Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи моніторингу біомедичної інформації”

КбПЗ_2023

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Гаєвський В.С.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 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

Гаєвському Віктору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи моніторингу біомедичної інформації*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 37-13 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 моніторингу біомедичної інформації*

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*

КОПІЯ 2023

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.	

КОПЗ_2023

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

Гаєвський В.С. Дослідження та програмна реалізація системи моніторингу біомедичної інформації. 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 моніторингу біомедичної інформації.

Метою розробки є дослідження та програмна реалізація системи моніторингу біомедичної інформації.

Об'єктом дослідження є процес моніторингу біомедичної інформації.

Предметом дослідження є методи моніторингу біомедичної інформації.

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

Результат роботи – програмна реалізація системи моніторингу біомедичної інформації.

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

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

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

Програму розроблено в середовищі Delphi 10.

Ключові слова: комп'ютерні науки, біомедична інформація

КОПЗ – 2023

ABSTRACT

Haievskyi V.S. Research and software implementation of the biomedical information monitoring system. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software designed for the biomedical information monitoring system was developed.

The purpose of the development is research and software implementation of the biomedical information monitoring system.

The object of the study is the process of monitoring biomedical information.

The subject of research is methods of monitoring biomedical information.

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

The result of the work is the software implementation of the biomedical information monitoring 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 environment.

Keywords: computer science, biomedical information

КОПЗ – 2023

ЗМІСТ

КОПЗ_2023

КОПЗ-2023

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БМІ	–	біомедична інформація
БМС	–	біомедичні системи
ВМП	–	віртуальний медичний пристрій
ЕКГ	–	електрокардіограма
ЕЕГ	–	електроенцефалограма
ЕОГ	–	електроокулограма
ЕМГ	–	електроміограма
ЕРГ	–	електроретинограма
ОС	–	операційна система
ПЗ	–	програмне забезпечення
ЦОС	–	цифрова обробка сигналів

КбПЗ_2023

ВСТУП

Актуальність теми. Наразі комп'ютерні технології почали широко застосовуватися для дослідження біологічних систем і феноменів, таких як електрична активність серцево-судинної системи, мозку, нейромишечної системи й ін. Сучасною тенденцією є кількісний і об'єктивний аналіз біомедичних систем і феноменів через аналіз сигналів [1]. Методи цифрової обробки біомедичних сигналів, що характеризують такі системи, створюються з обліком їхніх специфічних особливостей. Аналіз сигналу від біомедичної системи не є простим завданням, важлива інформація в сигналі часто замаскована шумами й наведеннями, має місце його варіабельність, спостерігається крайня мінливістю й розмаїтість ознак у біомедичних сигналах і системах у порівнянні, наприклад, з фізичними системами або спостереженнями. Ці фактори визначають актуальність розробки спеціальних методів для об'єктивного аналізу біомедичних сигналів з використанням алгоритмів обробки, реалізованих на комп'ютері [4]. Такі методи можуть бути засновані на комп'ютерному моделюванні, сутність якого – у побудові моделі, що представляє собою програмний комплекс, що алгоритмічно описує поводження об'єкта або розвиток процесу. Важливо також, що комп'ютерний аналіз біомедичних сигналів, якщо він виконується з використанням адекватної логіки, потенційно здатний підсилити об'єктивну складову інтерпретації, що дається експертом [2].

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи моніторингу біомедичної інформації.

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

– Огляд існуючих систем моніторингу біомедичної інформації.
– Дослідження системи моніторингу біомедичної інформації.
– Програмна реалізація системи моніторингу біомедичної інформації.

КПЗ–2023

Об'єктом дослідження є процес моніторингу біомедичної інформації.

Предметом дослідження є методи моніторингу біомедичної інформації.

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

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

– Удосконалено метод моніторингу біомедичної інформації.

– Розроблено вітчизняний продукт моніторингу біомедичної інформації, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу біомедичної інформації.

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 моніторингу біомедичної інформації, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

КОПІЯ 2023

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Тенденцією теперішнього часу є створення й використання для обробки, практично, кожного типу квазіперіодичних (зокрема – біомедичних) сигналів (або групи близьких за формою сигналів) окремої, розробленої тільки для нього математичної моделі [4, 6, 7]). Найчастіше, така розмаїтість моделей не є виправданою, вона утрудняє порівняльний аналіз результатів моделювання, ускладнює завдання цілеспрямованого проведення досліджень необхідністю освоєння великої кількості різнотипних методів. Отже, для підвищення ефективності обробки потоків такого роду інформації, а часто й у реальному масштабі часу, потрібна розробка й реалізація математичних методів моделювання, не тільки оптимальних з погляду швидкості одержання кінцевого результату, використання мінімуму комп'ютерної пам'яті або інших ресурсів і забезпечення потрібної точності, але й які дають можливість всебічного й швидкого зіставлення моделей для оцінки якості одержуваних результатів. Розробка сучасних комп'ютерних методів моделювання, створення нових моделей і програмних комплексів, їхнє застосування для аналізу біомедичної інформації (наприклад, сигналів електрокардіограм – ЕКГ і електроенцефалограм – ЕЕГ) є в цей час актуальним: Так, існує досить багато моделей, заснованих на застосуванні математичної теорії сплесків [3, 5]. Є десятки їхніх різних модифікацій, у яких використовуються різні набори параметрів і припущень, тому важко зіставляти точність одержуваних з їхньою допомогою результатів і оцінити обґрунтованість сформульованих виводів [2, 4]. Через недостатню дослідженість актуальним є визначення найбільш ефективних дискретних сплесків для цифрової обробки тривалого біомедичного сигналу.

КОПЗ-2023

1.2 Область застосування

Опираючись на результати аналізу проблем, що виникають при обробці специфічних біомедичних сигналів, і численні літературні джерела по даній тематиці було визначено, як побудувати найбільш перспективні для використання в даній роботі моделі, щоб забезпечити досягнення поставлених у роботі цілей і одержати необхідні результати найбільш обґрунтованим і простим способом. Були розглянуті методи, засновані на теорії сплесків, які стали широко застосовуватися в дослідженнях порівняно недавно, однак встигли зарекомендувати себе як потужний апарат для комплексного аналізу сигналів найрізноманітнішої структури. У цей час існують десятки модифікацій сплесків (як безперервних, так і дискретних), ведеться активне їхнє застосування при обробці інформації. Що стосується дискретного перетворення Фур'є (ДПФ), то він є одним із класичних і часто вживаних методів для обробки цифрових сигналів і відрізняється достатньою простотою використовуваних алгоритмів, наочною алгебраїчною схемою.

У багатьох наукових статтях і монографіях затверджується, що сплески краще пристосовані для роботи з оцифрованими сигналами, чим ДПФ. Однак, як показав проведений аналіз, окремі технічні деталі організації обчислень можуть грати істотну, а іноді й визначальну роль. Через те, що існує безліч модифікацій і сплесків, і ДПФ, подібне твердження вимагає ретельного аналізу використовуваних методів і алгоритмів. У підсумку проведеного аналізу в якості базових для побудови моделей були обрані сплески Добеші й безперервне вейвлет-перетворення, і для порівняння – дискретне перетворення Фур'є.

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

КПЗ – 2023

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

Програмне забезпечення для візуалізації та аналізу клітин і тканин

Зображення клітин є одним із найбільш застосовуваних методів отримання інформації та надання візуальних доказів їх присутності в живих клітинах. Вдосконалені мікроскопи можуть бачити деталі аж до дифракційних меж або кілька сотень нанометрів або навіть зменшуватися за допомогою методів суперроздільності. Необхідний аналіз зображення, такий як кількісне денситометричне визначення, визначення довжини клітини, розміру та кількості клітин, відносної інтенсивності флуоресценції різних флуорофорів, і дані використовуються для порівняння між зразками

Контроль мікроскопа та базовий аналіз зображень

Щоб зробити зображення, мікроскопом і його компонентами потрібно керувати за допомогою програмного забезпечення. Це програмне забезпечення або надається основними конструкторами мікроскопів, або може бути рішенням з відкритим кодом. Усе програмне забезпечення пропонує варіанти аналізу зображень, які можуть бути базовими (2D, візуалізація уповільненої зйомки) або більш складними (3D, деконволюція, автоматизація).

Zeiss AxioVision і ZEN Blue/Black

Власне програмне забезпечення Carl Zeiss Microscopy доступне у версіях для біології та матеріалів. Одним із перших програм Zeis був AxioVision for Biology на базі.net framework (Windows), який виконує 5 основних функцій: отримання зображень; обробка зображення; аналіз зображення; архівація зображень; та налаштування програмного забезпечення. Кожній функції, у свою

КОПЗ-2023

чергу, допомагають різні модулі, які виконують додаткові завдання, щоб отримати більше інформації з предмета дослідження. Наприклад, із 14 модулів у системі отримання зображень назва модуля подвійної камери дозволяє використовувати два модулі одночасно, мікроматриці тканин пропонують широкомасштабний аналіз, панорама надає загальне уявлення про отримані зображення, крім високої чіткості, висока швидкість і модулі швидкого збору, які можуть допомогти отримати великий обсяг інформації за коротші проміжки часу з високою роздільною здатністю. AxioVision пропонує такі методи обробки зображень, як деконволюція, формування зображень плюс, щоб покращити якість зображень і забезпечити 4-й вимір часу для 3D-зображень за допомогою модуля Inside4D. Модуль ASSAYbuilder дозволяє виконувати скринінг і аналіз високого вмісту, пов'язаний із тестами на основі клітин, такими як цитотоксичність, апоптоз, диференціація клітин, локалізація молекул і експресія репортера. Можна отримати до 50 різноманітних параметрів 3D-вимірів, що дозволяє проводити інтерактивні вимірювання. Програмне забезпечення може виконувати автоматичні вимірювання та бути запрограмованим на те саме за допомогою процедур програмування. Він також може виконувати відстеження клітин і забезпечувати відповідні вимірювання. Модуль фізіології має функції FRET і контролює флуоресцентні сигнали всіх компонентів мікроскопа. Він може знімати близько 140 кадрів в секунду за допомогою AxioVision HS. Visual Basic for Applications (VBA) для AxioVision надає свободу рук у своїх власних межах для розробки програм на основі функціональних можливостей. Зрештою, отримані та проаналізовані зображення можна записати відповідно до стандартів FDA на основі Регламенту 21 CFR, частина 11. AxioVision GxP автоматично запускається після відкриття ОС і перевіряє жорсткі диски на наявність файлів і папок, завдяки чому може відстежувати та документувати кожну виконану транзакцію. в окремих файлах XML.

Програмне забезпечення Carl Zeiss AxioVision, наприклад, використовувалося для дослідження ролі гібридних клітин Th1/2 у природних

КПЗ-2023

імунних відповідях проти паразитів [9], ролі CDON у регуляції виживання пухлинних клітин [10], для аналізу морфології шкірних сенсорних нейронні дужки у волосистій шкірі миші [11] і клітинне походження дорослих функціональних кровоносних судин [12], щоб вивчити амінокислотний гомеостаз [13], імунну толерантність плода [14] та інгібування трансляції білка miR430 у рибок даніо [13]. 15]. Багатовимірний режим збору даних AxioVision використовувався для отримання сповільненої зйомки зображень для вивчення сигналізації кальцію за допомогою PKG, а його фізіологічний модуль використовувався для аналізу флуоресценції окремих окінетів [16].

Окрім AxioVision, також часто цитували Zeiss LSM Image Browser. Він використовувався для виконання редокс-візуалізації в реальному часі для вивчення регуляції нейронів SCN через циркадні зміни окисно-відновного стану [17], серед інших застосувань [13, 18-23].

