcxpChartProvider;
    {$ENDIF}
    procedure OpenSampleCube(FileCube: String);
    procedure OpenSampleReport;
    procedure LoadExamplesTree;
    property PaintStyle: TfcxPaintStyle read GetPaintStyle write SetPaintStyle;
  public
    constructor Create(AOwner: TComponent); override;
    property ActiveCube: TTreeNode read FActiveCube write SetActiveCube;
  end;

var
  MainForm: TMainForm;

implementation

{$ifdef fpc}
  {$R *.lfm}
{$else}
```

```
  {$R *.dfm}
{$endif}

uses
  typinfo,
  about,
  fcxUtils,
  fcxStylesEditor;

type
  TTreeNodeKind = (tnkGroup, tnkCube, tnkSchema);

  TTreeNodeValue = record
    FileName: string;
    Description: string;
    StateIndex: Integer;
    NodeKind: TTreeNodeKind;
  end;
  PTreeNodeValue = ^TTreeNodeValue;

{$IFDEF FPC}
  TAdoQuery = TSQLQuery;
{$ENDIF}

{$IFDEF FPC}
const
  INVALID_HANDLE_VALUE = {$ifndef Unix}THandle(-1){$else}Pointer(-1){$endif};
{$ENDIF}

{$IF NOT DEFINED(FPC) AND NOT DEFINED(DELPHI_6UP)}
const
  PathDelim  = '\';
  DriveDelim = ':';
  PathSep    = ';';
{$IFEND}


function PosEx(const SubStr, S: string; Offset: Cardinal = 1): Integer;
var
  I,X: Integer;
  Len, LenSubStr: Integer;
Begin

// Створення необхідних компонентів, якщо вони відсутні на формі.
fcxDBDataSet1: = TfcxDBDataSet.Create (Self);
fcxDataSource1: = TfcxDataSource.Create (Self);
fcxCube1: = TfcxCube.Create (Self);
fcxSlice1: = TfcxSlice.Create (Self);
fcxSliceGrid1: = TfcxSliceGrid.Create (Self);
fcxSliceGrid1.Parent: = Self;
fcxSliceGrid1.Align: = alClient;

// Налаштовуємо зв'язку між компонентами
fcxDBDataSet1.DataSet: = DataSet1;
fcxDataSource1.DataSet: = fcxDBDataSet1;
fcxCube1.DataSource: = fcxDataSource1;
fcxSlice1.Cube: = fcxCube1;
fcxSliceGrid1.Slice: = fcxSlice1;

// Очищаємо опис полів
fcxDataSource1.DeleteFields;

// Вказуємо тип джерела даних куба
fcxCube1.CubeSource: = fccs_DataSource;
```

```
// Завантажуємо дані в куб
fcxCube1.Open;
  if Offset = 1 then
    Result := Pos(SubStr, S)
  else
  begin
    I := Offset;
    LenSubStr := Length(SubStr);
    Len := Length(S) - LenSubStr + 1;
    while I <= Len do
    begin
      if S[I] = SubStr[1] then
      begin
        X := 1;
        while (X < LenSubStr) and (S[I + X] = SubStr[X + 1]) do
          Inc(X);
        if (X = LenSubStr) then
        begin
          Result := I;
          exit;
        end;
      end;
      Inc(I);
    end;
    Result := 0;
  end;
end;
{$ENDIF}

function GetDataPath: String;
var
  ADir: String;
  I: Integer;
  {$ifdef darwin}
  ExeName: String;
  {$endif}
begin
  {$ifdef darwin}
  ADir := ExpandFileName(ParamStr(0));
  // we need a path outside the bundle
  I := LastDelimiter(PathDelim, ADir);
  if I > 0 then
  begin
    ExeName := Copy(ADir, I + 1, Length(ADir));
    ExeName := '/' + ExeName + '.app/Contents/MacOS/' + ExeName;
    I := Pos(ExeName, ADir);
    if I > 0 then
      Delete(ADir, I, Length(ADir));
  end;
  {$else}
  ADir := ExtractFileDir(ParamStr(0));
  {$endif}
  I := LastDelimiter(PathDelim + DriveDelim, ADir);
  Result := Copy(ADir, 1, I) + 'data' + PathDelim;
end;

constructor TMainForm.Create(AOwner: TComponent);
var
  ps: TfcxPaintStyle;
  Item: TMenuItem;
begin
  inherited;

  FCurrentStyle := -1;
```

```
  Pages.ActivePageIndex := 0;
  for ps := Low(TfcxPaintStyle) to High(TfcxPaintStyle) do
  begin
    Item := NewItem(GetEnumName(TypeInfo(TfcxPaintStyle), Ord(ps)), 0, False,
True, PaintStyleClick, 0, '');
    Item.RadioItem := True;
    PaintStyleMenu.Add(Item);
  end;
  PaintStyle := psDefault;

  // test imagelist
  MDGrid.DataZone.Images := CellImages;
  MDCube.Formats.SetDefaults;
end;

procedure TMainForm.LoadExamplesTree;
var
  AFile: TextFile;
  AOneStr, ANameLang, AValueLang : string;
  LastGroup, LastCube, Node: TTreeNode;
  APos1, APos2, APos3, StateIndex: integer;
  NodeValue: PTreeNodeValue;
  ARefField:TfcxReferenceAttributeSFProperties ;
  AAttribute:TfcxSourceField ;
begin
  // Завантажуємо опис полів
  fcxDataSource1.AddFields ;

  // Налаштовуємо створення атрибутів День , Місяць, Рік для поля ' Date1 '
  fcxDataSource1.Fields.FieldByName [' Date1'].SplitProperty.DateSplitPaths:= [
odt_Day , AAttribute:= TfcxSourceField ( fcxDataSource1.Fields.FieldByName ['
IdClient '].SplitPrope;
  // Зробимо атрибут , який зберігає відображуване значення поля ' IdClient':
  // Створюємо новий атрибут для поля ' IdClient ' .
  // Тип атрибуту - Поле з джерела
  AAttribute.SourceFieldType:= fcxsft_Reference ;
  ARefField:= TfcxReferenceAttributeSFProperties (
  AAttribute.SourceFieldProperties ) ;

  // Джерело атрибута збігається з джерелом основного поля
  ARefField.DataSet:= TfcxReferenceSourceFieldProperties (
  fcxDataSource1.Fields.FieldBy;
// Ім'я поля атрибута в джерелі - ' FullName '
  ARefField.DataField.DataFieldName:= ' FullName ';

// Вказуємо ім'я створеного атрибута як джерело відображуваного значення
 fcxDataSource1.Fields.FieldByName [' IdClient '].
              SourceFieldProperties.CaptionSourceAttri

// Та ж задача, але ім'я клієнта береться з іншого джерела даних
// Створюємо новий атрибут для поля ' IdClient'.

AAttribute:=
TfcxSourceField(fcxDataSource1.Fields.FieldByName['IdClient'].SplitProp;

// Тип атрибуту - Поле з джерела
  AAttribute.SourceFieldType:= fcxsft_Reference ;
  ARefField:= TfcxReferenceAttributeSFProperties (
AAttribute.SourceFieldProperties ) ;

// Джерело атрибуту - fcxDataSet2 , посилається на словникову
// таблицю , що зберігає код ' Id '
  ARefField.DataSet:= fcxDataSet2 ;
// Ім'я ключового поля в джерелі атрибуту - ' Id '
  ARefField.DataField.IdField.DataFieldName:= ' Id ';

// Ім'я поля атрибута в джерелі атрибуту - ' FullName '
  ARefField.DataField.DataFieldName:= ' FullName ';
```

```
// Вказуємо ім'я створеного атрибута як джерело відображуваного значення
fcxDataSource1.Fields.FieldByName['IdClient'].SourceFieldProperties.
CaptionSource;
end ;

  NewInDemo.Clear;
  AssignFile(AFile, GetDataPath + fcxResources.Get('dmConf'));
  Reset(AFile);
  try
    LastGroup := nil;
    LastCube := nil;
    Tree.Items.BeginUpdate;
    while not Eof(AFile) do
    begin
      Readln(AFile, AOneStr);
      if Trim(AOneStr) = '' then
        Continue;
      if Copy(AOneStr, 1, 4) = 'GROU' then
      begin
        Node := Tree.Items.AddChild(nil, StringToControl(Trim(copy(AOneStr, 7,
Length(AOneStr)))));
        Node.SelectedIndex := 0;
        Node.ImageIndex := 0;
        New(NodeValue);
        NodeValue^.FileName := '';
        NodeValue^.Description := '';
        NodeValue^.NodeKind := tnkGroup;
        NodeValue^.StateIndex := -1;
        Node.Data := NodeValue;
        LastGroup := Node;
      end
      else
      if copy(AOneStr, 1, 4) = 'CUBE' then
      begin
        APos1 := PosEx(';', AOneStr, 1);
        if APos1 = 0 then continue;
        APos2 := PosEx(';', AOneStr, APos1+1);
        if APos2 = 0 then continue;
        APos3 := PosEx(';', AOneStr, APos2+1);
        if APos3 = 0 then continue;
        StateIndex := StrToInt(trim(copy(AOneStr, 6, APos1 - 6)));
        if FSkipAdoExamples and (StateIndex in [2, 12]) then
          Continue;
        Node := Tree.Items.AddChild(LastGroup,
StringToControl(trim(copy(AOneStr, APos2 + 1, APos3 - APos2 - 1))));
        Node.ImageIndex := 1;
        Node.SelectedIndex := 1;
        New(NodeValue);
      NodeValue^.FileName := trim(copy(AOneStr, APos1 + 1, APos2 - APos1 - 1));
        NodeValue^.Description := trim(copy(AOneStr, APos3 + 1,
Length(AOneStr)));
        NodeValue^.StateIndex := StateIndex;
        NodeValue^.NodeKind := tnkCube;
        Node.Data := NodeValue;
        LastCube := Node;
      end
      else
      if (copy(AOneStr, 1, 4) = 'SCHE') then
      begin
        APos1 := PosEx(';', AOneStr, 1);
        if APos1 = 0 then continue;
        APos2 := PosEx(';', AOneStr, APos1+1);
        if APos2 = 0 then continue;
        APos3 := PosEx(';', AOneStr, APos2+1);
```