В останні роки програмне забезпечення AxioVision поступово замінюється на ZEN Blue і ZEN Black (таблиця 3). ZEN Blue в основному використовувався для керування широкопольними мікроскопами Zeiss. Нині воно замінює програмне забезпечення ZEN Black, яке керує конфокальними мікроскопами. Він пропонує 2D аналіз і опцію 3D візуалізації. В якості опції є алгоритм деконволюції. Однією з важливих особливостей ZEN є Experimental Designer, який створює блоки, які можуть автоматизувати конвеєрне отримання та аналіз.

Елементи NIS

Програмне забезпечення Nikon – це NIS Elements. Він керує мікроскопами Nikon. Він пропонує 2D аналіз, 3D візуалізацію та модуль деконволюції. Дуже цікавим модулем NIS є JOBS, конструктор експериментів, який пропонує автоматизацію завдань і досить простий у використанні.

LAS

Програмне забезпечення для контролю та аналізу мікроскопа Leica є LAS. Він керує всіма мікроскопами Leica. Він пропонує стандартний 2D-аналіз, 3D-візуалізацію, деконволюцію та деякі основні завдання аналізу, як-от побудова

КОПІЯ 2023

профілів інтенсивності, вимірювання відстаней тощо.

CellSens

Програмне забезпечення Olympus – CellSens. Він пропонує базовий 2D аналіз, 3D візуалізацію, деконволюцію. Завдяки Graphic Experiment Management і Well Navigator він пропонує деякі основні завдання автоматизації. Його властивою характеристикою є те, що це дуже візуальне програмне забезпечення, досить просте для розуміння та використання.

iQ3 – Fusion

Останніми роками Andor комерціалізує деякі мікроскопи (в основному обертові диски). Програмне забезпечення для контролю та аналізу спочатку було iQ3. Це базове програмне забезпечення, і можна легко встановити відповідний протокол отримання. Для найновіших мікроскопів Andor, таких як серія Dragonfly, Andor пропонує програмне забезпечення Fusion, яке є простішим у використанні та має кілька цікавих модулів аналізу, як-от 3D-візуалізація в поєднанні з програмним забезпеченням Imaris і модулем деконволюції.

SoftWorx

Softworx – це програмне забезпечення для візуалізації та аналізу від Applied Precision (GE Healthcare). Він поєднує в собі інструменти перегляду, обробки й аналізу зображень із програмним забезпеченням для керування даними в простому у використанні форматі. Програмне забезпечення постачається з програмою перегляду в реальному часі з вибором каналу, z та керуванням часом. Він може виконувати 4D вимірювання та анімацію, працювати в пакетному режимі та надсилати, отримувати доступ через FTP/SFTP. Він виконує видалення базової лінії, вимірювання з вільними точками, точками зрізів і точками поверхні, виконує корекцію та відновлення зображення шляхом деконволюції. Хоча його переваги відчуваються в пакеті аналізу спільної локалізації, де легко виділити, візуалізувати та кількісно визначити регіональне перекриття в 3D і 4D зображеннях з результатами, представленими двома способами, новим 3D або 4D каналом, а також статистичним звітом. Наприклад, Applied Precision softWoRx v

КПЗ – 2023

5.0 використовувався для виконання реконструкції та вирівнювання зображень 3D-SIM для вивчення молекулярної архітектури насоса SpoIIIE та механізму його набору та складання [24]. Обмежень багато, оскільки це не всеохоплюючий інструмент для виконання широкого спектру досліджень, але обмежений.

Метаморф

Програмне забезпечення MetaMorph від Molecular Devices спочатку було включено багатьма постачальниками обладнання для мікроскопів, перш ніж постачальники почали пропонувати власну версію програмного забезпечення для керування мікроскопом і аналізу зображень. MetaMorph може читати бібліотеки (DLL) компонентів мікроскопа, тому для керування мікроскопом потрібно мати версію MetaMorph, яка розпізнає його компоненти. З роками MetaMorph зможе надавати оновлення для нових компонентів мікроскопа (камери, столики тощо), щоб програмне забезпечення продовжувало контролювати нові мікроскопи.

MetaMorph – це базове програмне забезпечення, просте у використанні, яке в той же час пропонує багато варіантів аналізу.

Багатовимірний модуль збору даних дозволяє виконувати складні послідовності збору даних за допомогою простого та керованого інтерфейсу користувача. Отримане зображення або відео одночасно передається в пам’ять, щоб полегшити апаратне прискорення для захоплення зображення. Зображення пов’язані з відповідними графіками та таблицями, що допомагає користувачеві співвідносити зображення та дані. Модуль Live Replay, який підтримується цією функцією копіювання назад у пам’ять, може допомогти захопити події в реальному часі з FRET, FRAP, живих зображень і експериментів із сповільненою зйомкою. Модуль сканування слайдів допомагає зшивати серію зображень із великої частини зразка тканини. надання відтворюваного зображення високої роздільної здатності, що зменшує помилку експериментів з обробкою землі. Наприклад, програмне забезпечення Molecular Devices MetaMorph було використано для дослідження впливу експресії цитокінів на диференціацію Т-хелперних клітин [25], ролі Vav2 і Vav3 у раку шкіри [26], петлі ДНК,

КОПЗ_2023

індукованої факторами транскрипції в E. coli [27], регуляторна роль Erf2 у пальмітоілуванні білка та мейотичному вході в Schizosaccharomyces pombe [28], точне відстеження часу, контрольоване циклічним активатором у Arabidopsis [29], роль PICK1 та ICA69 у регулюванні утворення та дозрівання гранул інсуліну [30], роль клітинної адгезії та натягу кори головного мозку в сортуванні клітин під час гаструляції [31], регуляція ретроградного синаптичного сигналу за допомогою NRX-1 і NLG-1 у C.elegans [32], роль шляху Ras/Epac2 у підтримці базисної дендритної складності кортикальних нейронів [33] і залучення септину рисової грибкової інфекції [21]. Побратими MetaMorph, Metavue і MetaXpress, використовувалися для конфокальної візуалізації для вивчення пасивного адвективного транспорту [34] і для захоплення широкопольних зображень для вивчення ролі XIAP у інгібуванні каспази після диференціювання скелетних м'язів [35].

Labview

Інше програмне забезпечення, яке керує мікроскопами, це Labview (National Instruments). Labview – це платформа системного проектування та середовище розробки для мови візуального програмування. Він пропонує графічний підхід до програмування, який допомагає візуалізувати кожен аспект програми, спрощуючи інтеграцію апаратного забезпечення від постачальників і розробку індивідуальних інтерфейсів користувача. Це програмне забезпечення, більшою мірою призначене для розробників мікроскопів, які хочуть мати гнучкість у дизайні керування компонентами. Подібно до MetaMorph, Labview має розпізнавати компоненти з їх DLL. Він може запропонувати призначені для користувача інтерфейси для керування компонентами, візуалізації та автоматизації завдань обробки зображень.

Мікроменеджер

Micromanager – це програмний пакет, який може керувати мікроскопами. Це безкоштовне програмне забезпечення, яке можна поєднувати з програмним забезпеченням ImageJ з відкритим кодом, щоб забезпечити повне рішення для

КОПІЯ 2023

керування мікроскопами та пропонувати модулі аналізу зображень. Micromanager може читати конкретне обладнання мікроскопа. Якщо апаратне забезпечення, яким потрібно керувати, не входить до списку підтримуваних мікроменеджерів, можна звернутися безпосередньо до спільноти мікроменеджерів і, можливо, можна зробити розвиток для конкретної апаратної підтримки.

Програмне забезпечення для 3D візуалізації та аналізу

Дані, отримані під час отримання мікроскопом, необхідно візуалізувати та проаналізувати. Коли застосовуються методи 3D-зображення або навіть 4D (проміжок часу), дані стають великими за розміром, і їх складніше спочатку візуалізувати, а потім проаналізувати. В останні роки існує купа програмного забезпечення, присвяченого 3D та 4D візуалізації великих даних. Великі набори даних для мікроскопії мають терабайтний діапазон. З розвитком мікроскопії та появою методів візуалізації, таких як мікроскопія світлового листа, виробництво великих даних стає неминучим, а візуалізація та обробка великих даних є більш ніж необхідною. У таблиці 4 показано основне програмне забезпечення для 3D/4D візуалізації та аналізу для мікроскопії.

Програмне забезпечення можна розділити на 2 великі родини. З одного боку, ми знаходимо комерційні рішення, а з іншого – рішення, що надходять від дослідницьких лабораторій, які часто мають відкритий код.

Комерційне програмне забезпечення 3D/4D

Що стосується комерційних рішень, то найбільш використовуваним є програмне забезпечення Imaris від Bitplane. Це інтерактивне програмне забезпечення для візуалізації та аналізу 3D-зображень і мікроскопічних зображень із проміжком часу з передовими рішеннями для великих наборів даних.

Він використовує формат файлу на основі стандарту HDF5 (Hierarchical Data Format 5), який дозволяє використовувати бібліотеки та інструменти HDF5 для легкої візуалізації набору даних, а потім для більш ретельного аналізу, коли потрібні додаткові деталі. Він зберігає не лише оригінальні дані зображення, але

КПІ – 2023

й версії вихідних даних із меншою роздільною здатністю. Це дозволяє програмному забезпеченню візуалізації завантажувати дані лише з низькою роздільною здатністю, якщо їх достатньо.

Паралельно Imaris пропонує низку рішень для аналізу, як-от автоматизоване виявлення клітин і органел, відстеження ниток для нейронних структур і не тільки, аналіз колокалізації, алгоритми відстеження окремих частинок, деконволюція, зшивання плиток для зображення великої площі. У той же час він може створювати посилання на спеціальний аналіз за допомогою іншого програмного забезпечення, наприклад Matlab, Java, R або Python.

Однією з головних незручностей Imaris є те, що потрібен дуже потужний ПК, оскільки для великих даних споживається велике оперативна пам'ять і дисковий простір. Список опублікованих робіт команд, які використовують imaris для 3D-візуалізації та аналізу, дуже довгий, оскільки зараз це, ймовірно, найбільш використовуване програмне забезпечення для цих програм.

Ще одним потужним комерційним програмним забезпеченням для візуалізації 3D/4D є Vision4D від Arivis (табл. 4). Він пропонує високу інтерактивність візуалізації навіть на стандартних ПК із простою підтримкою 3D-графіки. Можна встановити робочі процеси за допомогою Matlab і Python. Завдяки ефективній і легкій візуалізації 4D-даних у 2019 році стало можливим візуалізувати зображення, отримані з комбінованої мікроскопії розширення та решітчастої світлової мікроскопії. Вони проливають світло на нейронні ланцюги в усьому мозку дрозофіли та первинній соматосенсорній корі головного мозку миші [36].

Старішим програмним забезпеченням для 3D-візуалізації є Volocity від Quorum Technologies. Він пропонує безкоштовний переглядач і простий у використанні. З іншого боку, він не пропонує багато модулів аналізу.

Зовсім недавнє та інноваційне програмне забезпечення для 2D-5D аналізу та візуалізації – це Aivia (DRVISION). Aivia пропонує цікаві рішення для критичних завдань, таких як відображення великих зображень і аналіз складних

КОПЗ-2023

біологічних явищ. Він базується на низці технологій машинного навчання для сегментації зображень, класифікації об'єктів і виявлення новизни. Він ще не пов'язаний з багатьма модулями аналізу.

Безкоштовні 3D/4D рішення

Інше велике сімейство програмного забезпечення для візуалізації та аналізу 3D/4D – це рішення, які надходять від дослідницьких лабораторій і найчастіше мають відкритий код. Є три різні плагіни, готові до використання та реалізовані в програмному забезпеченні аналізу зображень ImageJ/Fiji (на основі Java). Першим і найпоширенішим є плагін BigSataViewer, розроблений дослідником Т. Піцшем (MPI-CBG Dresden) [37]. Він досить добре обробляє великі дані, реалізовані у форматі HDF5. Це рішення для візуалізації з невеликою кількістю варіантів аналізу. Другий плагін – це ClearVolume, ініційований Л. Роєром [38]. Це плагін для візуалізації, який дозволяє 3D-спостереження зразків у реальному часі в режимі потокової передачі даних. Він має спеціальний інтерфейс для керування програмним забезпеченням micromanager/OpenSpim, щоб керувати цими світловими інструментами в реальному часі. Третій плагін – це SciView від К. Харінгтона (Університет Айдахо) та У. Гюнтера (MPI-CBG). Цей плагін забезпечує тривимірну візуалізацію та представлення віртуальної реальності.

Ще одне програмне забезпечення, призначене для візуалізації та аналізу 3D/4D зображень, – Vaa3D. Він підтримується як HHMI – Janelia Research Campus, так і Allen Institute for Brain Science, і використовується в ряді проектів по всьому світу [39]. Це відкритий код. Він швидкий і пропонує різні плагіни для подальшого аналізу зображень.

Перспективним програмним забезпеченням для інтерактивної візуалізації об'ємних даних, що минув у часі, створених світловими мікроскопами, є Spimagine, що використовує пакет python (M. Weigert, MPI-CBG). Пакет надає загальний засіб перегляду даних 3D+t і використовує прискорення GPU через OpenCL, пропонуючи швидке рішення візуалізації.

КПЗ-2023

Програмне забезпечення для аналізу та проектування молекулярної біології

Перераховані основні постачальники та їхнє основне програмне забезпечення для додатків молекулярної біології, цитоване серед статей, які досліджувала Labome. Як і програмне забезпечення для зображень клітин і тканин, більшість постачальників програмного забезпечення для молекулярної біології також надають апаратні системи, такі як секвенатори та прилади для ПЛР. Інші програми для молекулярної біології, такі як Blast2GO [40], набувають популярності.

Applied Biosystems є одним із основних постачальників інструментів для молекулярної біології, таких як ПЛР або кПЦР, і секвенаторів генів. Applied Biosystems Primer Express і програмне забезпечення SDS використовувалися для ПЛР або кПЦР для вивчення інгібування експресії генів осциляторів [41] і організації експресії генів [42], серед іншого [43-55]. І його GeneMapper цитувався в статтях для сканування генів [56-58].

Illumina BeadStudio, CASAVA, ELAND і Pipeline використовувалися для генотипування SNP і секвенування ChIP, екзома, транскриптома та РНК [41, 59-70].

Програми Bio-Rad Opticon і LaserSharp були використані для ПЛР для вивчення молекулярного механізму консолідації пам'яті [71], серед іншого [71-76].

SoftGenetics, на відміну від усіх перерахованих вище постачальників, є програмною компанією без будь-яких апаратних продуктів. Його програми Mutation Surveyor і GeneMarker використовувалися для аналізу секвенування ДНК [77-79], генотипування [80], мікросателітного аналізу [81] та аналізу довжини хвоста полі(А) [15].

Gene Codes Corporation, компанія з біоінформатики, надає секвенсор для аналізу секвенування ДНК або РНК. Він цитувався в ряді публікацій [60, 81-84, 84].

КПЗ-2023

Програмне забезпечення для аналізу проточної цитометрії

Дві статті Labome розглядають теоретичні основи та практичні міркування проточної цитометрії та FACS. Програмне забезпечення для проточної цитометрії Tree Star FlowJo та BD домінує в цій галузі.

Tree Star FlowJo

Аналіз проточною цитометрією передбачає ручну організацію, аналіз і створення звіту з файлами FCS, отриманими з проточного цитометра. Цей аналіз, як правило, включає велику кількість зразків, і FlowJo є повним програмним забезпеченням, яке може допомогти на всіх рівнях аналізу. На початку процесу програмне забезпечення генерує шаблони робочих областей, заощаджуючи час організації вибірки, стратегію аналізу та параметри виводу звіту. Після цього FlowJo допомагає в пакетному аналізі зразків, підтримує узгодженість аналізу для SOP, скорочує час простою за рахунок усунення ручного аналізу та створення звітів, тим самим збільшуючи потужність лабораторії для обробки високопродуктивних аналізів. FlowJo отримує файли для аналізу, виконує автоматичну компенсацію за допомогою обробки сигналу та обчислює гейти та статистику перед створенням електронних таблиць і звітів із графіками. FlowJo може аналізувати дані, отримані будь-яким проточним цитометром будь-якого виробника.

FlowJo надає статистичні дані з великої кількості зразків, які можна швидко згенерувати, порівняти та експортувати. Він також може виконувати спеціалізовані аналізи, такі як ДНК/клітинний цикл, кінетика (потік кальцію), проліферація, калібрування та статистичне порівняння, і представляє вихідні дані як у табличному, так і в графічному форматі, використовуючи складні інструменти представлення даних для створення якісного матеріалу для публікації. Дані придатні для експорту в різні формати, готові для аналізу іншими пакетами або програмним забезпеченням. FlowJo виконує загальний робочий процес, який включає завантаження зразків, групування їх за допомогою загальних процедур, детальний аналіз простого зразка прототипу для

КПЗ_2023

виправлення стробування та статистики, застосування відповідних аналізів до всіх зразків, перевірку правильності стробування для аналізованих зразків, генерування графічного результату та, нарешті, таблицю з конкретною статистикою з усіх зразків. Пакет є одним із найповніших і найбільш цитованих програм у цій категорії, дуже мало можна віднести до того, що вони проти цього програмного забезпечення, яке забезпечує велику підтримку користувачів і базу знань через свою домашню сторінку та блог.

Наприклад, програмне забезпечення TreeStar FlowJo було використано для аналізу даних проточної цитометрії, щоб дослідити дизайн безмембранних органел для включення неприродних амінокислот [89], важливість регуляції рівнів шлюбу для переходу ооцитів до ембріона у дрозофіли [89]. 90], роль гібридних клітин Th1/2 у природних імунних відповідях проти паразитів [9], важливість передачі сигналу TCR для розвитку природних Т-клітин-кілерів [91], регуляторна роль Erf2 у пальмітоілуванні білка та мейотичному проникненні в Schizosaccharomyces pombe [28], роль Lsd1 під час дозрівання клітин крові [92], вплив мутацій JAK2 на гемопоетичні стовбурові клітини [93], індукована вірусом еволюція рецептора трансферину [94], роль miR-146a в кровотворенні миші стовбурових клітин [95], різних механізмів передачі ендосомальних TLR, опосередкованих UNC93B1 [96], і механізму імунного ухилення від вірусу коров'ячої віспи [97]. Деякі статті цитували FlowJo, наданий також BD [98, 99], наприклад FlowJo v10 [99].

BD

CellQuest

CellQuest Pro, власне програмне забезпечення від BD Biosciences, дозволяє користувачеві отримувати та аналізувати дані безпосередньо з проточного цитометра. Він здійснює заходи перевірки якості, такі як оптимізація електроніки для зразків, дозволяючи налаштовувати детектори FSC, SSC, стробувати для популяції, що цікавить, і виконувати коригування налаштувань детектора та компенсації флуоресценції. Програмне забезпечення отримує дані, а потім

КОПІЯ 2023

відображає їх, після чого відбувається аналіз. Файл даних містить дані з посиланнями або посиланнями на проаналізовані файли, тому документ експерименту містить усі файли для графіка, регіонів, воріт, статистики квадрантів, маркерів і анотованого тексту, але не файли даних. Програмне забезпечення надає кілька варіантів графіків, як-от розкид, контур і гістограма. Користувач повинен створити маркери гістограми для генерації статистики у визначеному розділі діаграми.

CellQuest Pro залишається мінімалістичним інструментом, який може виконати дуже простий рівень аналізу, але його достатньо, щоб зробити висновки. Обмеження також поширюється на такі функції, як сумісність імпорту та експорту зовнішніх типів даних і створення зображень якості публікації.

Програмне забезпечення BD Biosciences CellQuest, як приклад, було використано для аналізу даних проточної цитометрії для дослідження ролі аутофагії в імуногенності протипухлинної хіміотерапії у мишей [100], функції віперину в інфекційному процесі [101], ролі BID, BIM і PUMA в активації BAX- і BAK-залежної програми загибелі клітин [102].

FACSDivaa

FACSDiva, продукт BD Biosciences, який підтримує аналізатори та сортувальники проточної цитометрії, являє собою набір інструментів для налаштування додатків, збору та аналізу даних для спрощення звичайного робочого процесу в лабораторіях. Вони підтримують кілька цитометрів, але обмежуються цитометрами від BD, включаючи 140 комбінацій лазерів, дзеркал і фільтрів, стандартні типи стробування, як-от інтервал, прямокутник, ієрархічна прив’язка, квадранти та шарнірні квадранти, а також логічні стробування, як-от And, Or, Not, Rest Of. Програмне забезпечення може створювати експерименти, планшети та критерії отримання з програмного забезпечення, відмінного від BD. Аналіз даних можна виконувати вручну або пакетно, використовуючи як стандартні, так і надійні статистичні інструменти, а також створювати графіки, ворота, ієрархії населення та статистичні представлення на глобальному аркуші.

КОПЗ_2023

Більша увага приділяється підтримці та досягненню узгодженості якості даних у програмному забезпеченні. Функція налаштування та відстеження цитометра (CS&T) BD FACSDiva допомагає встановити базову лінію та налаштувати варіабельність приладу. Програмне забезпечення допомагає зменшити кількість помилок оператора, встановлюючи системну затримку, забезпечуючи тим самим послідовність результатів. Можливості відстеження контролю якості вимірюють налаштування приладу та надають звіт про продуктивність. Графіки Леві-Дженнінгса допомагають користувачам зрозуміти продуктивність приладу та визначити проблеми з обслуговуванням.

Програмне забезпечення BD Biosystem FACSdiva згадувалося в дослідженнях щодо контрольної точки G2/M [103], кровоносних судин [12], гематоенцефалічного бар'єру [104], ендосимбіонтів комах [105], антитіл, що нейтралізують ВІЛ [106], травм спинного мозку [107].], HSC людини з тривалим приживленням [108], віруси грипу [109] та регресія пухлини [110].

Програмне забезпечення для аналізу гелевих і блот-зображень

Програмне забезпечення для гелевих і блот-зображень зазвичай постачається разом із інструментами для візуалізації, за винятком ImageJ. Незалежне програмне забезпечення включає «Програмне забезпечення для денситометричного аналізу зображень», яке оцінює нелінійність авторадіографічних плівок під час калібрування, а також розраховує константи зв'язування білок-ДНК за допомогою аналізів електрофоретичної рухливості [111].

Інфрачервона система візуалізації LI-COR Biosciences Odyssey

Інфрачервона система візуалізації LI-COR Biosciences Odyssey є популярним вибором для аналізу білкового гелю та зображень Вестерн-блот, наприклад, [86, 112].