```
        if APos3 = 0 then continue;
        Node := Tree.Items.AddChild(LastCube, StringToControl(trim(copy(AOneStr,
APos2 + 1, APos3 - APos2 - 1))));
        Node.ImageIndex := 2;
        Node.SelectedIndex := 2;
        New(NodeValue);
        NodeValue^.FileName := trim(copy(AOneStr, APos1 + 1, APos2 - APos1 -
1));
        NodeValue^.Description := trim(copy(AOneStr, APos3 + 1,
Length(AOneStr)));
        NodeValue^.StateIndex := StrToInt(trim(copy(AOneStr, 8, APos1 - 8)));
        NodeValue^.NodeKind := tnkSchema;
        Node.Data := NodeValue;
      end
      else
      if copy(AOneStr, 1, 4) = 'LANG' then
      begin
        APos1 := PosEx('=', AOneStr, 1);
        if APos1 = 0 then continue;
        ANameLang := trim(copy(AOneStr, 6, APos1 - 6));
        AValueLang := trim(copy(AOneStr, APos1 + 1, Length(AOneStr)));
{
        if ANameLang = 'FIELDSORDER' then
          lbFieldListOrder.Caption := AValueLang
        else if ANameLang = 'USECHARTEVENT' then
          UseChartEvent.Caption := AValueLang;
}
      end
      else
      if copy(AOneStr, 1, 9) = 'NEWINDEMO' then
      begin
        if NotesSheet.Caption = '' then
          NotesSheet.Caption := StringToControl(trim(copy(AOneStr, 11,
Length(AOneStr))))
        else
          NewInDemo.Lines.Add(StringToControl(trim(copy(AOneStr, 11,
Length(AOneStr)))));
      end
      else
        Continue;
    end;
  finally
    Tree.Items.EndUpdate;
    CloseFile(AFile);
  end;
end;

procedure TMainForm.OpenSampleCube(FileCube: String);
begin
  MDCube.LoadfromFile(GetDataPath + 'cubes' + PathDelim + FileCube);
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
  {$IFNDEF MSWindows}
  FSkipAdoExamples := True;
  {$ELSE}
  FSkipAdoExamples := False;
  try
    CreateAdoObjects;
  except
    on E: Exception do
      FSkipAdoExamples := True;
      MessageDlg(E.Message, mtError, [mbOk], 0);
  end;
```

```
  {$ENDIF}
  CreateReportObjects;
  LoadExamplesTree;
  SliceGridSheet.Caption := fcxResources.Get('dmCrossTbl');
  SliceGrid1.Caption := fcxResources.Get('dmCrossTbl');
  ChartSheet.Caption := fcxResources.Get('dmChart');
  CubeGridSheet.Caption := fcxResources.Get('dmSourceTbl');
  MDGrid.OnUpdateSelection := OnUpdateSelection;
end;

procedure TMainForm.FormShow(Sender: TObject);
begin
  {$IFNDEF FPC}
  Tree.Items[0].Item[0].Selected := True;
  {$ELSE}
  Tree.Items[0].Items[0].Selected := True;
  {$ENDIF}
end;

procedure TMainForm.OpenExample(Node: TTreeNode; AClearCaption: boolean = True);
begin
  if AClearCaption then
    MDCube.Caption := '';

  Pages.ActivePageIndex := 0;
  case PTreeNodeValue(Node.Data).NodeKind of
    tnkGroup:
      begin
        // imposible
      end;
    tnkCube:
      begin
        ActiveCube := Node;
      end;
    tnkSchema:
      begin
        ActiveCube := Node.Parent;
        if MDCube.CubeSource <> fccs_None then
        begin
          MDCube.CubeSource := fccs_None;
          OpenSampleCube(PTreeNodeValue(Node.Parent.Data)^.FileName);
        end
        else
          MDCube.ClearGroups;

        MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim +
PTreeNodeValue(Node.Data)^.FileName);
        MDCube.Caption := PTreeNodeValue(Node.Data)^.Description;
      end;
  end;
end;

procedure TMainForm.TreeChange(Sender: TObject; Node: TTreeNode);
begin
  if PTreeNodeValue(Node.Data).StateIndex = -1 then
  begin
    Tree.FullCollapse;
    Node[0].Selected := True;
    Node[0].Expand(True);
  end else
  begin
    CubeDescr.Lines.Text :=
StringToControl(PTreeNodeValue(Node.Data)^.Description);
    Node.Expand(True);
    OpenExample(Node);
```

```
  end;
end;

procedure TMainForm.TreeCustomDrawItem(Sender: TCustomTreeView; Node: TTreeNode;
  State: TCustomDrawState; var DefaultDraw: Boolean);
begin
  if Node.Count <> 0 then
    Tree.Canvas.Font.Style := [fsBold]
  else
    Tree.Canvas.Font.Style := [];
end;

procedure TMainForm.TreeDeletion(Sender: TObject; Node: TTreeNode);
begin
  Dispose(PTreeNodeValue(Node.Data));
end;

procedure TMainForm.PaintStyleClick(Sender: TObject);
begin
  PaintStyle := TfcxPaintStyle(TMenuItem(Sender).MenuIndex);
end;

procedure TMainForm.SetPaintStyle(const Value: TfcxPaintStyle);
begin
  MDGrid.PaintStyle := Value;
  CubeGrid.PaintStyle := Value;
  PaintStyleMenu.Items[Ord(Value)].Checked := True;
end;

function TMainForm.GetPaintStyle: TfcxPaintStyle;
begin
  Result := MDGrid.PaintStyle;
end;

procedure TMainForm.ShowSplitFieldsInFieldListChange(Sender: TObject);
begin
  MDSlice.ShowSplitFieldsInFieldList:=TfcxShowSplitFields(
  ShowSplitFieldsInFieldList.ItemIndex);
end;

procedure TMainForm.OnUpdateSelection(Sender: TObject);
begin
  Selection.Caption := 'Col: ' + IntToStr(MDSlice.SelectedCol) + ' ' +
 /                     'Row: ' + IntToStr(MDSlice.SelectedRow) + ' ' +
                       'Fact: ' + IntToStr(MDSlice.SelectedFact);
end;

procedure TMainForm.DoGetCellImageIndex(Sender: TfcxSliceDataZone;
  const Cell: TfcxMeasureCell; var ImageIndex: Integer);
begin
  if FUserPaint then
    if Cell.IsGrandTotal then
      ImageIndex := 2
    else
    if Cell.IsTotal then
      ImageIndex := 1
    else
      ImageIndex := 0;
end;

procedure TMainForm.DoGetCellStyle(Sender: TfcxSliceDataZone;
  const Cell: TfcxMeasureCell; States: TfcxThemeCellStates;
  Style: TfcxCustomThemeStyle);
begin
  if FUserPaint then
```

```
    if States = [] then
    begin
      Style.GradientDirection := tgdVertical;
      Style.FillColor := RGB(100 + (Cell.MeasureIndex + 1) * 40,
(Cell.MeasureIndex + 1) * 60, (Cell.MeasureIndex + 1) * 80);
      Style.GradientColor := GetHighLightColor(Style.FillColor{$IFDEF
DELPHI_7UP}, 60 {$ENDIF});
      Style.Font.Style := [fsBold];
    end;
end;

procedure TMainForm.DoGetAxisCellStyle(Sender: TfcxSliceCustomAxisZone;
  const AInfo: TfcxCellInfo; State: TfcxThemeState; Style:
TfcxCustomThemeStyle);
var
  AColor: TColor;
begin
  if FUserPaint then
  begin
    AColor := Style.FillColor;
    if AColor <> clNone then
    begin
      Style.GradientDirection := tgdHorizontal;
      Style.FillColor := GetHighLightColor(ColorToRGB(AColor){$IFDEF
DELPHI_7UP}, 12 {$ENDIF});
      Style.GradientColor := GetShadowColor(ColorToRGB(AColor){$IFDEF
DELPHI_7UP}, 12 {$ENDIF});
    end;
  end;
end;

procedure TMainForm.fcxDirDatasetOpen(Sender: TObject);
begin
  if not SelectDirectory('Select directory', '', FDirectory) then
    FDirectory := ExtractFilePath(ParamStr(0));
  FEof := False;
end;

procedure TMainForm.fcxDirDatasetFirst(Sender: TObject);
begin
  FEof := FindFirst(FDirectory + PathDelim + AllFilesMask, faAnyFile,
FSearchRec) <> 0;
end;

procedure TMainForm.fcxDirDatasetNext(Sender: TObject);
begin
  FEof := FindNext(FSearchRec) <> 0;
end;

function TMainForm.fcxDirDatasetEof(Sender: TObject): Boolean;
begin
  Result := FEof;
end;

procedure TMainForm.fcxDirDatasetClose(Sender: TObject);
begin
  FindClose(FSearchRec);
end;

function TMainForm.fcxDirDatasetGetVarData(Sender: TObject;
  AFieldIndex: Integer; out AValue: Variant): Boolean;
begin
  Result := True;
  case AFieldIndex of
    0: AValue := FileDateToDateTime(FSearchRec.Time);
```

```
    1: AValue := FSearchRec.Size;
    2: AValue := FSearchRec.Attr;
    3: AValue := FSearchRec.Name;
  else
    Result := False;
  end;
end;

procedure TMainForm.fcxDirDatasetDataFieldAt(Sender: TObject;
  AFieldIndex: Integer; var ADataFieldPropery: TfcxDataFieldPropery);
begin
  case AFieldIndex of
    0:
      with ADataFieldPropery do
      begin
        FieldName := 'Time';
        DisplayLabel := 'Time';
        Index := 0;
        Size := SizeOf(TDateTime);
        Visible := True;
        FieldType := ftDateTime;
      end;
    1:
      with ADataFieldPropery do
      begin
        FieldName := 'Size';
        DisplayLabel := 'Size';
        Index := 1;
        Size := SizeOf(Integer);
        Visible := True;
        FieldType := ftInteger;
      end;
    2:
      with ADataFieldPropery do
      begin
        FieldName := 'Attr';
        DisplayLabel := 'Attr';
        Index := 2;
        Size := SizeOf(Integer);
        Visible := True;
        FieldType := ftInteger;
      end;
    3:
      with ADataFieldPropery do
      begin
        FieldName := 'Name';
        DisplayLabel := 'FileName';
        Index := 3;
        Size := 0;
        Visible := True;
        FieldType := ftString;
      end;
  end;
end;

function TMainForm.fcxDirDatasetActive(Sender: TObject): Boolean;
begin
  Result := FSearchRec.FindHandle <> INVALID_HANDLE_VALUE;
end;

procedure TMainForm.MDGridGetDataCellText(Sender: TfcxSliceDataZone;
  const Cell: TfcxMeasureCell; var Text: String);
begin
  if FUserPaint then
    if Cell.IsGrandTotal then
```