Інші програми

ImageJ

ImageJ – це програмне забезпечення з відкритим вихідним кодом, яке

КПЗ-2023

постійно розвивається та вдосконалюється за допомогою NIH. Він є у вільному доступі на звичайних обчислювальних платформах, таких як Windows, Linux і Mac, і має зростаючу та привабливу спільноту користувачів. Він розроблений з відкритою архітектурою, базується на мові Java, пропонує численні плагіни та дозволяє створювати налаштовані сценарії (макроси). Його можна автоматизувати за допомогою макросів із тих, які закривають усі відкриті вікна, копіюють дані в буфер обміну, створюють 3D-спектри, загальний багатоколірний аналіз спільної локалізації для отримання об'ємної частки кістки, фіксують послідовність сповільнених зображень у вигляді стека та як днів прогресує все більше і більше різноманітних плагінів, які відповідають вимогам спільноти користувачів. Програмне забезпечення дуже просте, і його можна використовувати з будь-яким набором зображень, взятих для аналізу. Хоча окремого модуля як такого немає, доступні простіші інструменти можна класифікувати для керування програмним забезпеченням для користувачів і розробників. Під час отримання зображень була доступна підтримка майже всіх платформ для отримання вхідних зображень. Набір аналізу складається з багатьох методів аналізу зображень як простих, так і складних. Пул доступних фільтрів, як-от фільтри ШПФ, які відфільтровують великі структури (корекція затінення) і малі структури (згладжування) заданого розміру за допомогою фільтрації Гауса в просторі Фур'є до віднімання фону, що котиться, що дає кращі результати, ніж новий код ImageJ для приблизно 16 -розрядні зображення. Як видно з цього фільтра, недолік якого полягає у створенні артефактів для багатьох зображень, якщо радіус кулі >=10, існує властива відсутність повноти при використанні одного або наступних методів.

Недоліки ImageJ – це його власні переваги простоти. Власне програмне забезпечення могло б усунути прогалину одного методу шляхом послідовного підключення іншого доповнюючого методу, щоб мінімізувати проблеми в аналізі даних, якщо не повністю викорінити. Таким чином, ImageJ буде кращим і чіткішим інструментом для користувачів, які розуміють метод, що

КПЗ-2023

використовується, на відміну від тих, які є у пропрієтарному програмному забезпеченні, де його можна легко натиснути та зберегти тип вбудованих обчислень. Документація відіграє важливу роль у створенні плагіна ImageJ, і її реалізація залежить від бажання кожного розробника.

ImageJ та його пов'язану версію Fiji [89, 117, 118] можна використовувати для кількісного визначення та обробки зображень, отриманих, наприклад, із смуг Вестерн-блот [119], імуноцитохімічних експериментів [12, 23, 120-125], кальцієвих живих клітин зображення [119] та гістологічні дані [98]. Antón-Bolaños N та інші використовували ImageJ для аналізу даних відстеження аксонів і даних кальцієвих флуоресцентних датчиків [126].

Icy – це відкрита платформа спільноти для інформатики біозображень, яка надає програмні ресурси для візуалізації, анотації та кількісної оцінки даних біозображень. Вихідний код доступний безпосередньо та надається під час кожного завантаження програми. Він розроблений в Інституті Пастера. Як і ImageJ, він пропонує численні плагіни та дозволяє створювати налаштовані сценарії (макроси).

CellProfiler

CellProfiler – це безкоштовне програмне забезпечення з відкритим кодом для кількісного аналізу біологічних зображень. Його розробка почалася в 2004 році в Broad Institute [127]. Немає попереднього досвіду програмування або комп'ютерного зору, оскільки він простий у використанні. Це дозволяє автоматично кількісно вимірювати фенотипи з тисяч зображень.

Imaris

Imaris (Bitplane) – це дуже інтуїтивно зрозуміле та потужне програмне забезпечення для візуалізації, аналізу, сегментації та інтерпретації наборів даних 3D та 4D мікроскопії. Окрім потужного режиму 3D-візуалізації, який ми бачили вище в тій статті, Imaris включає модулі для аналізу структур ниток, колокалізації, відстеження частинок, вимірювання та статистики, побудови даних, інтерпретації та видобутку, а також пропонує можливість розробки

КБПЗ_2023

спеціальних розширень. в MATLAB та інших мовах програмування.

Аміра

Amira – це програмне забезпечення для візуалізації та аналізу наборів 3D/4D даних. Він розроблений Thermo Scientific. Він використовується для різних методів зображення, включаючи КТ, МРТ і 3D-світлову або електронну мікроскопію.

Гюйгенс Професіонал

Huygens Professional – це пакет програмного забезпечення для обробки зображень, який є частиною програмного забезпечення Huygens Software і підходить для усунення шумів і деконволюції зображень. Він містить потужні алгоритми, сумісні з GPU, для швидкого аналізу зображень. Він містить деякі плагіни аналізу, наприклад аналізатор об'єктів, аналізатор колокалізації або відстеження об'єктів.

Іластик

Ilastik – це простий, безкоштовний, зручний інструмент із відкритим кодом для інтерактивної класифікації, сегментації та аналізу зображень. Його створено як модульну програмну структуру, яка наразі має робочі процеси для автоматизованої класифікації рівня пікселів і об'єктів, автоматичного та напівавтоматичного відстеження об'єктів, напівавтоматизованої сегментації та підрахунку об'єктів без виявлення. Більшість операцій є інтерактивними, навіть на великих наборах даних, потрібно просто намалювати мітки й одразу побачити результат. Він розроблений в EMBL [128].

Adobe Photoshop

Adobe Photoshop є єдиним ненауковим програмним забезпеченням, але належить до експертної сфери мистецтва та дизайну, у всьому наборі програмного забезпечення, розглянутого тут. Інструмент використовується науковим співтовариством через його легкість у використанні, високу якість обробки зображень і формат файлу, здатність передавати стандарти публікації та редагування для якісних цілей. Наприклад, Szőnyi A та інші підрахували кількість

КОПЗ-2023

позитивних клітин на конфокальних мікроскопічних зображеннях зрізів мозку після імуногістохімічного фарбування [117].

Недоліком є ненауковий характер, оскільки він доповнює функції покращення художньої якості, а не надає інструменти збору чи аналізу даних.

Програмне забезпечення розробляється для іншої цільової групи. Його використовували для покращення якості зображень для наукових досліджень із зауваженням, що воно займає перше місце в таблиці аналізу зображень, випереджаючи більш наукове та стандартне програмне забезпечення, що підтримує його популярність серед спільнота використовувати Photoshop для наукової комунікації та досліджень.

Статті в розглянутому наборі даних все ще можуть продемонструвати свою корисність у таких дуже різноманітних сферах, як стовбурові клітини, біологія розвитку нейронів мозку.

Matlab

Matlab (MAathWorks) – мова високого рівня та інтерактивне середовище для чисельних обчислень, візуалізації та програмування. Мова, інструменти та вбудовані математичні функції дозволяють користувачеві досліджувати різні підходи та знаходити рішення швидше, ніж за допомогою електронних таблиць або традиційних мов програмування, таких як C/C++ або Java.

Хоча в Matlab є модуль або панель інструментів, пов’язана з біологією, Computational Biology Toolbox, що цікаво, жодна цитата в цьому огляді не цитувала використання цього конкретного інструментарію.

У сфері аналізу математичних даних Matlab пропонує один із найбільш вичерпних списків можливостей перегляду даних. Чотири основні області його можливостей

1. Числове обчислення, де можна використовувати вбудовані математичні функції для вирішення наукових проблем.

2. Аналіз і візуалізація даних, за допомогою яких дані в кінцевому форматі можуть бути отримані для висновків і представлені у вигляді багатовимірних

КОПЗ-2023

графіків і різних форм графіків залежно від засобів опису та вираження інтерпретації. Тут дані можуть бути отримані шляхом взаємодії з апаратним забезпеченням різноманітної природи за допомогою відповідних доповнень, таким чином одержуючи необроблені дані, зображення, відео для аналізу та візуалізації.

3. Програмування та розробка алгоритмів дозволяє писати коди рідною мовою Matlab або C, C++, Java та.NET, таким чином інтегруючи програми Matlab з іншими мовами. Це допомагає використовувати весь потенціал Matlab для виконання матричних і векторних операцій для вирішення наукових проблем.

4. Як крок у майбутнє, додатки на базі Matlab можуть бути написані в сценаріях і ділитися як код, виконувані файли або програмні інструменти, таким чином дозволяючи власникам, які не є власниками Matlab, використовувати потенціал коду. Середовище розробки графічного інтерфейсу користувача можна використовувати, щоб змусити користувачів використовувати весь потенціал функцій Matlab простим клацанням миші.

Хоча Computational ToolBox може допомогти в таких сферах, як секвенування наступного покоління (NGS), аналіз мікроматриць, мас-спектрометрія та дослідження генної онтології, а програмне забезпечення SimuLink, що супроводжує основні допоміжні засоби програмного забезпечення Matlab у дослідженнях системної біології, обидва є ексклюзивним програмним забезпеченням, яке може використовувати MatLab кодування та база алгоритмів для підвищення якості даних.

Біологи все більше страждають від використання обчислювальних інструментів, таких як Matlab, оскільки вони також можуть писати код залежно від своїх потреб. Отже, Matlab був благом для тих дослідників, які мали можливість використовувати електронні інструменти поряд з комп'ютерами та отримувати багато цінних інтерпретацій аналізу необроблених даних за допомогою Matlab і подібних програм. Отже, знання основ математики та деякі навички кодування є передумовою для використання та розширення практичних

КПЗ-2023

можливостей Matlab у біологічних дослідженнях. Наприклад, Antón-Bolaños N та інші використовують програми Matlab для аналізу позаклітинних електричних записів *In vivo та даних флюоресценції таламічного кальцію, а також для проведення статистичного аналізу [*126]. Szőnyi A та ін. також використовували спеціально написані функції та сценарії в середовищі MATLAB для аналізу електрофізіологічних записів in vivo у мишей, що ведуть вільну поведінку, щоб вивчити роль ГАМКергічних клітин інцертного ядра стовбура мозку у формуванні контекстної пам'яті [117].

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

КПЗ-2023

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

КПЗ_2023

– 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,

КОПЗ-2023

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.

КПЗ_2023

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.

КПЗ-2023

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.

КбПЗ-2023

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 моніторингу біомедичної інформації.

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;

КПЗ-2023

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.

КбПЗ_2023

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Проблема опису живого організму в цілому далека від свого рішення внаслідок великого різноманіття й складності взаємозалежних процесів, що протікають у ньому, недоліку апріорної інформації про умови існування й властивостях досліджуваних біосистем [1, 2]. У рамках даної проблеми аналіз медико-біологічних даних, розробка математичних моделей і чисельних алгоритмів оцінки параметрів стану біосистем являють собою напрямок, що активно розвивається, застосування методів математичного моделювання й інформаційних технологій у біомедицинських дослідженнях [2, 3].

Основна методологічна проблема оцінки параметрів стану біосистем полягає в різнорідності й фрагментарності первинних біомедицинських даних і, як наслідок, обмеженості їхнього змістовного аналізу традиційними статистичними методами. Наслідком різнорідності вихідних даних є вимога наявності досить більших обсягів багатомірних вибіркових даних для одержання стійких і надійних інтегральних оцінок параметрів стану біосистем, заснованих на спільному аналізі комплексу обмірюваних показників. Крім того, особливістю біооб’єктів є те, що параметри, що характеризують їхній стан, як правило, недоступні для безпосередніх вимірів і оцінюються на основі аналізу «непрямих» експериментальних даних.

Збір, обробка й автоматизований аналіз фізіологічної інформації людини є найважливішою складовою частиною багатьох діагностичних методів сучасної медицини. Комп'ютерні системи збору й обробки електрофізіологічних сигналів є складними апаратно-програмними комплексами, що складаються з безлічі програмних компонентів, що виконують функції реєстрації біомедичної інформації (БМІ), її обробки й системного аналізу, а також діагностичної й

КПЗ-2023

сервісної операції. Основним підходом до проектування подібних комплексних систем довгий час була реалізація монолітної програмної архітектури із заздалегідь заданою функціональністю, забезпечуваною жорстко зв'язаними один з одним програмними компонентами. Функціональні властивості подібних систем практично неможливо було розширити, тому що вони були здатні виконувати лише ті функції, які були закладені на етапі проектування. Однак сучасні вимоги, запропоновані до даних систем, значною мірою, пов'язані з можливостями постійного розширення й нарощування їхніх функціональних властивостей. Важливими проблемами є також універсалізація біомедичного програмного забезпечення, під яким, насамперед, розуміється проблема повторного використання коду, і подолання наявних перешкод на шляху інтеграції різнорідних комп'ютерних біомедичних систем (БМС).