```
      Text := Format('(%s) %s', [fcxResources.Get('sGrandTotal'), Text])
    else
    if Cell.IsTotal then
      Text := Format('(%s) %s', [fcxResources.Get('sTotal'), Text]);
end;

procedure TMainForm.employeeCalcFields(DataSet: TDataSet);
begin
  DataSet.FieldByName('Name').AsString :=
DataSet.FieldByName('LastName').AsString + ' ' +
DataSet.FieldByName('FirstName').AsString;
end;

function TMainForm.fcxCalendarDatasetActive(Sender: TObject): Boolean;
begin
  Result := True;
end;

procedure TMainForm.fcxCalendarDatasetDataFieldAt(Sender: TObject; AFieldIndex:
    Integer; var ADataFieldPropery: TfcxDataFieldPropery);
begin
  case AFieldIndex of
    0:
      with ADataFieldPropery do
      begin
        FieldName := 'Date';
        DisplayLabel := 'Date';
        Index := 0;
        Size := SizeOf(TDateTime);
        Visible := True;
        FieldType := ftDate;
      end;
  end;
end;

function TMainForm.fcxCalendarDatasetEof(Sender: TObject): Boolean;
begin
  Result := FDate > EndOfTheYear(Now);
end;

procedure TMainForm.fcxCalendarDatasetFirst(Sender: TObject);
begin
  FDate := StartOfTheYear(Now);
end;

function TMainForm.fcxCalendarDatasetGetVarData(Sender: TObject; AFieldIndex:
    Integer; out AValue: Variant): Boolean;
begin
  AValue := FDate;
  Result := True;
end;

procedure TMainForm.fcxCalendarDatasetNext(Sender: TObject);
begin
  FDate := FDate + 1;
end;

procedure TMainForm.fcxCalendarDatasetOpen(Sender: TObject);
begin
  FDate := StartOfTheYear(Now);
end;

procedure TMainForm.Editstyles1Click(Sender: TObject);
begin
  with TfcxStylesEditorDialog.Create(nil) do
```

```
  begin
    Styles := MDGrid.Styles;
    if Execute then
      MDGrid.Styles := TfcxSliceGridStyles(Styles);
  end;
end;

procedure TMainForm.About1Click(Sender: TObject);
begin
  with TAboutForm.Create(Application) do
  begin
    ShowModal;
    Free;
  end;
end;

procedure TMainForm.SetActiveCube(const Value: TTreeNode);
var
  Data: PTreeNodeValue;
  I: Integer;
  D: Cardinal;
begin
  if (FActiveCube <> Value) then
  begin
    FActiveCube := Value;
    Data := PTreeNodeValue(Value.Data);
    FUserPaint := Data^.StateIndex = 5;
    Panel1.Visible := Data^.StateIndex = 16;
    Panel3.Visible := Data^.StateIndex in [12, 22];
    Panel4.Visible := Data^.StateIndex = 23;
    case Data^.StateIndex of
      0, 5, 22, 23: // regular cube
        OpenSampleCube(Data^.FileName);
      1: // speed test
      begin
        D := GetTickCount;
        OpenSampleCube(Data^.FileName);
        D := GetTickCount - D;
        MessageDlg(Format('Cube was loaded with %d msecs', [D]), mtInformation,
[mbOk], 0);
      end;
      2,12: // ADO example
        begin
          Button4.Enabled := True;
          Button5.Enabled := True;
          TADOQuery(fcxDBDSItems.DataSet).SQL.Text :=
'SELECT items.orderno, items.partno, items.qty, orders.custno, orders.empno,
orders.saledate'#$D#$A+'FROM items LEFT OUTER JOIN orders ON items.orderno =
orders.orderno where items.orderno < 1100';
          MDCube.Close;
          MDCube.DataSource := Demo;
          MDCube.CubeSource := fccs_DataSource;
          MDCube.Open;
MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim + Data^.FileName);
          MDCube.Caption := Data^.Description;
          MDSlice.XAxisContainer.InsertMeasuresField(0);
        end;
      3: // FR
        begin
          OpenSampleReport;
          Exit;
        end;
      4: // radio filter
        begin
          OpenSampleCube(Data^.FileName);
```

```
          for I := 0 to MDSlice.FieldsOfRegion[rf_Page].Count - 1 do

TfcxAxisField(MDSlice.FieldsOfRegion[rf_Page].Items[I]).SliceField.UVFilterType
:= uvft_Single;
          MDGrid.PageDimsZone.DropDown(0);
        end;
      6, 16: // user dataset
        begin
          FSearchRec.FindHandle := INVALID_HANDLE_VALUE;
          MDCube.Close;
          fcDataSource1.DataSet := fcxDirDataset;
          MDCube.DataSource := fcDataSource1;
          MDCube.CubeSource := fccs_DataSource;
          MDCube.Open;
          MDCube.Caption := Data^.Description;

MDSlice.YAxisContainer.AddDimension(MDSlice.SliceFields.ItemByName['Name'],
'Name', 'File Name');

MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Size'],
'Size', 'File Size', af_FirstValue);

MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Attr'],
'Attr', 'File Attr', af_FirstValue);

MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Time'],
'Time', 'File Time', af_FirstValue);
          MDSlice.XAxisContainer.InsertMeasuresField(0);
        end;
      7: // user dataset calendar
        begin
          MDCube.Close;
          MDCube.DataSource := fcxCalendarDataSource;
          MDCube.CubeSource := fccs_DataSource;
          MDCube.Open;
          MDCube.Caption := Data^.Description;
          if MDCube.Caption = 'Calendar' then
            MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim +
'calendar_en.mds')
          else
            MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim +
'calendar.mds');
  //
MDSlice.AddSliceFieldToRegion(MDSlice.SliceFields.ItemByName['Date'], 'Date',
'Date', rf_CapYAx);
  //
MDSlice.AddSliceFieldToRegion(MDSlice.SliceFields.ItemByName['Time'], 'Time',
'File Time', rf_CapFacts, af_FirstValue);
  //        MDSlice.MeasuresContainer.Container := MDSlice.XAxisContainer;
        end;
      else
        begin
          OpenSampleCube(Data^.FileName);
        end;
    end;
  end;
end;

procedure TMainForm.TreeKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Key := 0;
end;

procedure TMainForm.CreateAdoObjects;
```

```
var
  Items, Customer, Employee, Parts, Orders, Vendors: TAdoQuery;
  I: Integer;
  employeeName: TStringField;
{$IFNDEF FPC}
  DemoConnection: TAdoConnection;
{$ELSE}
  DemoConnection: TODBCConnection;
{$ENDIF}

  function NewQuery: TAdoQuery;
  begin
    Result := TAdoQuery.Create(Self);
    {$IFDEF FPC}
    Result.DataBase := DemoConnection;
    Result.UsePrimaryKeyAsKey := False;
    {$ELSE}
    Result.Connection := DemoConnection;
    Result.CursorType := ctStatic;
    {$ENDIF}
  end;

  function NewTable(const ATableName: String): TAdoQuery;
  begin
    Result := NewQuery;
    Result.SQL.Text := 'SELECT * FROM ' + ATableName;
  end;

begin
{$IFNDEF FPC}
  DemoConnection := TADOConnection.Create(Self);
  DemoConnection.Provider := 'Microsoft.Jet.OLEDB.4.0;Data Source=' +
GetDataPath + 'demo.mdb;Persist Security Info=False';
  DemoConnection.LoginPrompt := False;
  DemoConnection.Mode := cmShareDenyNone;
{$ELSE}
  InitialiseODBC('libiodbc.dylib');
  DemoConnection := TODBCConnection.Create(Self);
  DemoConnection.Params.Values['DSN'] := 'MS Access Database';
  DemoConnection.Params.Values['DBQ'] := GetDataPath + 'demo.mdb';
  DemoConnection.Transaction := TSQLTransaction.Create(Self);
{$ENDIF}
  DemoConnection.Connected := True;
  Items := NewQuery;
  Items.SQL.Text :=
    'SELECT items.orderno, items.partno, items.qty, orders.custno, orders.empno,
orders.saledate'#$D#$A+
//    'FROM items LEFT OUTER JOIN orders ON items.orderno = orders.orderno';
    'FROM items LEFT OUTER JOIN orders ON items.orderno = orders.orderno where
items.orderno < 1100';
  fcxDBDSItems.DataSet := Items;

  Customer := NewTable('CUSTOMER');
  fcxDBDSCustomer.DataSet := Customer;

  Employee := NewTable('EMPLOYEE');
  {$IFDEF FPC}
  Employee.Prepare;
  {$ENDIF}
  Employee.OnCalcFields := employeeCalcFields;
  Employee.FieldDefs.Update;
  fcxDBDSEmployee.DataSet := Employee;
  for I := 0 to Employee.FieldDefs.Count - 1 do
    Employee.FieldDefs[I].CreateField(Employee);
```

```
  employeeName := TStringField.Create(Self);
  employeeName.FieldName := 'Name';
  employeeName.FieldKind := fkCalculated;
  employeeName.Size := 36;
  employeeName.DataSet := Employee;

  Parts := NewTable('PARTS');
  fcxDBDSParts.DataSet := Parts;

  Orders := NewTable('ORDERS');
  fcxDBDSOrders.DataSet := Orders;

  Vendors := NewTable('VENDORS');
  fcxDBVendors.DataSet := Vendors;
end;

procedure TMainForm.TreeClick(Sender: TObject);
begin
  if PTreeNodeValue(Tree.Selected.Data).StateIndex <> -1 then
  begin
    CubeDescr.Lines.Text :=
StringToControl(PTreeNodeValue(Tree.Selected.Data)^.Description);
    OpenExample(Tree.Selected, False);
    Tree.Selected.Expand(True);
  end;
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 AStr: String;
begin
  AStr := MDCube.Caption;
  FSearchRec.FindHandle := INVALID_HANDLE_VALUE;
  MDCube.Close;
  fcDataSource1.DataSet := fcxDirDataset;
  MDCube.DataSource := fcDataSource1;
  MDCube.CubeSource := fccs_DataSource;
  MDCube.Open;
  MDCube.Caption := AStr;
  MDSlice.YAxisContainer.AddDimension(MDSlice.SliceFields.ItemByName['Name'],
'Name', 'File Name');
  MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Size'],
'Size', 'File Size', af_FirstValue);
  MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Attr'],
'Attr', 'File Attr', af_FirstValue);
  MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Time'],
'Time', 'File Time', af_FirstValue);
  MDSlice.XAxisContainer.InsertMeasuresField(0);
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
  FSearchRec.FindHandle := INVALID_HANDLE_VALUE;
  MDCube.AppendData;
  MDSlice.YAxisContainer.AddDimension(MDSlice.SliceFields.ItemByName['Name'],
'Name', 'File Name');
  MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Size'],
'Size', 'File Size', af_FirstValue);
  MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Attr'],
'Attr', 'File Attr', af_FirstValue);
  MDSlice.MeasuresContainer.AddMeasure(MDSlice.SliceFields.ItemByName['Time'],
'Time', 'File Time', af_FirstValue);
  MDSlice.XAxisContainer.InsertMeasuresField(0);
end;
```