За останнє десятиліття досягнуть значний прогрес в області проектування складних програмних систем, що у корені змінив підхід до їхньої розробки й моделювання. Однак відсутність відповідних стандартів не дозволяє повністю скористатися перевагою нового підходу. Успіхи процесу стандартизації, у значній мірі, складаються в розробці документів рекомендаційного характеру, що регламентують інфраструктуру нижчої ланки – протоколів обміну, форматів файлів даних, медичних записів і повідомлень, а також концептуальні моделі взаємодії систем. Поза розглядом залишається, так зване, проміжне програмне забезпечення (ПЗ), під яким розуміється певний функціонально закінчений набір програмних засобів, інтегрованих у рамках обраної операційної системи (ОС), що забезпечує прозору роботу програм у неоднорідному середовищі. Неоднорідними середовищами, з погляду інформатики, є системи (локальні або глобальні), що складаються з компонентів, не сумісних один з одним з погляду програмного оточення.

Таким чином, для реального забезпечення взаємодії різнорідних програмних і апаратних систем необхідний вироблення єдиних специфікацій програмних інтерфейсів ПЗ проміжної ланки. У цей момент ця робота має

КОПЗ-2023

високий пріоритет у провідних світових установ по стандартизації, хоча й далека до завершення.

Із цього погляду актуальним є дослідження загальних властивостей комп'ютерних біомедичних систем і розробка з урахуванням вимог сучасних стандартів єдиних інформаційних моделей їхнього функціонування. Реалізація на цій основі універсальної об’єктно-орієнтованої інфраструктури, під якою розуміється безліч програмних компонентів і інтерфейсів із чіткою регламентацією можливостей їхнього використання, дає можливість перейти від монолітної програмної архітектури до компонентно-компонентно-орієнтованої розподіленого, вирішити проблеми універсалізації програмного забезпечення (ПЗ) і значно підвищити економічну ефективність розробок. Істотна різнорідність і різна функціональна спрямованість біомедичних систем робить завдання в загальному значенні практично нездійсненним. Однак існує відносно широкий спектр програмних біомедичних систем, для яких подібне завдання може бути успішно вирішене й, насамперед, для комп'ютерних біомедичних систем збору й математичної обробки електрофізіологічної інформації.

У цілому, у медико-біологічних дослідженнях складається досить суперечлива ситуація. З одного боку, накопичені різноманітні масиви даних, що відбивають усілякі, що зустрічаються в клінічній практиці ситуації [1, 4], а з іншого боку – непропорційно мала кількість інформації, одержувана з їхнього аналізу. У закордонній літературі зустрічається спеціальний термін, що характеризує подібну ситуацію – DRIP-синдром (Data-Rich, Information-Poor – багато даних, мало інформації). Це пов'язане з тим, що, незважаючи на очевидні успіхи, використання математичних методів і обчислювальної техніки в ряді випадків виявляється недостатньо ефективним з погляду прикладних цілей дослідження: спроби точного опису приводять до надзвичайно складного для аналізу математичним моделям, а недостатні обсяги даних не дозволяють проводити адекватні реальним процесам обчислювальні експерименти. Як результат, при гарних теоретичних побудовах практичне застосування

КОПЗ-2023

математичних моделей і алгоритмів для кількісної оцінки параметрів стану біосистем приводить до широкого розкиду у величині й надійності одержуваних оцінок.

У цей час для дослідження біосистем широко застосовуються методи статистичного аналізу даних [1, 2, 6, 8]. Основна проблема оцінки стану біосистем на їхній основі полягає в різнорідності й фрагментарності масивів первинних біомедицинських даних і, як наслідок, обмеженості їхнього змістовного аналізу традиційними статистичними методами. Зокрема, наслідком такої різнорідності є вимога наявності досить більших обсягів багатомірних вибіркових даних для одержання стійких і надійних інтегральних оцінок стану біосистем.

Інші методи одержання узагальнених оцінок стану біосистем засновані на, так званих, «модельно-незалежних» підходах: побудові напівемпіричних індексів стану, таких як:

- біохімічні, клінічний індекси [5];
- аналізі вербальних даних на основі теорії нечітких множин [2];
- методах багатомірного шкалювання [2];
- нейромережних технологіях [3, 7];
- методах мета-аналізу даних [3, 8] і ряді інших.

Така розмаїтість використовуваних методів ясно показує, що проблема одержання оцінок стану біосистем далека від свого рішення. Об'єктивне існування загальних системних закономірностей функціонування біосистем обумовлює необхідність комплексного підходу до розробки математичних моделей і алгоритмів оцінки параметрів їхнього стану [1, 2, 5], причому ефективність використання всієї сукупності різнорідних біомедицинських даних істотно визначається можливостями їхнього узгодження, тобто розробки концепції їхнього спільного використання для одержання більше надійних оцінок параметрів стану біосистеми або нової інформації про її властивості.

КОПЗ-2023

З методологічної точки зору кількісною оцінкою стану біосистеми є інтегральні характеристики, що розраховуються по сукупності багатомірних даних, що характеризують її стосовно деякого референтного стану, у якості якого найбільше часто використовується функціональний стан біосистеми який представляє норму або, у випадку діагностики захворювань, – стан здоров'я. Якщо властивості системи повністю відомі, то якість одержуваних інтегральних характеристик може бути досліджене аналітично. В умовах недоліку апріорної інформації про властивості біосистеми більше адекватними будуть оцінки, отримані методами статистичного моделювання.

Таким чином, розробка концепції узгодження біомедицинських даних, математичних моделей і методів оцінки параметрів стану біосистем є актуальною, як з погляду рішення фундаментальних і прикладних проблем в області медицини й біології, так і додатка методів математичного моделювання до дослідження біологічних об'єктів. Важливим складовим елементом у реалізації даного підходу є створення комплексу проблемно-орієнтованих комп'ютерних програм, що забезпечують як можливість нагромадження результатів досліджень у вигляді деякої інтегрованої системи даних, так і проведення кількісних оцінок стану біосистем різного рівня структурно-функціональної організації.

По фізичній природі джерела реєструємі біомедичні сигнали можна розділити на кілька груп [3]:

– біоелектричні – пов'язані з активністю нервових і м'язових кліток;

– імпедансні – складаються в зміні імпедансу тканин при пропущенні змінних струмів, із частотами до 1 Мгц;

– біоакустичні – створюються різними фізіологічними процесами людини: подих, рух і т.д.;

– біомагнітні – пов'язані з випромінюванням людиною надзвичайно слабких магнітних полів;

– біомеханічні – включають всі види механічних функцій людини;

КОПЗ-2023

– біохімічні – є результатом вимірів хімічних процесів, що відбуваються в тканинах;

– оптичні – пов'язані з реєстрацією електромагнітних випромінювань в оптичній області спектра, обумовлених функціональною діяльністю біологічних систем або наведених ззовні.

3.2 Розробка структурної схеми

На основі теоретичних результатів, представлених у другому розділі, розроблена інтегрована компонентно-орієнтована програмна система реєстрації й математичної обробки електрофізіологічних сигналів реального часу. Ядром програмної системи виступає архітектура динамічних модулів, що дає можливість реалізувати програмний комплекс повністю на компонентній основі. Розроблений комплекс, побудований на базі персональної ЕОМ під керуванням ОС Windows, може бути інтегрований практично з будь-якою апаратною реалізацією пристроїв реєстрації електрофізіологічних сигналів, як за допомогою модулів експорту/імпорту в стандартні формати зберігання БМІ, так і за допомогою модулів, що підключаються динамічно, прямого уведення ЕКГ, розроблених для конкретного пристрою реєстрації.

Серед технічних особливостей розробленого ПЗ необхідно виділити наступні [6]:

– динамічне підключення пристроїв реєстрації електрофізіологічних даних;

– підтримка можливостей реєстрації одночасно з декількох пристроїв;

– наявність комунікаційних інтерфейсів для передачі біомедичних записів через Internet і засоби мобільного зв'язку;

– доступність програмного забезпечення у двох конфігураціях: для портативної ПЕОМ і виконанні, що вбудовується, для побудови БМС під керуванням Windows-сумісних ОС.

КПЗ-2023

Основними завданнями, розв'язуваними апаратно-програмним комплексом, є:

1. Організація бази даних з можливістю нагромадження як електрокардіографічних, так і інших клінічних даних, при цьому уведення даних може здійснюватися як із пристроїв реєстрації, а також із зовнішніх файлів, створених іншими додатками.

2. Обробка цифрових сигналів різними математичними методами й збереження результатів обробки в інтегрованої БД:

– попередня обробка сигналу (лінійна, адаптивна КІХ-, БІХ-фільтрація перешкод);

– реалізація ряду сучасних кардіографічних методів (виділення й аналіз елементів ЕКГ);

– забезпечення можливості швидкої інтеграції модулів, що реалізують нові методи подання або обробки даних на основі розробленого програмного інструментарію в рамках компонентної архітектури системи [6];

– підготовка й подання даних у необхідному виді, у тому числі для друку або публікації;

– доступ до віддалених медичних інформаційних ресурсів [5].

3. Підтримка функцій обміну (у тому числі в реальному масштабі часу) електрокардіографічними й клінічними даними з іншими подібними системами на підставі встановлених стандартів.

Розроблене програмне забезпечення, будучи програмною частиною ЕКГ-системи реального часу, забезпечує переносний апаратно-програмний комплекс, що забезпечує основний набір функцій комплексу функціональної діагностики, підтримує, з погляду цільового застосування, наступні операції:

– реєстрація ЕКГ і розрахунок по одній з наступних схем відведень: 12 стандартних, по Небу, по Франку, біполярні ортогональні, Мак Фі [6];

– сполучення із пристроями, що реєструють допоміжну біомедичную інформацію (монітор артеріального тиску, пульсовий оксиметр і т.д.);

КПЗ_2023

– відображення на екрані й друк від 1 до 12 відведень сигналів із частотами дискретизації до 1 кГц і розрядністю 22 біта при швидкостях розгорнення 1.25, 2.5, 5, 10 см/сек з різними масштабами по амплітуді;

– підтримка функцій попередньої обробки, що налаштовуються, ЕКГ реального часу для придушення зовнішніх і інструментальних перешкод ЕКГ і артефактів;

– можливість динамічного підключення нових компонентів, що реалізують додаткові функції обробки, аналізу й візуалізації даних;

– забезпечення на додаток до стандартного режиму запису ЕКГ режиму досліджень, що дозволяє переглядати послідовність амплітуд і довжин певних QRS-комплексів;

– гнучке керування процесом друку; кардіограма, виведена на будь-який матричний, струминний або лазерний принтер (до 1200 точок/дюйм включно) відповідає прийнятим у практиці масштабам (від 1.25 до 10 см/сек і від 0.5 до 4 мВ/см);

– підтримка режиму запису, обробки й візуалізації тривалих ЕКГ, обсяг даних обмежується вільним простором, виділеним на жорсткому диску;

– підвищена надійність системи в режимі запису ЕКГ за рахунок реалізації механізму безпосереднього збереження записаної інформації на жорсткий диск у реальному часі;

– широка область застосування (клінічне, лабораторне, експериментальне).

До складу системи можуть входити: пристрій реєстрації ЕКГ, монітор артеріального тиску, пульсовий оксиметр, портативна ЕОМ, генератор тестових ЕКГ-сигналів, **а** також інші пристрої реєстрації електрофізіологічних сигналів.