```
procedure TMainForm.Button4Click(Sender: TObject);
var
  Data: PTreeNodeValue;
  AStream: TMemoryStream;
begin
  Data := PTreeNodeValue(FActiveCube.Data);
  if Data^.StateIndex = 22 then
  begin
    Button4.Enabled := False;
    AStream := TMemoryStream.Create;
    MDSlice.SaveToStream(AStream);
    AStream.Position := 0;
    MDCube.AppendFromFile(GetDataPath + 'cubes' + PathDelim +
'2_0_append_2.mdc');
    MDSlice.LoadFromStream(AStream);
    AStream.Free
  end
  else
  if Data^.StateIndex = 23 then
  begin
    Button7.Enabled := False;
    AStream := TMemoryStream.Create;
    MDSlice.SaveToStream(AStream);
    AStream.Position := 0;
    MDSlice.LoadFromStream(AStream);
    AStream.Free
  end
  else
  begin
    Button4.Enabled := False;
    TADOQuery(fcxDBDSItems.DataSet).SQL.Text :=
      'SELECT items.orderno, items.partno, items.qty, orders.custno,
orders.empno, orders.saledate'#$D#$A+
      'FROM items LEFT OUTER JOIN orders ON items.orderno = orders.orderno where
items.orderno between 1100 and 1199';
    MDCube.AppendData;
    MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim + Data^.FileName);
    MDCube.Caption := Data^.Description;
    MDSlice.XAxisContainer.InsertMeasuresField(0);
  end
end;

procedure TMainForm.Button5Click(Sender: TObject);
var
  Data: PTreeNodeValue;
  AStream: TMemoryStream;
begin
  Data := PTreeNodeValue(FActiveCube.Data);
  if Data^.StateIndex = 22 then
  begin
    Button5.Enabled := False;
    AStream := TMemoryStream.Create;
    MDSlice.SaveToStream(AStream);
    AStream.Position := 0;
    MDCube.AppendFromFile(GetDataPath + 'cubes' + PathDelim +
'2_0_append_3.mdc');
    MDSlice.LoadFromStream(AStream);
    AStream.Free
  end
  else
  if Data^.StateIndex = 23 then
  begin
    Button8.Enabled := False;
    AStream := TMemoryStream.Create;
    MDSlice.SaveToStream(AStream);
```

```
    AStream.Position := 0;
    MDSlice.LoadFromStream(AStream);
    AStream.Free
  end
  else
  begin
    Button5.Enabled := False;
    TADOQuery(fcxDBDSItems.DataSet).SQL.Text :=
      'SELECT items.orderno, items.partno, items.qty, orders.custno,
orders.empno, orders.saledate'#$D#$A+
      'FROM items LEFT OUTER JOIN orders ON items.orderno = orders.orderno where
items.orderno >= 1200';
    MDCube.AppendData;
    MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim + Data^.FileName);
    MDCube.Caption := Data^.Description;
    MDSlice.XAxisContainer.InsertMeasuresField(0);
  end;
end;

procedure TMainForm.Button3Click(Sender: TObject);
var
  Data: PTreeNodeValue;
  AStream: TMemoryStream;
begin
  Data := PTreeNodeValue(FActiveCube.Data);
  if Data^.StateIndex = 22 then
  begin
    Button4.Enabled := True;
    Button5.Enabled := True;
    AStream := TMemoryStream.Create;
    MDSlice.SaveToStream(AStream);
    AStream.Position := 0;
    MDCube.LoadFromFile(GetDataPath + 'cubes' + PathDelim + '2_0_append_3.mdc');
    MDSlice.LoadFromStream(AStream);
    AStream.Free
  end
  else
  if Data^.StateIndex = 23 then
  begin
    Button7.Enabled := True;
    Button8.Enabled := True;
    AStream := TMemoryStream.Create;
    MDSlice.SaveToStream(AStream);
    AStream.Position := 0;
    MDSlice.LoadFromStream(AStream);
    AStream.Free
  end
  else
  begin
    Button4.Enabled := True;
    Button5.Enabled := True;
    TADOQuery(fcxDBDSItems.DataSet).SQL.Text :=
      'SELECT items.orderno, items.partno, items.qty, orders.custno,
orders.empno, orders.saledate'#$D#$A+
      'FROM items LEFT OUTER JOIN orders ON items.orderno = orders.orderno where
items.orderno < 1100';
    MDCube.Close;
    MDCube.DataSource := Demo;
    MDCube.CubeSource := fccs_DataSource;
    MDCube.Open;
    MDSlice.LoadfromFile(GetDataPath + 'cubes' + PathDelim + Data^.FileName);
    MDCube.Caption := Data^.Description;
    MDSlice.XAxisContainer.InsertMeasuresField(0);
  end;
end;
```

```
procedure TMainForm.MDGridGetAxisCellImageIndex(
  Sender: TfcxSliceCustomAxisZone; const AInfo: TfcxCellInfo;
  var ImageIndex: Integer);
var
  AxisInfo: TfcxAxisLevelInfo;
  AttrIndex: Integer;
  AttrValue: Variant;
begin
  if Assigned(Tree.Selected) and
     (PTreeNodeValue(Tree.Selected.Data)^.StateIndex in [6, 16]) and
     (AInfo.Data.CellProperties * [pca_GrandTotal, pca_StartTotal] = []) then
  begin
    AxisInfo := Sender.AxisContainer.LevelInfo[AInfo.Data.TreeRect.Level];
    if Assigned(AxisInfo.RegionField) and (AxisInfo.RegionField.Name = 'Name')
then
    begin
      AttrIndex :=
Sender.AxisContainer.Slice.MeasuresContainer.MeasureFields.IndexByName['Attr'];
      if AttrIndex = -1 then
        AttrValue := Null
      else
        AttrValue := Sender.AxisContainer.Slice.GetMeasureValueXY(-1,
AInfo.Data.NodeLevel, 0, AInfo.Data.NodeIndex, AttrIndex, -1, -1);
      if TVarData(AttrValue).VType = varInteger then
      begin
        if (faDirectory and Integer(AttrValue)) <> 0 then
          ImageIndex := 0
        else
          ImageIndex := 1;
      end
      else
        ImageIndex := -1;
    end;
  end;
end;

procedure TMainForm.PrintBtnClick(Sender: TObject);
begin
{$IFDEF FASTREPORT4}
  frxReport.LoadFromFile(GetDataPath + 'cubes' + PathDelim + 'ver2rep.fr3');
  frxReport.ShowReport();
  frxReport.DesignReport();
{$ENDIF}
end;

procedure TMainForm.frxReportGetValue(const VarName: String; var Value:
Variant);
begin
{
  if VarName = 'CUBEDESCRIPTION' then
    value := PTreeNodeValue(Tree.Selected.Data)^.Description
  else
  if VarName = 'SCHEMADESCRIPTION' then
    if ListBox1.itemindex >= 0 then
      value :=
PTreeNodeValue(Tree.Selected.Data)^.Shemas[ListBox1.itemindex].Description
    else
      value := ''
  else
}
  if VarName = 'CUBECAPTION' then
    value := MDCube.Caption
{
  else
```

```
  if VarName = 'SCHEMACAPTION' then
    value := MDSlice.Caption
}
end;

procedure TMainForm.CreateReportObjects;
begin
{$IFDEF FASTREPORT4}
  frxReport := TfrxReport.Create(Self);
  frxReport.OnGetValue := frxReportGetValue;
  frxDBDataset := TfrxDBDataset.Create(Self);
  frxDBDataset.UserName := 'frxDBDataset1';
  fcxpSliceGridProvider := TfcxpSliceGridProvider.Create(Self);
  fcxpSliceGridProvider.SliceGrid := MDGrid;
  fcxpSliceGridProvider.UserName := 'fcxpSliceGridProvider1';
  fcxpSliceGridProvider.PaintSizes.AutoSizeStyle := ssAutoRowHeight;
  fcxpChartProvider := TfcxpChartProvider.Create(Self);
  fcxpChartProvider.Chart := fcxChart1;
  fcxpChartProvider.UserName := 'fcxpChartProvider1';
{$ELSE}
  PrintBtn.Visible := False;
{$ENDIF}
end;

procedure TMainForm.OpenSampleReport;
begin
{$IFDEF FASTREPORT4}
  frxReport.LoadfromFile(GetDataPath + 'fr4.fr3');
  frxReport.DesignReport;
{$ENDIF}
end;

end.
```

БІБЛІОТЕКА CUBE

```
unit Cube;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Вихідний код до магістерської роботи
на тему: Дослідження та програмна реалізація системи
         підтримки прийняття управлінських рішень
Виконав: студент 2 курсу, групи КН-22М-2 Іванченко Олександр Олександрович, 2023
рік
Керівник: Петренюк В.І.
}

interface
{$I fcx.inc}

uses
  SysUtils, Classes, fcxSlice, fcxTypes, fcxComponent
{$IFDEF INCLUDE_FAST_SCRIPT}
  , fs_iinterpreter, fs_ipascal
{$ENDIF}
{$IFDEF Delphi_6UP}
  , Variants
{$ENDIF};
//FMX uses
{$ELSE FMX}

interface
{$I fcx.inc}

uses
  System.SysUtils, System.Classes, System.Variants,
  FMX.fcxSlice, FMX.fcxTypes, FMX.fcxComponent
{$IFDEF INCLUDE_FAST_SCRIPT}
  , FMX.fs_iinterpreter, FMX.fs_ipascal
{$ENDIF}
;
{$ENDIF FMX}

type

  TCommonScriptItems = class(TPersistent)
  protected
    FItems: TStringList;
    function GetCount: Integer; virtual;
    function IndexOf(AString: String): Integer; virtual;
  public
    FTotalOnX, FTotalOnY: Boolean;
    FBaseLevel, FSecondLevel: TfcxSmallCount;
    FXLevel, FYLevel: TfcxSmallCount;
    FIndexInBaseLevel, FIndexInSecondLevel: Integer;
    FIndexInXLevel, FIndexInYLevel: Integer;
    FBaseAdditionalTotalIndex: TfcxSmallCount;
    FSecondAdditionalTotalIndex: TfcxSmallCount;
    FBaseIndex, FSecondIndex: integer;
    FRow : integer;
    FCol : integer;
    FCurrentMeasureIndex: Integer;
    constructor Create;  overload; virtual;
    destructor Destroy; override;
    procedure ClearItems; virtual;
  published
    property Count: Integer read GetCount;
```

```
  end;

  TfcxCustomObject = class(TPersistent)
  private
    function GetValueIsNil: Boolean;
  protected
    FValue: Pointer;
  published
    property ValueIsNil: Boolean read GetValueIsNil;
  end;

function CreateInterpreter(AOwner: TComponent): IfcInterpreter;
function CreateDimensions(ASlice: TfcxSlice): TPersistent;
function CreateMeasures(ASlice: TfcxSlice): TPersistent;
function CreateCustomObject(ASlice: TfcxSlice): TPersistent;
{$IFDEF INCLUDE_FAST_SCRIPT}
procedure fcxAddRTTI(AScript: TfsScript);
{$ENDIF}

implementation


//VCL uses section
{$IFNDEF FMX}
uses
  fcxCodeUtils;
//FMX uses
{$ELSE FMX}
uses
  FMX.fcxCodeUtils;
{$ENDIF FMX}

type
  // script helper classes
  TScriptItems = class;
  TScriptItem = class(TPersistent)
  protected
    FOwner: TScriptItems;
    FSlice: TfcxSlice;
  published
    constructor Create(ASlice: TfcxSlice; AScriptItems: TScriptItems); virtual;
  end;