Конфігурація модулів, що підключаються, залежить від конкретних початкових установок. У цьому випадку, у контексті реєстрації ЕКГ активізовані компоненти відображення ЕКГ, попередньої обробки, інтегрованої БД, а також модуль візуального керування пристроєм реєстрації ЕКГ.

КПТВ 2023

Конфігурація розробленого ПЗ у вбудованому виконанні, побудована на базі промислової одноплатної ЕОМ, представляючої у якості інтерфейсу керування 7 функціональних клавіш розташованих по периметрі РК-екрана й маніпулятор. Програмною платформою даної ЕОМ може бути довільна Windows-сумісна ОС, у тому числі є можливої адаптація й під платформу Microsoft Pocket PC 2002, що є розвитком що вбудовуються ОС лінії Windows CЄ.

Функціональні призначення клавіш керування відображаються на екрані в конкретний момент роботи програми. Панелі керуючих клавіш також реалізовані у вигляді модулів, що підключаються, для уніфікації процедури керування екранними видами.

Структура програмного забезпечення розробленого ПЗ у вбудованому виконанні, повністю відповідає повноцінній версії. Версія розробленого ПЗ, що вбудовується, функціонує у двох базових режимах: у режимі знімання й у режимі аналізу ЕКГ. У режимі знімання відбувається відображення в реальному часі кардіограми одержуваної від пацієнта. Останні 15 сек. відображуваної кардіограми при переході в режим аналізу використовується для аналізу ЕКГ. Модуль аналізу ЕКГ робить виміри характеристик певних QRS-комплексів і порівнює значення з нормативами, що відповідають, підлоги, віку, вазі й росту пацієнта.

Знята кардіограма й результати аналізу зберігаються на локальному диску й карті пам'яті, що підключається. Ці дані також можуть бути переспрямовані на наступні пристрої передачі даних: інтерфейс IrDa, модем (у тому числі для супутникової й стільникової телефонії), стандартний мережний інтерфейс. розробленого ПЗ у дослідницькому й виконанні, що вбудовується, мають єдиний менеджер друку, що підтримує режим безперервного паперу, що забезпечує необхідні масштаби по амплітуді й часу подання електрокардіографічного й супутніх сигналів

З метою полегшення створення й модернізації нових програмних комплексів ми вирішили використовувати структурований системний підхід. Так,

КПЗ-2023

на рисунку 3.1 зображена структурна схема розробленої системи, що повинна стати основою для формування конкретних програмних комплексів, наприклад, реографа, електроенцефалографа, спірографа й т.д. У цю систему повинні ввійти загальне ядро цих комплексів, базові алгоритми обробки, система інтерфейсу й т.п. Переваги цього підходу очевидні, тому що у випадку модернізації якого або базового алгоритму немає необхідності міняти його у всіх комплексах, досить змінити його в базовій системі, і, автоматично, новий алгоритм працює у всіх комплексах, створених на основі цієї системи. Точно так само вносяться й всі нові алгоритми. У випадку переходу на нову версію операційної системи, досить обновити тільки базову систему. Прискорюються також і строки створення будь-якого нового комплексу. Новий комплекс має інтерфейс і систему керування подібну з попередніми комплексами, що відразу вирішує проблему сумісності й усуває необхідність переучування персоналу працюючого з комплексом.

Апаратна частина являє собою пристрій реєстрації й попередньої обробки сигналу й звичайно здійснює аналого-цифрове перетворення. Буферизація даних на виході є звичайним для систем реального часу з метою синхронізації або впакування переданих даних. Інтерфейсною частиною з боку програмної системи виступає системний додаток – драйвер, що забезпечує взаємодію з апаратною частиною на низькому рівні. Засоби візуалізації й друку забезпечують подання даних у зручному для аналізу форматі. Все більша вага здобуває підсистема комунікаційних інтерфейсів, тому вона в даному поданні виділена в окремий блок.

При проектуванні діагностичної системи, я базувався саме на принципах реалізації ідеї медичних інформаційних технологій «без границь».

КПЗ–2023

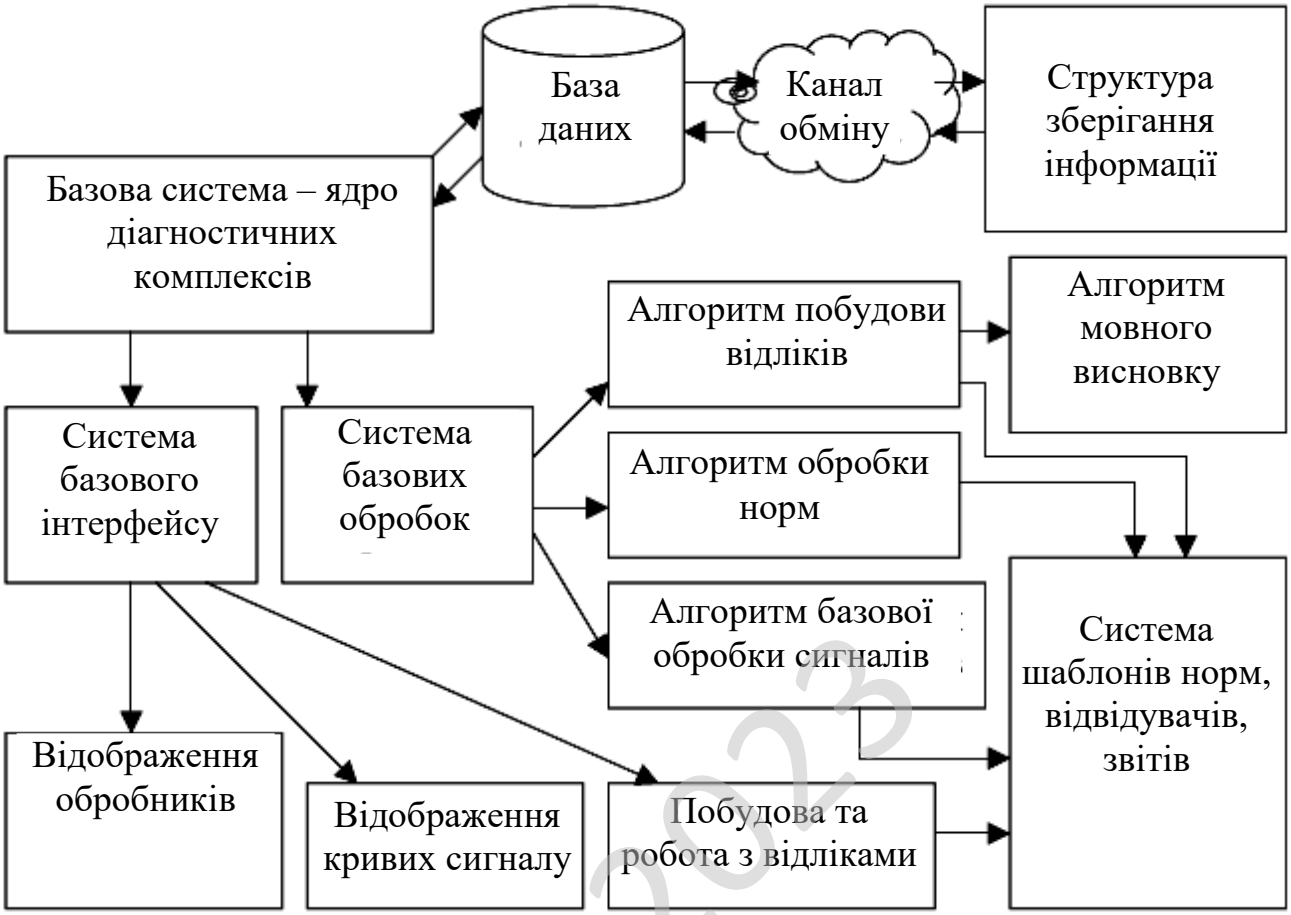


Рисунок 3.1 – Структурна схема системи

Доступ до діагностичної інформації, її безпека, можливість мережної обробки даних

Медико-діагностична система використовує стандартизовані рішення для зберігання й передачі діагностичної інформації. База даних, що зберігає всю інформацію про відвідування пацієнтів у строгому ієрархічному порядку, створена на файловій основі у вигляді чіткої ієрархічної структури каталогів (папок). Кожна папка, залежно від свого положення в ієрархічній структурі, зберігає файли з її описом, додатковою службовою інформацією й папки більш низького рангу.

Така структура є найпростішою й дозволяє організовувати легкий доступ до бази даних по мережі, без необхідності установки додаткових драйверів.

КОПЗ_2023

Розміщення такої бази на сервері установи дозволяє організовувати віддалений доступ до будь-якого її елемента. Така база даних легко стерпна на різні носії (для цього потрібно тільки файловий менеджер) і при цьому на одному носії можна зберігати кілька баз даних. Медико-діагностична система при роботі з відвідуванням створює його тимчасову копію на локальному носії й, при всіх змінах у процесі роботи, вносить змінені дані в цю локальну копію. При збереженні відвідування локальна копія переноситься в базу даних. Цей механізм разом з обробкою журналу транзакцій дозволяє автоматично відновлювати загублені дані у випадку яких або збоїв, наприклад, при вимиканні живлення комп'ютера.

Платформна переносимість

Для забезпечення платформної переносимості медична система створювалася під сімейство операційних систем Windows, тому що саме ці операційні системи найпоширеніші в області домашніх і робочих систем і мають тенденцію до платформної незалежності. Єдність API цих операційних систем дозволяє з малими витратами робити адаптацію медичної системи для використання з портативними комп'ютерами. Архітектура, що просувається в останній час корпорацією Microsoft.NET дозволяє створювати дійсно крос-платформні програмні продукти, однак має деякі недоліки, як, наприклад, істотні вимоги до пропускної здатності лінії зв'язку, внаслідок чого варто ретельно зважити виграш і накладні витрати, пов'язані з перекладом на цю архітектуру. З огляду на сучасні тенденції розвитку Internet в Україні, у цей час нам представляється поки недоцільним створювати медичні системи на основі цієї архітектури.

КОПЗ-2023

3.3 Розробка функціональної схеми

У рамках представленої архітектури системні компоненти умовно по своєму функціональному призначенню розділяються на три рівні: рівень даних, функціональної логіки й подання. Кожний із цих компонентів реалізує необхідні для даного рівня інтерфейси доступу до укладеної функціональності й користується надаваними системними сервісами в рамках, доступних для цього рівня. Рівень даних становлять об'єкти, відповідальні за операції вводу/виводу із зовнішніми джерелами даних (наприклад, пристрій реєстрації ЕКГ, монітор артеріального тиску), а також універсальний компонент зберігання гетерогенних, тобто різнорідних, даних, і, так звана, фабрика контейнерів даних – об'єктів, використовуваних для передачі різного роду інформації через інтерфейси компонентів.

Функціональна логіка реалізується набором об'єктів – менеджерів компонент, часто умовно називаних документами, що має методи обробки даних і засобу керування компонентами візуалізації. У свою чергу, об'єкти візуалізації можуть мати методи обробки даних, хоча їхнім безпосереднім завданням є відображення інформації й забезпечення інтерфейсу з користувачем. Завдяки принципу відкритої архітектури й наявності чіткої специфікації, можливі розширення й інтеграція в комплекс нових функціональних модулів, що реалізують нові методики аналізу або надають можливість здійснювати знімання даних на нових пристроях вводу ЕКГ.

Таким чином, структурно програмна система може розглядатися як ієрархічна архітектура окремих елементів, серед яких основними елементами є різноманітні компоненти, що виконують заздалегідь задані функції в рамках наступних підсистем:

- клінічних методів обробки даних, візуалізації й друку БМІ;
- зовнішніх комунікаційних інтерфейсів;
- реєстрації біомедичних даних;

КПЗ-2023

– нагромадження й зберігання БМІ.