  TRegionFieldItem = class(TScriptItem)
  protected
    FField: TfcxCommonFieldOfRegion;
    function GetFieldName: String; virtual;
    function GetCaption: String; virtual;
    function GetCurrentCaption: String; virtual;
    function GetCurrentValue: Variant; virtual;
    function GetRegion: TfcxRegionOfField;
    function GetIndexInRegion: Integer;
  published
    constructor Create(ASlice: TfcxSlice; AScriptItems: TScriptItems; AField:
TfcxCommonFieldOfRegion); reintroduce; virtual;
    property FieldName: String read GetFieldName;
    property Caption: String read GetCaption;
    property CurrentValue: Variant read GetCurrentValue;
    property Value: Variant read GetCurrentValue;
    property CurrentCaption: String read GetCurrentCaption;
  end;
  // class Dimension
  TDimension = class(TRegionFieldItem)
  private
    FLevelInfo: TfcxAxisLevelInfo;
```

```
    FSubGroup: TDimension;
    function GetSubGroup: TDimension;
  protected
    function GetCurrentCaption: String; override;
    function GetCurrentValue: Variant; override;
  public
  published
    constructor Create(ASlice: TfcxSlice; AScriptItems: TScriptItems;
ALevelInfo: TfcxAxisLevelInfo); reintroduce; virtual;
    destructor Destroy; override;
    property SubGroup: TDimension read GetSubGroup;
  end;
  TfcxVarArr = array of Variant;
  // class Measure
  TMeasure = class(TRegionFieldItem)
  private
    function GetColOffsetValueWithDimValue(ADimValue: Variant): Variant;
    function GetRowOffsetValueWithDimValue(ADimValue: Variant): Variant;
    function GetTotalValueForDims(ADimNames: String): Variant;
    function GetColRowOffsetValue(ColOffset, RowOffset: integer): Variant;
    function GetColRowOffsetWLevelValue(ColOffset, RowOffset,
      ColLevelOffset, RowLevelOffset: integer): Variant;
    function GetValueWithColDimValues(ADimValues: TfcxVarArr): Variant;
    function GetValueWithRowDimValues(ADimValues: TfcxVarArr): Variant;
    function GetColOffsetTotalValueForDims(Offset: integer;
      ADimNames: String): Variant;
    function GetRowOffsetTotalValueForDims(Offset: integer;
      ADimNames: String): Variant;
  protected
    function GetColOffsetValue(Offset: integer): Variant;
    function GetRowOffsetValue(Offset: integer): Variant;
    function GetCurrentCaption: String; override;
    function GetCurrentValue: Variant; override;
  public
    property ColOffsetValue[Offset: integer]: Variant read GetColOffsetValue;
    property RowOffsetValue[Offset: integer]: Variant read GetRowOffsetValue;
    property ColRowOffsetValue[ColOffset, RowOffset: integer]: Variant read
GetColRowOffsetValue;
    property ColRowOffsetValueWLevel[ColOffset, RowOffset, ColLevelOffset,
RowLevelOffset: integer]: Variant read GetColRowOffsetWLevelValue;
    property ColOffsetValueWithDimValue[ADimValue: Variant]: Variant read
GetColOffsetValueWithDimValue;
    property RowOffsetValueWithDimValue[ADimValue: Variant]: Variant read
GetRowOffsetValueWithDimValue;
    property TotalValueForDims[ADimNames: String]: Variant read
GetTotalValueForDims;
    property ColOffsetTotalValueForDims[Offset: integer; ADimNames: String]:
Variant read GetColOffsetTotalValueForDims;
    property RowOffsetTotalValueForDims[Offset: integer; ADimNames: String]:
Variant read GetRowOffsetTotalValueForDims;
  published
  end;

  TScriptItems = class(TCommonScriptItems)
  protected
    FSlice: TfcxSlice;
  public
    constructor Create(ASlice: TfcxSlice); overload; virtual;
  published
    property Row : integer read FRow;
    property Col : integer read FCol;
  end;

  TMeasures = class(TScriptItems)
  private
```

```
    function GetDetailValue(ARecordIndex: Integer; AFieldName: String): Variant;
    function GetRecordCount: integer;
    function GetCurrentMeasureName: String;
  protected
    function GetCount: Integer; override;
    function GetItem(AIndex: Integer): TMeasure;
    function GetItemByCaption(AIndex: String): TMeasure;
    function GetItemByName(AIndex: String): TMeasure;
  public
    property Items[AIndex: Integer]: TMeasure read GetItem;
    property ItemByCaption[AIndex: String]: TMeasure read GetItemByCaption;
    property ItemByName[AIndex: String]: TMeasure read GetItemByName;
    property DetailValue[ARecordIndex: Integer; AFieldName: String]: Variant
read GetDetailValue;
    property RecordCount: integer read GetRecordCount;
  published
    property XLevel: TfcxSmallCount read FXLevel;
    property YLevel: TfcxSmallCount read FYLevel;
    property CurrentMeasureIndex: Integer read FCurrentMeasureIndex;
    property CurrentMeasureName: String read GetCurrentMeasureName;
  end;

  TDimensions = class(TScriptItems)
  private
    function GetXAxisItem(AIndex: Integer): TDimension;
    function GetXAxisLevelsCount: TfcxSmallCount;
    function GetYAxisItem(AIndex: Integer): TDimension;
    function GetYAxisLevelsCount: TfcxSmallCount;
  protected
    function GetCount: Integer; override;
    function GetItem(AIndex: Integer): TDimension;
    function GetItemByCaption(AIndex: String): TDimension;
    function GetItemByName(AIndex: String): TDimension;
  public
    property Items[AIndex: Integer]: TDimension read GetItem;
    property ItemByCaption[AIndex: String]: TDimension read GetItemByCaption;
    property ItemByName[AIndex: String]: TDimension read GetItemByName;
    property XAxisItems[AIndex: Integer]: TDimension read GetXAxisItem;
    property YAxisItems[AIndex: Integer]: TDimension read GetYAxisItem;
  published
    property IsTotalByCol: boolean read FTotalOnX;
    property IsTotalByRow: boolean read FTotalOnY;
    property XLevel: TfcxSmallCount read FXLevel;
    property YLevel: TfcxSmallCount read FYLevel;
    property XAxisLevelsCount: TfcxSmallCount read GetXAxisLevelsCount;
    property YAxisLevelsCount: TfcxSmallCount read GetYAxisLevelsCount;
  end;

  TfCubeStub = class(TInterfacedObject, IfcInterpreter)
  private
    FScript: TStringList;
    FScriptLanguage: String;
    FCompiled: Boolean;
  public
    constructor Create(AOwner: TComponent);
    destructor Destroy; override;
    procedure AddVariable(const Name, Typ: String; const Value: Variant);
    function CallFunction(const Func: Pointer; const Params: Variant): Variant;
    function Compile: Boolean;
    function GetErrorMsg: String;
    function GetCompiled: Boolean;
    function GetFunctionPointer(AFunctionName: String): Pointer;
    function GetObject: TObject;
    function GetScript: TStrings;
    procedure SetScript(const AScript: TStrings);
```

```
    function GetScriptLanguage: String;
    procedure SetScriptLanguage(const Value: String);
    procedure Clear;
    procedure EnumFunctions(const List: TStrings);
  end;

{$IFDEF INCLUDE_FAST_SCRIPT}
  TfcxFunctionsFastCube2 = class(TfsRTTIModule)
  private
    function CallMethod(Instance: TObject; ClassType: TClass;
      const MethodName: String; Caller: TfsMethodHelper): Variant;
    function GetIndexProp(Instance: TObject; ClassType: TClass;
      const MethodName: String; var Params: Variant): Variant;
    function GetProp(Instance: TObject; ClassType: TClass;
      const PropName: String): Variant;
    procedure SetProp(Instance: TObject; ClassType: TClass;
      const PropName: String; Value: Variant);
  public
    procedure AddClasses(AScript: TfsScript);
    constructor Create(AScript: TfsScript); override;
  end;

  procedure fcxAddRTTI(AScript: TfsScript);
  var
    rtti: TfsRTTIModule;
  begin
    AScript.AddRTTI;
    AScript.AddedBy := TObject(1); // do not clear
    rtti := TfsRTTIModule(TfcxFunctionsFastCube2.NewInstance);
    rtti.Create(AScript);
    TfcxFunctionsFastCube2(rtti).AddClasses(AScript);
    AScript.Add('', rtti);
    AScript.AddedBy := nil;
  end;

type
  TfcxFSInterpreter = class(TInterfacedObject, IfcInterpreter)
  private
    FOwner: TComponent;
    FCommonInterpreter: TfsScript;
    FInterpreter: TfsScript;
    FCompiled: Boolean;
    FCommonScriptChangedTick: Cardinal;
    FScript: TfcxScriptStringList;
    function GetScriptChanged: boolean;
  public
    constructor Create(AOwner: TComponent);
    destructor Destroy; override;
    procedure AddVariable(const Name, Typ: String; const Value: Variant);
    function CallFunction(const Func: Pointer; const Params: Variant): Variant;
    function Compile: Boolean;
    function GetErrorMsg: String;
    function GetCompiled: Boolean;
    function GetFunctionPointer(AFunctionName: String): Pointer;
    function GetObject: TObject;
    function GetScript: TStrings;
    procedure SetScript(const AScript: TStrings);
    function GetScriptLanguage: String;
    procedure SetScriptLanguage(const Value: String);
    procedure Clear;
    procedure EnumFunctions(const List: TStrings);
    property ScriptChanged: boolean read GetScriptChanged;
  end;

{ TfcxFunctionsFastCube2 }
```

```
procedure TfcxFunctionsFastCube2.AddClasses(AScript: TfsScript);
begin
  with AScript do
  begin
    with AddClass(TDimension, 'TPersistent') do
    begin
    end;
    AddEnum('TfcxRegionOfField', 'rf_Page, rf_CapXAx, rf_CapYAx, rf_CapFacts,
rf_None');
    with AddClass(TDimensions, 'TPersistent') do
    begin
AddDefaultProperty('ItemByName', 'String', 'TDimension', GetIndexProp, True);
AddIndexProperty('ItemByCaption', 'String', 'TDimension', GetIndexProp, True);
AddIndexProperty('Items', 'Integer', 'TDimension', GetIndexProp, True);
AddIndexProperty('XAxisItems', 'Integer', 'TDimension', GetIndexProp, True);
AddIndexProperty('YAxisItems', 'Integer', 'TDimension', GetIndexProp, True);
    end;

    with AddClass(TMeasure, 'TPersistent') do
    begin
AddIndexProperty('ColOffsetValue', 'Integer', 'Variant', GetIndexProp, True);
AddIndexProperty('RowOffsetValue', 'Integer', 'Variant', GetIndexProp, True);
AddIndexProperty('ColOffsetValueWithDimValue', 'Variant', 'Variant',
GetIndexProp, True);
AddIndexProperty('RowOffsetValueWithDimValue', 'Variant', 'Variant',
GetIndexProp, True);
AddIndexProperty('TotalValueForDims', 'String', 'Variant', GetIndexProp, True);
AddIndexProperty('ColOffsetTotalValueForDims', 'Integer, String', 'Variant',
GetIndexProp, True);
AddIndexProperty('RowOffsetTotalValueForDims', 'Integer, String', 'Variant',
GetIndexProp, True);
AddIndexProperty('ColRowOffsetValue', 'Integer, Integer', 'Variant',
GetIndexProp, True);
AddIndexProperty('ColRowOffsetValueWLevel', 'Integer, Integer, Integer,
Integer', 'Variant', GetIndexProp, True);
AddMethod('function GetValueWithRowDimValues(ADimValues: Variant): Variant',
CallMethod);
AddMethod('function GetValueWithColDimValues(ADimValues: Variant): Variant',
CallMethod);
    end;

with AddClass(TMeasures, 'TPersistent') do
    begin
      AddDefaultProperty('ItemByName', 'String', 'TMeasure', GetIndexProp,
True);
      AddIndexProperty('ItemByCaption', 'String', 'TMeasure', GetIndexProp,
True);
      AddIndexProperty('Items', 'Integer', 'TMeasure', GetIndexProp, True);
      AddIndexProperty('DetailValue', 'Integer, String', 'Variant',
GetIndexProp, True);
      AddProperty('RecordCount', 'Integer', GetProp, nil);
// for compatibility
      AddMethod('procedure PrepareDetailInfo', CallMethod);
    end;

    with AddClass(TfcxSliceField, 'TPersistent') do
    begin
      AddProperty('FilterCount', 'Integer', GetProp, nil);
      AddProperty('IsFiltered', 'Boolean', GetProp, nil);
    end;

    with AddClass(TfcxSliceFields, 'TPersistent') do
    begin
```

```
      AddDefaultProperty('ItemByName', 'String', 'TfcxSliceField', GetIndexProp,
True);
      AddIndexProperty('ItemByCaption', 'String', 'TfcxSliceField',
GetIndexProp, True);
      AddIndexProperty('Items', 'Integer', 'TfcxSliceField', GetIndexProp,
True);
    end;

    with AddClass(TfcxCustomObject, 'TPersistent') do
    begin
      AddProperty('Value', 'Pointer', GetProp, SetProp);
    end;
  end;
end;

function TfcxFunctionsFastCube2.CallMethod(Instance: TObject; ClassType: TClass;
  const MethodName: String; Caller: TfsMethodHelper): Variant;
begin
  Result := 0;
  if ClassType = TMeasures then
  begin
    if MethodName = 'PREPAREDETAILINFO' then
// for compatibility
      ;
  end
  else
  if ClassType = TMeasure then
  begin
    if MethodName = 'GETVALUEWITHROWDIMVALUES' then
      Result := TMeasure(Instance).GetValueWithRowDimValues(Caller.Params[0])
    else
    if MethodName = 'GETVALUEWITHCOLDIMVALUES' then
      Result := TMeasure(Instance).GetValueWithColDimValues(Caller.Params[0])
      ;
  end;
end;

constructor TfcxFunctionsFastCube2.Create(AScript: TfsScript);
begin
  inherited Create(AScript);
end;

{ for future
function TfcxFunctionsFastCube2.CallMethod(Instance: TObject; ClassType: TClass;
  const MethodName: String; Caller: TfsMethodHelper): Variant;
begin
  // no need in mean time - for future use only
  Result := 0;
  if ClassType = TDimension then
  begin
  end
  else if ClassType = TMeasure then
  begin
  end;
end;
}

function TfcxFunctionsFastCube2.GetIndexProp(Instance: TObject; ClassType:
TClass;
  const MethodName: String; var Params: Variant): Variant;
begin
  Result := 0;

  if ClassType = TDimensions then
  begin
```

```
    if MethodName = 'ITEMBYNAME.GET' then
      Result := PtrUInt(TDimensions(Instance).ItemByName[Params[0]]) else
    if MethodName = 'ITEMBYCAPTION.GET' then
      Result := PtrUInt(TDimensions(Instance).ItemByCaption[Params[0]]) else
    if MethodName = 'ITEMS.GET' then
      Result := PtrUInt(TDimensions(Instance).Items[Params[0]]) else
    if MethodName = 'XAXISITEMS.GET' then
      Result := PtrUInt(TDimensions(Instance).XAxisItems[Params[0]]) else
    if MethodName = 'YAXISITEMS.GET' then
      Result := PtrUInt(TDimensions(Instance).YAxisItems[Params[0]])
  end else
  if ClassType = TMeasures then
  begin
    if MethodName = 'ITEMBYNAME.GET' then
      Result := PtrUInt(TMeasures(Instance).ItemByName[Params[0]]) else
    if MethodName = 'ITEMBYCAPTION.GET' then
      Result := PtrUInt(TMeasures(Instance).ItemByCaption[Params[0]]) else
    if MethodName = 'ITEMS.GET' then
      Result := PtrUInt(TMeasures(Instance).Items[Params[0]]) else
    if MethodName = 'DETAILVALUE.GET' then
      Result := TMeasures(Instance).DetailValue[Params[0], Params[1]]
  end
  else
  if ClassType = TMeasure then
  begin
    if MethodName = 'COLOFFSETVALUE.GET' then
      Result := TMeasure(Instance).ColOffsetValue[Params[0]] else
    if MethodName = 'ROWOFFSETVALUE.GET' then
      Result := TMeasure(Instance).RowOffsetValue[Params[0]] else
    if MethodName = 'COLOFFSETVALUEWITHDIMVALUE.GET' then
      Result := TMeasure(Instance).ColOffsetValueWithDimValue[Params[0]] else
    if MethodName = 'ROWOFFSETVALUEWITHDIMVALUE.GET' then
      Result := TMeasure(Instance).RowOffsetValueWithDimValue[Params[0]] else
    if MethodName = 'TOTALVALUEFORDIMS.GET' then
      Result := TMeasure(Instance).TotalValueForDims[Params[0]] else
    if MethodName = 'COLOFFSETTOTALVALUEFORDIMS.GET' then
      Result := TMeasure(Instance).ColOffsetTotalValueForDims[Params[0],
Params[1]] else
    if MethodName = 'ROWOFFSETTOTALVALUEFORDIMS.GET' then
      Result := TMeasure(Instance).RowOffsetTotalValueForDims[Params[0],
Params[1]] else
    if MethodName = 'COLROWOFFSETVALUE.GET' then
      Result := TMeasure(Instance).ColRowOffsetValue[Params[0], Params[1]] else
    if MethodName = 'COLROWOFFSETVALUEWLEVEL.GET' then
      Result := TMeasure(Instance).ColRowOffsetValueWLevel[Params[0], Params[1],
Params[2], Params[3]]
  end
  else
  if ClassType = TfcxSliceFields then
  begin
    if MethodName = 'ITEMBYNAME.GET' then
      Result := PtrUInt(TfcxSliceFields(Instance).ItemByName[Params[0]]) else
    if MethodName = 'ITEMBYCAPTION.GET' then
      Result := PtrUInt(TfcxSliceFields(Instance).ItemByCaption[Params[0]]) else
    if MethodName = 'ITEMS.GET' then
      Result := PtrUInt(TfcxSliceFields(Instance).Items[Params[0]])
  end
end;

function TfcxFunctionsFastCube2.GetProp(Instance: TObject; ClassType: TClass;
  const PropName: String): Variant;
begin
  Result := 0;

  if ClassType = TMeasures then
```

```
  begin
    if PropName = 'RECORDCOUNT' then
      Result := TMeasures(Instance).RecordCount
  end
  else
  if ClassType = TfcxCustomObject then
  begin
    if PropName = 'VALUE' then
      Result := PtrUInt(TfcxCustomObject(Instance).FValue);
  end
  else
  if ClassType = TfcxSliceField then
  begin
    if PropName = 'FILTERCOUNT' then
      Result := TfcxSliceField(Instance).UVFilteredValuesCount
    else if PropName = 'ISFILTERED' then
      Result := TfcxSliceField(Instance).UVFilteredValuesCount <> 0
  end
end;

procedure TfcxFunctionsFastCube2.SetProp(Instance: TObject;
  ClassType: TClass; const PropName: String; Value: Variant);
begin
  if ClassType = TfcxCustomObject then
  begin
    if PropName = 'VALUE' then
      TfcxCustomObject(Instance).FValue := Pointer(PtrUInt(Value));
  end
end;

{ TfcxFSInterpreter }

constructor TfcxFSInterpreter.Create(AOwner: TComponent);
begin
  inherited Create;
  FOwner := AOwner;
  FCompiled := False;
  FCommonInterpreter := TfsScript.Create(nil);
  FCommonInterpreter.Parent := fsGlobalUnit;
  fcxAddRTTI(FCommonInterpreter);
  FScript := TfcxScriptStringList.Create;
  FInterpreter := TfsScript.Create(nil);
  FInterpreter.SyntaxType := 'PascalScript';
  fcxAddRTTI(FInterpreter);
  FCommonScriptChangedTick := 0;
  fcxEmptyCode(FScript, GetScriptLanguage);
end;

procedure TfcxFSInterpreter.AddVariable(const Name, Typ: String; const Value:
Variant);
begin
  FInterpreter.AddVariable(Name, Typ, Value);
end;

function TfcxFSInterpreter.CallFunction(const Func: Pointer; const Params:
Variant): Variant;
begin
  if Assigned(Func) then
  begin
    FInterpreter.CallFunction2(Func, Params);
    Result := TfsProcVariable(Func).Params[0].Value;
  end
  else
    Result := UnAssigned;
end;
```