Функціональна схема системи представлена на рисунку 3.2.

Компонентно-орієнтована архітектура додатка, побудована на декількох рівнях реалізації, вимагає наявності більш складної процедури первісної активації програмної системи. На відміну від монолітних додатків, дана процедура через наявність безлічі незалежних одиниць, що виконуються, може стати досить складної для реалізації. Для рішення даного завдання в рамках розроблена багаторівнева процедура ініціалізації й динамічного конфігурування окремих підсистем.

Початковою фазою процедури активації є завантаження й ініціалізація основних елементів ПЗ, зокрема, основного додатка, менеджера модулів, що підключаються, без яких нормальне функціонування системи неможливо. Визначення контексту використання дозволяє визначити подальшу процедуру ініціалізації додатка й список використовуваних у даному контексті підсистем. Так, наприклад, у випадку віддаленої активації з додатка Microsoft Word для редагування впровадженого елемента OLE, підсистема реєстрації й телекомунікаційних інтерфейсів не буде ініціалізована. Завершальним етапом ініціалізації системи є завантаження всіх модулів, що підключаються, які автоматично підключаються до системних сервісів, надаваним основним додатком і вже активізованими підсистемами.

Розглянемо більш докладно перераховані вище підсистеми.

Підсистема зовнішніх комунікаційних інтерфейсів

Одним з найбільш важливих елементів інтеграції різнорідних систем реєстрації біомедичних сигналів є виробіток загальних стандартів зберігання й подання біомедичних даних. Це є важливим кроком на шляху здешевлення, універсалізації й підвищення якості біомедичного програмного забезпечення на базі ЕОМ. Однак у рішенні проблем сумісності комп'ютерних електрокардіографів на рівні обміну даними за останні 30 років серйозних змін не відбулося. Великі фірми-виробники ЕКГ-приборів (Siemens, Hewlett Packard, Fukuda, NEC, Oxford, Del Mar і т.д.) розробили свої корпоративні стандарти, які

КПЗ_2023

практично всі є закритими, тобто недоступними для застосування в системах інших виробників.

Безсумнівним успіхом процесу стандартизації з'явилася розробка прийнятого медичного промисловістю стандарту DICOM/MEDICOM [5], який служить, в основному, для подання зображень у радіології. Стандартизація систем реєстрації й обробки фізіологічних сигналів, що почалася порівняно недавно й не підкріплена мотивацією великих виробників медичних систем, не змогла виробити єдиний формат. Кількість різноманітних специфікацій досить великий – кожний великий виробник медичного встаткування має у своєму розпорядженні свій власний стандарт. Разом з тим, у цей час існує ряд специфікацій і рекомендацій, прийнятих різними установами по стандартизації, деякі з яких є національними стандартами, тому досить складним завданням є оптимальний вибір між існуючими специфікаціями.

Аналіз кола завдань, розв'язуваних сучасними системами реєстрації й обробки сигналів, дозволяє сформулювати ряд вимог, пропонованих до форматів подання біомедичних даних, найбільш важливими з яких є наступні положення:

– універсальний формат подання даних, що підтримує можливість подання сигналів різного роду: кардіологічні, нейрофізіологічні, респіраторні й одночасне зберігання різнорідних сигналів, записаних з різною частотою дискретизації й розрядності відліків;

– підтримка різноманітних форм анотування вимірів: текстова, голосова, візуальна;

– забезпечення механізму зберігання другорядних не прив'язаних до оцінок часу даних, наприклад, результати аналізу записів, синдромальні висновки лікаря-фахівця;

– багаторівнева система сумісності, при якій не потрібна реалізація повної функціональності, декларованої в стандарті, для досягнення мінімального рівня інтеграції;

– підтримка фізичних маніпуляцій з даними, з можливістю їхнього здійснення в масштабі реального часу.

КПІ – 2023

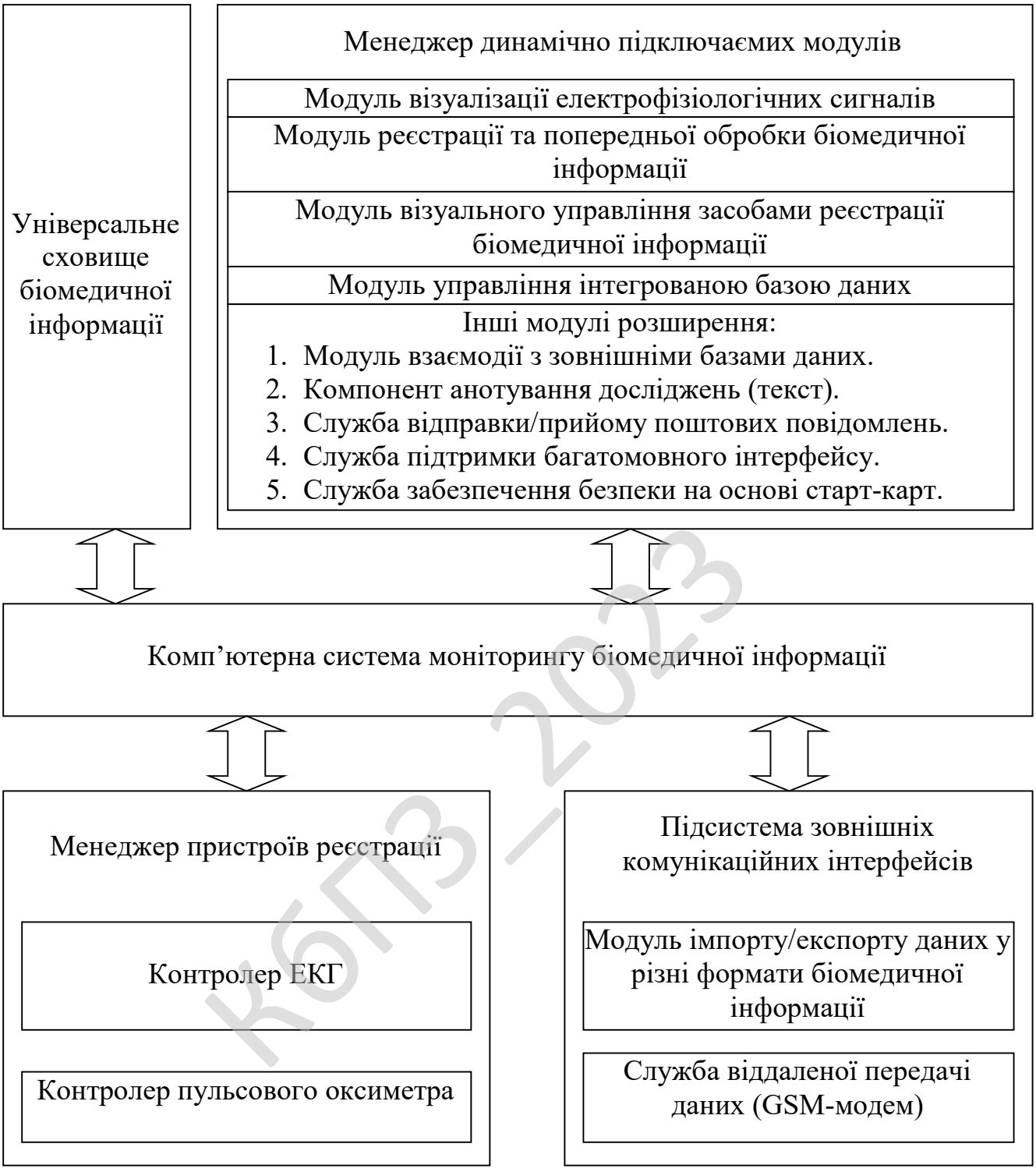


Рисунок 3.2 – Функціональна схема системи

Додатковими критеріями, використовуваними при аналізі безлічі існуючих стандартів, є:

КбПЗ_2023

– популярність стандарту в медичній промисловості, що оцінюється процентним співвідношенням систем, що підтримують даний формат;

– простота реалізації, що звичайно перебуває у зворотному співвідношенні до обсягу специфікації;

– компактність подання даних, наявність функцій стиску даних.

Існуючі системи специфікацій можна розділити в цілому на два класи. Перший клас становлять спеціальні стандарти, розроблені для вузької предметної області, у яких регламентації піддається сама структура даних. До другої групи варто віднести, так звані, метастандарти, у яких побудована інформаційна модель предметної області, у рамках якої деталізовані моделі кожного предметного об'єкта, визначена система зв'язків компонент предметних об'єктів, розроблені відповідна номенклатура й система кодування [6]. Під номенклатурою розуміється деякий список різноманітних типів даних, у поданні інформаційної моделі – об'єктів, що виникають у процесі медичного документообігу. У рамках таких метастандартів, будь-який формат подання даних є деякою логічною побудовою розробленої метамоделі.

Підсистема візуалізації й друку біомедичних даних

Основне призначення підсистеми візуалізації будь-якої системи реєстрації електрофізіологічної інформації реального часу складається у виконанні цілого ряду функцій, найбільш важливими з яких є:

– відображення, переданої з різних джерел реєстрації або отриманої в результаті застосування медичних методик інформації в найбільш зручному для сприйняття лікарем-фахівцем виді;

– своєчасна реакція на параметри, що змінилися, вхідних даних за допомогою графічної або звукової сигналізації.

У цьому випадку, час реакції залежить від ряду факторів [7]:

– від часу доставки даних, що залежить від операційної системи (ОС), інтерфейсних протоколів, що забезпечують передачу інформації від пристрою реєстрації до підсистеми візуалізації;

КОПЗ-2023

– від часу реакції медичного працівника, що експлуатує систему.

Таким чином підсистема візуалізації, що є основою будь-якого інтерфейсу користувача, виступає основним інтерфейсом між системою реєстрації й лікарем-фахівцем, що інтерпретує отримані дані з метою визначення діагностичного висновку. Тому якісне подання інформації є дуже важливим фактором якості діагностики.

Системи реєстрації й обробки сигналів реального часу мають два яскраво виражених режими візуалізації: режим реального часу й режим повторного перегляду. У цьому контексті друк розглядається як окремий випадок режиму перегляду з особливим масштабом.

Сучасні методи діагностики висувають ряд серйозних вимог до підсистеми візуалізації, з яких найбільш важливими й трудомісткими для реалізації є наступні:

– підтримка синхронного відображення декількох потоків даних, отриманих від різних джерел, що мають різні частоти дискретизації й спектральний состав;

– підтримка декількох масштабів відображення даних по осі амплітуди сигналу й часу;

– відображення допоміжної й похідної інформації, наприклад, оцінки міток, подій і т.д.

– підтримка декількох форм відображення даних.

У режимі перегляду БМІ додатковою вимогою є підтримка виміру амплітуд і тимчасових відрізків по двох і більш довільно встановлених точках. Таким чином, модуль візуалізації є компонентом, що обслуговує декілька багатоканальних потоків, причому обсяги переданої інформації через транспортне з'єднання можуть досягати значних розмірів. Так передача стандартної 12-канальної ЕКГ із частотою дискретизації 1 кГц із обліком 4-байтового подання відліків із плаваючою точкою становить порядку 375 кБод, причому кількість відліків, оброблюваних в одну секунду, становить 12000. Так

КОПЗ_2023

як розв'язна здатність сучасних стандартних засобів візуалізації становить усього порядку 100-200 точок/дюйм, то для ефективного рішення завдань візуалізації важливою проблемою є скорочення кількості відліків, які використовуються для відтворення безперервних сигналів при частотах дискретизації значно переважаюча розв'язна здатність із урахуванням обраного масштабу. Відповідно до вимог американської асоціації кардіологів [3] мінімальними частотами, що рекомендуються, дискретизаціями є 500 Гц, а для методик ЕКГ ВР – більше 1 кГц, тому при відображенні для більшості масштабів є значний ступінь надмірності вхідних даних. Надмірність полягає в тім, що різним відлікам у вихідному сигналі відповідає одна тимчасова позиція на пристрої візуалізації.