```
function TfcxFSInterpreter.Compile: Boolean;
begin
  Result := False;
  if FOwner is TfcxComponent then
  begin
    if FCompiled and not FScript.IsChanged and (FCommonScriptChangedTick =
TfcxComponent(FOwner).GetCommonScriptChangedTick) then
// Не треба повторно компілювати
      Result := True
    else
    begin
      FInterpreter.Clear;
      FInterpreter.Lines.Assign(FScript);
// Підтягти загальний скрипт
      if TfcxComponent(FOwner).CommonScript <> nil then
      begin
        if TfcxComponent(FOwner).CommonScriptLanguage = '' then
          FCommonInterpreter.SyntaxType := 'PascalScript'
        else
          FCommonInterpreter.SyntaxType :=
TfcxComponent(FOwner).CommonScriptLanguage;
        FCommonInterpreter.Lines.Assign(TfcxComponent(FOwner).CommonScript);
        FCommonScriptChangedTick :=
TfcxComponent(FOwner).GetCommonScriptChangedTick;
        FInterpreter.Parent := FCommonInterpreter;
        Result := FCommonInterpreter.Compile;
      end
      else
      begin
        FInterpreter.Parent := fsGlobalUnit;
        FCommonScriptChangedTick := 0;
        Result := True;
      end;
// Виклик події для підготовчих операцій
      if Result then
      begin
        TfcxComponent(FOwner).InitInterpreter;
        Result := FInterpreter.Compile;
      end;
      if Result then
      begin
        FCompiled := True;
        FScript.IsChanged := False;
      end
      else
      begin
// Потрібно записати інформацію про помилку
      end
    end;
  end
end;

function TfcxFSInterpreter.GetErrorMsg: String;
begin
  Result := FInterpreter.ErrorMsg;
end;

function TfcxFSInterpreter.GetFunctionPointer(AFunctionName: String): Pointer;
begin
  Result := TfsProcVariable(FInterpreter.FindLocal(AFunctionName));
end;

function TfcxFSInterpreter.GetObject: TObject;
begin
```

```
  Result := FInterpreter;
end;

function TfcxFSInterpreter.GetScript: TStrings;
begin
  Result := FScript;
end;

procedure TfcxFSInterpreter.SetScript(const AScript: TStrings);
begin
  FScript.Assign(AScript);
end;

procedure TfcxFSInterpreter.Clear;
begin
  FScript.Clear;
  FCompiled := False;
end;

destructor TfcxFSInterpreter.Destroy;
begin
  FInterpreter.Free;
  FCommonInterpreter.Free;
  FScript.Free;
  inherited;
end;

function TfcxFSInterpreter.GetCompiled: Boolean;
begin
  Result := FCompiled;
end;

function TfcxFSInterpreter.GetScriptChanged: boolean;
begin
  Result := FScript.IsChanged;
end;

procedure TfcxFSInterpreter.EnumFunctions(const List: TStrings);
var
  i: Integer;
  ProcVar: TfsProcVariable;
begin
  if Compile then
  begin
    for i := 0 to FInterpreter.Count - 1 do
      if FInterpreter.Items[i] is TfsProcVariable then
      begin
        ProcVar := TfsProcVariable(FInterpreter.Items[i]);
if (ProcVar.Count = 0) and ProcVar.IsFunc and (ProcVar.Typ = fvtVariant) then
          List.Add(ProcVar.Name);
      end;
  end;
end;

function TfcxFSInterpreter.GetScriptLanguage: String;
begin
  Result := FInterpreter.SyntaxType;
end;

procedure TfcxFSInterpreter.SetScriptLanguage(const Value: String);
begin
  if Value = '' then
    FInterpreter.SyntaxType := 'PascalScript'
  else
    FInterpreter.SyntaxType := Value;
```

```
end;

function CreateInterpreter(AOwner: TComponent): IfcInterpreter;
begin
{$IFDEF INCLUDE_FAST_SCRIPT}
  Result := TfcxFSInterpreter.Create(AOwner);
{$ELSE}
  Result := TfCubeStub.Create(AOwner);
{$ENDIF}
end;

{ TfCubeStub }

procedure TfCubeStub.AddVariable(const Name, Typ: String;
  const Value: Variant);
begin
end;

function TfCubeStub.CallFunction(const Func: Pointer; const Params: Variant):
Variant;
begin
  Result := Unassigned;
end;

procedure TfCubeStub.Clear;
begin
  FScript.Clear;
end;

function TfCubeStub.Compile: Boolean;
begin
  Result := False;
end;

constructor TfCubeStub.Create(AOwner: TComponent);
begin
  FCompiled := True;
  FScript := TStringList.Create;
end;

destructor TfCubeStub.Destroy;
begin
  FScript.Free;
  inherited;
end;

procedure TfCubeStub.EnumFunctions(const List: TStrings);
begin
end;

function TfCubeStub.GetCompiled: Boolean;
begin
  Result := FCompiled;
end;

function TfCubeStub.GetErrorMsg: String;
begin
  Result := '';
end;

function TfCubeStub.GetFunctionPointer(AFunctionName: String): Pointer;
begin
  Result := nil;
end;
```

```
function TfCubeStub.GetObject: TObject;
begin
  Result := nil;
end;

function TfCubeStub.GetScript: TStrings;
begin
  Result := FScript;
end;

function TfCubeStub.GetScriptLanguage: String;
begin
  Result := FScriptLanguage;
end;

procedure TfCubeStub.SetScript(const AScript: TStrings);
begin
  FScript.Assign(AScript);
end;

procedure TfCubeStub.SetScriptLanguage(const Value: String);
begin
  FScriptLanguage := Value;
end;

{ TScriptItem }

constructor TScriptItem.Create(ASlice: TfcxSlice; AScriptItems: TScriptItems);
begin
  FSlice := ASlice;
  FOwner := AScriptItems;
end;

{ TRegionFieldItem }

constructor TRegionFieldItem.Create(ASlice: TfcxSlice; AScriptItems:
TScriptItems; AField: TfcxCommonFieldOfRegion);
begin
  inherited Create(ASlice, AScriptItems);
  FField := AField;
end;

function TRegionFieldItem.GetFieldName: String;
begin
  if (FField <> nil) then
    Result := FField.Name;
end;

function TRegionFieldItem.GetCaption: String;
begin
  if (FField <> nil) then Result := FField.Caption;
end;

function TRegionFieldItem.GetCurrentCaption: String;
begin
  Result := '';
end;

function TRegionFieldItem.GetCurrentValue: Variant;
begin
  Result := UnAssigned;
end;

function TRegionFieldItem.GetIndexInRegion: Integer;
begin
```

```
  Result := FField.Index;
end;
function TRegionFieldItem.GetRegion: TfcxRegionOfField;
begin
  Result := FField.Owner.Container.Region;
end;

{ TMeasure }

function TMeasure.GetColOffsetTotalValueForDims(Offset: integer;
  ADimNames: String): Variant;
begin
  if FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureTotalValueForDimsBSOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
ADimNames,
      FOwner.FBaseLevel, FOwner.FSecondLevel, Offset, 0, True)
  else
    Result := FSlice.GetMeasureTotalValueForDimsBSOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
ADimNames,
      FOwner.FBaseLevel, FOwner.FSecondLevel, 0, Offset, True)
end;

function TMeasure.GetColOffsetValue(Offset: integer): Variant;
begin
  if FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, Offset, 0)
  else
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, 0, Offset)
end;

function TMeasure.GetColOffsetValueWithDimValue(ADimValue: Variant): Variant;
begin
  if FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSWOffsetOnValue(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, ADimValue, 0)
  else
    Result := FSlice.GetMeasureValueBSWOffsetOnValue(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, 0, ADimValue)
end;

function TMeasure.GetColRowOffsetValue(ColOffset,
  RowOffset: integer): Variant;
begin
  if FSlice.MeasuresContainer.BaseAxisIsX then
```

```
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, ColOffset, RowOffset)
  else
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, RowOffset, ColOffset)
end;

function TMeasure.GetColRowOffsetWLevelValue(ColOffset, RowOffset,
  ColLevelOffset, RowLevelOffset: integer): Variant;
begin
  if FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      ColLevelOffset, RowLevelOffset, ColOffset, RowOffset)
  else
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      RowLevelOffset, ColLevelOffset, RowOffset, ColOffset)
end;

function TMeasure.GetCurrentCaption: String;
begin
  Result := FSlice.GetMeasureValueCaptionBS(FOwner.FBaseLevel,
FOwner.FSecondLevel,
    FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
    FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex)
end;

function TMeasure.GetCurrentValue: Variant;
begin
  Result := FSlice.GetMeasureValueBS(FOwner.FBaseLevel, FOwner.FSecondLevel,
    FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
    FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex)
end;

function TMeasure.GetRowOffsetTotalValueForDims(Offset: integer;
  ADimNames: String): Variant;
begin
  if not FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureTotalValueForDimsBSOffset(FOwner.FBaseLevel,
      FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
ADimNames,
      FOwner.FBaseLevel, FOwner.FSecondLevel, Offset, 0, True)
  else
    Result := FSlice.GetMeasureTotalValueForDimsBSOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
ADimNames,
      FOwner.FBaseLevel, FOwner.FSecondLevel, 0, Offset, True)
end;

function TMeasure.GetRowOffsetValue(Offset: integer): Variant;
```

```
begin
  if not FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, Offset, 0)
  else
    Result := FSlice.GetMeasureValueBSWOffset(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, 0, Offset)
end;

function TMeasure.GetRowOffsetValueWithDimValue(ADimValue: Variant): Variant;
begin
  if not FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSWOffsetOnValue(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, ADimValue, 0)
  else
Result := FSlice.GetMeasureValueBSWOffsetOnValue(FOwner.FBaseLevel,
      FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      FOwner.FBaseLevel, FOwner.FSecondLevel, 0, ADimValue)
end;

function TMeasure.GetTotalValueForDims(ADimNames: String): Variant;
begin
  Result := FSlice.GetMeasureTotalValueForDimsBS(FOwner.FBaseLevel,
FOwner.FSecondLevel,
    FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
    FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
ADimNames)
end;

function TMeasure.GetValueWithColDimValues(
  ADimValues: TfcxVarArr): Variant;
begin
  if FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSOnWayValues(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      True, False, ADimValues, [])
  else
    Result := FSlice.GetMeasureValueBSOnWayValues(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      False, True, [], ADimValues)
end;

function TMeasure.GetValueWithRowDimValues(
  ADimValues: TfcxVarArr): Variant;
begin
  if not FSlice.MeasuresContainer.BaseAxisIsX then
    Result := FSlice.GetMeasureValueBSOnWayValues(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
```

```
      True, False, ADimValues, [])
  else
    Result := FSlice.GetMeasureValueBSOnWayValues(FOwner.FBaseLevel,
FOwner.FSecondLevel,
      FOwner.FIndexInBaseLevel, FOwner.FIndexInSecondLevel, FField.Index,
      FOwner.FBaseAdditionalTotalIndex, FOwner.FSecondAdditionalTotalIndex,
      False, True, [], ADimValues)
end;

{ TDimension }

constructor TDimension.Create(ASlice: TfcxSlice;
  AScriptItems: TScriptItems; ALevelInfo: TfcxAxisLevelInfo);
begin
  inherited Create(ASlice, AScriptItems, ALevelInfo.RegionField);
  FLevelInfo := ALevelInfo;
  FSubGroup := Nil;
end;

destructor TDimension.Destroy;
begin
  FSubGroup.Free;
  inherited;
end;

function TDimension.GetCurrentCaption: String;
begin
  Result := '';
  case GetRegion of
    rf_CapXAx: Result := FSlice.GetXAxisDimCaptionLI(FOwner.FXLevel,
FOwner.FIndexInXLevel, FLevelInfo.Level);
    rf_CapYAx: Result := FSlice.GetYAxisDimCaptionLI(FOwner.FYLevel,
FOwner.FIndexInYLevel, FLevelInfo.Level);
  end;
end;

function TDimension.GetCurrentValue: Variant;
begin
  Result := Null;
  case GetRegion of
    rf_CapXAx: Result := FSlice.GetXAxisDimValueLI(FOwner.FXLevel,
FOwner.FIndexInXLevel, FLevelInfo.Level);
    rf_CapYAx: Result := FSlice.GetYAxisDimValueLI(FOwner.FYLevel,
FOwner.FIndexInYLevel, FLevelInfo.Level);
  end;
end;

function TDimension.GetSubGroup: TDimension;
begin
  if FLevelInfo.LevelType = fcATLT_HasGroup then
  begin
    if FSubGroup = nil then
    begin
      case GetRegion of
        rf_CapXAx:
            if
(FSlice.XAxisContainer.LevelInfoWOMeasures[FLevelInfo.Level+1].RegionField =
FField) then
              FSubGroup := TDimension.Create(FSlice, FOwner,
FSlice.XAxisContainer.LevelInfoWOMeasures[FLevelInfo.Level+1]);
        rf_CapYAx:
            if
(FSlice.YAxisContainer.LevelInfoWOMeasures[FLevelInfo.Level+1].RegionField =
FField) then
```

```
              FSubGroup := TDimension.Create(FSlice, FOwner,
FSlice.YAxisContainer.LevelInfoWOMeasures[FLevelInfo.Level+1])
      end;
    end;
    Result := FSubGroup
  end
  else
    Result := nil;
end;

{ TScriptItems }

constructor TScriptItems.Create(ASlice: TfcxSlice);
begin
  inherited Create;
  FSlice := ASlice;
end;

{ TMeasures }

function TMeasures.GetCount: Integer;
begin
  Result := FSlice.MeasuresContainer.Count;
end;

function TMeasures.GetCurrentMeasureName: String;
begin
  Result := FSlice.MeasuresContainer.Measures[FCurrentMeasureIndex].Name
end;

function TMeasures.GetDetailValue(ARecordIndex: Integer; AFieldName: String):
Variant;
var
  AFieldIndex: integer;
begin
  if ARecordIndex < FSlice.MeasuresContainer.DetailRecordsCount[FBaseLevel,
FSecondLevel, FIndexInBaseLevel, FIndexInSecondLevel] then
  begin
      AFieldIndex :=
FSlice.SliceFields.Order[FSlice.SliceFields.IndexOfField(AFieldName)];
      if AFieldIndex <> -1 then
        Result :=
FSlice.SliceFields[AFieldIndex].GetUVValueFromRec(FSlice.MeasuresContainer.Detai
lRecords[FBaseLevel, FSecondLevel, FIndexInBaseLevel, FIndexInSecondLevel,
ARecordIndex])
      else
        Result := Unassigned;
  end
  else
    Result := Unassigned;
end;

function TMeasures.GetItem(AIndex: Integer): TMeasure;
begin
  if AIndex < Count then
    Result := GetItemByName(FSlice.MeasuresContainer.Measures[AIndex].Name)
  else
    Result := nil;
end;

function TMeasures.GetItemByCaption(AIndex: String): TMeasure;
var
  i: Integer;
begin
  Result := nil;
```

```
  for i := 0 to FSlice.MeasuresContainer.Count - 1 do
    if AnsiCompareText(FSlice.MeasuresContainer.Measures[i].Caption, AIndex) = 0
then
    begin
      Result := GetItemByName(FSlice.MeasuresContainer.Measures[i].Name);
      break;
    end;
end;

function TMeasures.GetItemByName(AIndex: String): TMeasure;
var
  i, idx: Integer;
  Measure: TMeasure;
begin
  idx := IndexOf(AIndex);
  if idx = - 1 then
  begin
    for i := 0 to FSlice.MeasuresContainer.Count - 1 do
      if AnsiCompareText(FSlice.MeasuresContainer.Measures[i].Name, AIndex) = 0
then
      begin
        Measure := TMeasure.Create(FSlice, Self,
FSlice.MeasuresContainer.Measures[i]);
        idx := FItems.AddObject(FSlice.MeasuresContainer.Measures[i].Name,
Measure);
        break;
      end;
  end;
  if idx = - 1 then
    Result := nil
  else
    Result := TMeasure(FItems.Objects[idx]);
end;

function TMeasures.GetRecordCount: integer;
begin
  Result := FSlice.MeasuresContainer.DetailRecordsCount[FBaseLevel,
FSecondLevel, FIndexInBaseLevel, FIndexInSecondLevel];
end;

{ TDimensions }

function TDimensions.GetCount: Integer;
begin
  Result :=
    FSlice.FieldsOfRegion[rf_Page].Count +
    FSlice.YAxisContainer.LevelCount +
    FSlice.XAxisContainer.LevelCount;
end;

function TDimensions.GetItem(AIndex: Integer): TDimension;

  function CheckHere(AFields: TfcxCommonFieldsOfRegion; var Index: Integer):
String;
  begin
    AIndex := AIndex - AFields.Count;
    if AIndex < 0 then
      Result := AFields.Items[AIndex + AFields.Count].Name
    else
      Result := '';
  end;
  function CheckHere2(AAxis: TfcxAxisContainer; var Index: Integer): String;
  begin
    AIndex := AIndex - AAxis.LevelCount;
    if AIndex < 0 then
```

```
Result := AAxis.LevelInfoWOMeasures[AIndex + AAxis.LevelCount].RegionField.Name
    else
      Result := '';
  end;

var
  AName: String;
begin
  Result := nil;
  if AIndex < Count then
  begin
    AName := CheckHere(FSlice.FieldsOfRegion[rf_Page], AIndex);
    if AName = '' then
      AName := CheckHere2(FSlice.YAxisContainer, AIndex);
    if AName = '' then
      AName := CheckHere2(FSlice.XAxisContainer, AIndex);
    if AName <> '' then
      Result := GetItemByName(AName);
  end;
end;

function TDimensions.GetItemByCaption(AIndex: String): TDimension;
var
  i: Integer;
begin
  Result := nil;
  for i := 0 to FSlice.FieldsOfRegion[rf_Page].Count - 1 do
    if AnsiCompareText(FSlice.FieldsOfRegion[rf_Page].Items[i].Caption, AIndex)
= 0 then
    begin
      Result := GetItemByName(FSlice.FieldsOfRegion[rf_Page].Items[i].Name);
      Exit;
    end;
  with FSlice.YAxisContainer do
    for i := 0 to LevelCount - 1 do
      if AnsiCompareText(LevelInfoWOMeasures[i].RegionField.Caption, AIndex) = 0
then
      begin
        Result := GetItemByName(LevelInfoWOMeasures[i].RegionField.Name);
        Exit;
      end;
  with FSlice.XAxisContainer do
    for i := 0 to LevelCount - 1 do
      if AnsiCompareText(LevelInfoWOMeasures[i].RegionField.Caption, AIndex) = 0
then
      begin
        Result := GetItemByName(LevelInfoWOMeasures[i].RegionField.Name);
        Exit;
      end;
end;

function TDimensions.GetItemByName(AIndex: String): TDimension;
var
  i, idx: Integer;
  Dimension: TDimension;
  ALevelInfo: TfcxAxisLevelInfo;
begin
  ALevelInfo.IsMeasure := False;
  ALevelInfo.IsVisible := True;
  ALevelInfo.LevelType := fcATLT_Simple;
  ALevelInfo.RegionField := nil;
  idx := IndexOf(AIndex);
  if idx = - 1 then
  begin
    for i := 0 to FSlice.FieldsOfRegion[rf_Page].Count - 1 do
```

```
      if AnsiCompareText(FSlice.FieldsOfRegion[rf_Page].Items[i].Name, AIndex) =
0 then
      begin
        ALevelInfo.RegionField :=
TfcxAxisField(FSlice.FieldsOfRegion[rf_Page].Items[i]);
        Dimension := TDimension.Create(FSlice, Self, ALevelInfo);
        idx := FItems.AddObject(AIndex, Dimension);
        Break;
      end;
  end;
  with FSlice.YAxisContainer do
    if idx = - 1 then
    begin
      for i := 0 to LevelCount - 1 do
        if AnsiCompareText(LevelInfoWOMeasures[i].RegionField.Name, AIndex) = 0
then
        begin
          Dimension := TDimension.Create(FSlice, Self, LevelInfoWOMeasures[i]);
          idx := FItems.AddObject(AIndex, Dimension);
          Break;
        end;
    end;
  with FSlice.XAxisContainer do
    if idx = - 1 then
    begin
      for i := 0 to LevelCount - 1 do
        if AnsiCompareText(LevelInfoWOMeasures[i].RegionField.Name, AIndex) = 0
then
        begin
          Dimension := TDimension.Create(FSlice, Self, LevelInfoWOMeasures[i]);
          idx := FItems.AddObject(AIndex, Dimension);
          Break;
        end;
    end;
  if idx = - 1 then
    Result := nil
  else
    Result := TDimension(FItems.Objects[idx]);
end;

function CreateDimensions(ASlice: TfcxSlice): TPersistent;
begin
  Result := TDimensions.Create(ASlice);
end;

function CreateMeasures(ASlice: TfcxSlice): TPersistent;
begin
  Result := TMeasures.Create(ASlice);
end;

function CreateCustomObject(ASlice: TfcxSlice): TPersistent;
begin
  Result := TfcxCustomObject.Create;
end;

function TDimensions.GetXAxisItem(AIndex: Integer): TDimension;
begin
  Result :=
GetItemByName(FSlice.XAxisContainer.LevelInfoWOMeasures[AIndex].RegionField.Name
);
  if (Result <> nil) and (Result.FLevelInfo.Level <> AIndex) then
    Result := Result.SubGroup;
end;

function TDimensions.GetXAxisLevelsCount: TfcxSmallCount;
```

```
begin
  Result := FSlice.XAxisContainer.LevelCount;
end;

function TDimensions.GetYAxisItem(AIndex: Integer): TDimension;
begin
  Result :=
GetItemByName(FSlice.YAxisContainer.LevelInfoWOMeasures[AIndex].RegionField.Name
);
  if (Result <> nil) and (Result.FLevelInfo.Level <> AIndex) then
    Result := Result.SubGroup;
end;

function TDimensions.GetYAxisLevelsCount: TfcxSmallCount;
begin
  Result := FSlice.YAxisContainer.LevelCount;
end;

{ TCommonScriptItems }

procedure TCommonScriptItems.ClearItems;
var
  i: integer;
begin
  for i := FItems.Count - 1 downto 0 do
  begin
    FItems.Objects[i].Free;
    FItems.Delete(i);
  end;
end;

constructor TCommonScriptItems.Create;
begin
  FItems := TStringList.Create;
end;

destructor TCommonScriptItems.Destroy;
begin
  ClearItems;
  FItems.Free;
  inherited;
end;

function TCommonScriptItems.GetCount: Integer;
begin
  Result := 0;
end;

function TCommonScriptItems.IndexOf(AString: String): Integer;
begin
  for Result := 0 to FItems.Count - 1 do
    if AnsiCompareText(FItems[Result], AString) = 0 then Exit;
  Result := -1;
end;
{ TfcxCustomObject }
function TfcxCustomObject.GetValueIsNil: Boolean;
begin
  Result := not Assigned(FValue);
end;

end.
```