Підсистема зберігання й нагромадження біомедичних даних

Найважливішим завданням підсистеми зберігання й нагромадження даних є забезпечення надійного механізму короткострокового й довгострокового збереження біомедичних даних різного роду, а також підтримка ефективного доступу до них. Під короткостроковим збереженням даних розуміється часовий період, що не перевищує однієї сесії роботи програмного додатка.

ЕКГ записи навіть невеликих довжин займають досить більші обсяги. Найпростіший розрахунок показує, що навіть при реєстрації ЕКГ зі стандартною системою відведень, запис якої становить порядку 1 мінути при частоті дискретизації 1000 Гц і квантування АЦП 16 біт, обсяг даних досягає:

W= 12*1000*16 = 192000 біт/сек або порядку 1,4 МБ/хв.

Тому при зростанні довжин знімання ЕКГ до мінути й більше (наприклад, в електрокардіографії фізичного навантаження й холтеровському моніторингу) виникає гостра необхідність у стиску ЕКГ-сигналу. Такий стиск може стати необхідним і при менших обсягах ЕКГ у випадках, коли необхідне збереження більшої кількості записаних ЕКГ в інтегрованої БД або існує необхідність у передачі ЕКГ за допомогою телефонного або мобільного зв'язку, а також засобами мережі Інтернет.

КОПЗ_2023

З іншого боку, при записах ЕКГ великої тривалості існує ймовірність апаратного або програмного збою ЕОМ і втрати вже оброблених даних і, отже, необхідно короткострокове збереження БМІ, що дозволяє відновити загублені дані при необхідності, а також розвантажити оперативну пам'ять ЕОМ від зайвого навантаження. Тому важливість підсистеми нагромадження даних для ЕКГ-систем, побудованих на платформі ЕОМ, що забезпечує можливості стиску біомедичних даних і короткострокового зберігання (буферизації) досить висока.

У БМС сховище забезпечує набудовується на величини до декількох мінут буферизацію БМІ в рамках внутрісистемного буфера FIFO. При цьому зберігання даних здійснюється у внутрішнім поданні системи. Короткострокове зберігання даних вимагає ефективної організації самого сховища для забезпечення швидкого доступу до даних у режимі перегляду. Зниження обсягів інформації, що бере участь в обміні між дисковим сховищем і буфером FIFO можливо двома способами:

– зменшення надмірності даних за рахунок скорочення кількості біт для подання одного відліку;

– потокового стиску даних.

Зменшення бітності відліків з 32-х до 16 біт у дозволяє без значних обчислювальних витрат скоротити інтенсивність обміну з дисковим сховищем. Стиск даних на цьому етапі має сенс лише в тому випадку, коли скорочення обсягів переданої інформації дає виграш у продуктивності. Однак недоліки сучасних методів стиску даних не дозволяють використовувати їх у режимі реального часу й роль, що відводиться їм, скорочується до застосувань у реалізаціях функцій імпорту/експорту в різні файлові формати подання БМІ.

Цифрові записи електрофізіологічних сигналів, у тому числі й ЕКГ, являють собою послідовності звітів, тому звичайні методики компресії даних застосовні й до них. Застосування того або іншого методу стиску БМІ визначаться необхідними характеристиками методу (продуктивність, коефіцієнт стиску), а також його властивостями, зокрема, можливості працювати в потоці,

КБПЗ_2023

необхідними ресурсами. Твердим вимогам реального часу здатні задовольнити досить вузький клас методів стиску, здатних до роботи в потоці даних [4].

Підсистема запису даних із пристроїв реєстрації електрофізіологічних сигналів і розподілу даних

Сучасні підходи до проектування міжкомпонентної взаємодії в програмних системах передбачають декілька можливих схем, у тому числі мережну модель і модель клієнт/сервер.

Відповідно до сучасних концепцій про механізм взаємодії медичних систем, що здійснюють реєстрацію, обробку й збереження біомедичної інформації [9], модель такої взаємодії, що передбачає можливість динамічного встановлення зв'язку (принцип plug&play), будується за класичною схемою клієнт/сервер. Дана схема є також основою єдиного комунікаційного стандарту MIB IEEE 1073 взаємодії медичних пристроїв. Розвитком даної парадигми взаємодії об'єктів є модель так званих програмних агентів, які ми можемо визначити, з Вудриджу [7] як апаратну або програмну систему, що володіє рядом властивостей:

– автономність – агенти мають здатність самостійно виконувати поставлені завдання або змінювати своє внутрішньо стан без прямого втручання ззовні;

– соціальність – здатність до взаємодії з іншими подібними системами;

– реактивність – властивість агента сприймати своє оточення й відповідати в заданий проміжок часу на його зміну;

– проактивність – можливість приймати ініціативу по виконанню тієї або іншої дії на себе.

Таким чином, парадигма програмних агентів, що найбільше повно погодиться з основними принципами організації підсистеми обслуговування й підтримки пристроїв реєстрації різної електрофізіологічної інформації, може бути ефективно використана для реалізації підсистеми реєстрації даних [7].

КПЗ_2023

Біомедичная система, що виступає стосовно додатка менеджера як програмний агент, відповідно до моделі предметної області IEEE 1073 є формальним позначенням для довільної системи, що обслуговує один або декілька біомедичних пристроїв реєстрації БМІ. Програмною частиною, що забезпечує керування й доступ до ресурсів блока управління, є контролер біомедичного пристрою реєстрації сигналів. Комунікаційний протокол будується на основі стандартних мережних рівнів ISO/OSI і підтримується шаром Association Control Service Element (ACSE) [2], що забезпечує логічний зв'язок між взаємодіючими сторонами й рівнем Common Medical Device Information Service Element (CMDISE) [3], що дає доступ до внутрішньої об'єктної інфраструктури додатка-агента, регламентованою інформаційною базою медичних даних (ІБМД) (англ. MDIB) стандарту IEEE 1073. MDIB є формальним описом всіх існуючих у рамках логічних моделей біомедичних систем об'єктів. Дані об'єкти є елементарними моделюємими елементами з «реального миру» такими, як медична система (МС), віртуальний медичний пристрій (ВМП), масив відліків реального часу (МВРВ), масив відліків (МВ), подія.

У цілому, завдання реалізації підсистеми реєстрації й розподіли медичних даних із пристроїв знімання складаються у взаємно однозначному відображенні подань ІБМД.

Найбільш важливим компонентом представленої схеми є ВМП, що служить абстрактним поданням компонента, що обслуговує пристрій реєстрації БМІ, і компонент абстрактного доступу до устаткування, що забезпечує інтеграцію із ВМП. Принципи взаємодії ВМП побудовані на основі парадигми програмних агентів, як концепції яка дозволяє найбільше повно вирішити завдання децентралізованого керування пристроями. У рамках архітектури ВМП має наступні властивості:

– внутрішній стан ВМП однозначно відповідає стану фізичного пристрою, зміни цих станів строго детерміновані;

КПЗ-2023

– ВМП забезпечує функції керування й контролю станів керованого апаратного пристрою;

– ВМП управляє створенням, розподілом масивів відліків реального часу й передачею подій.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3.

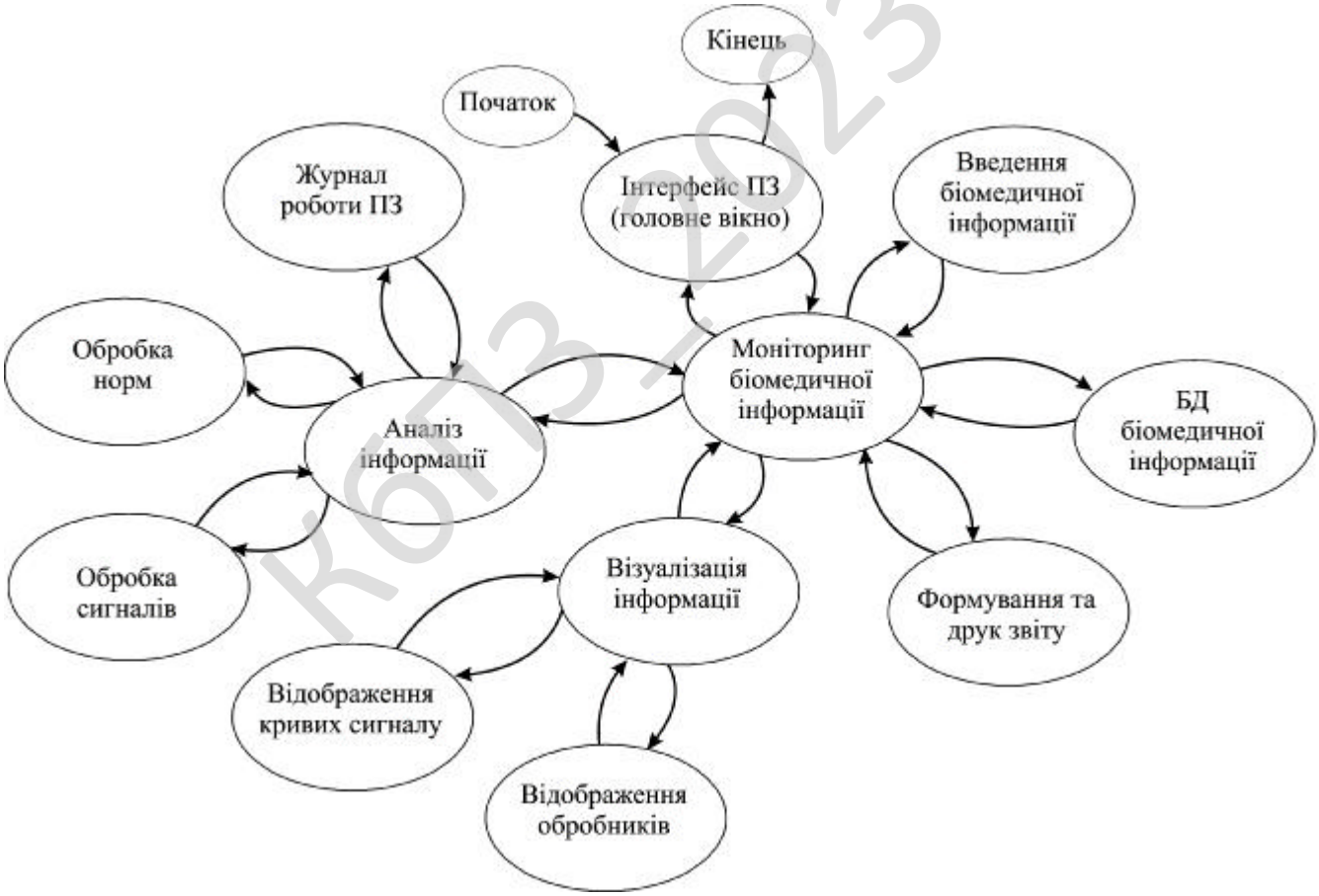


Рисунок 3.3 – Діаграма взаємодії процесів

КбПЗ-2023

Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ (головне вікно).

– Моніторинг біомедичної інформації.

– Введення біомедичної інформації.

– БД біомедичної інформації.

– Формування та друк звіту.

– Візуалізація інформації.

– Відображення обробників.

– Відображення кривих сигналу.

– Аналіз інформації.

– Обробка сигналів.

– Журнал роботи ПЗ.

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.

КбПЗ_2023

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація. UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір

КПЗ-2023

можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

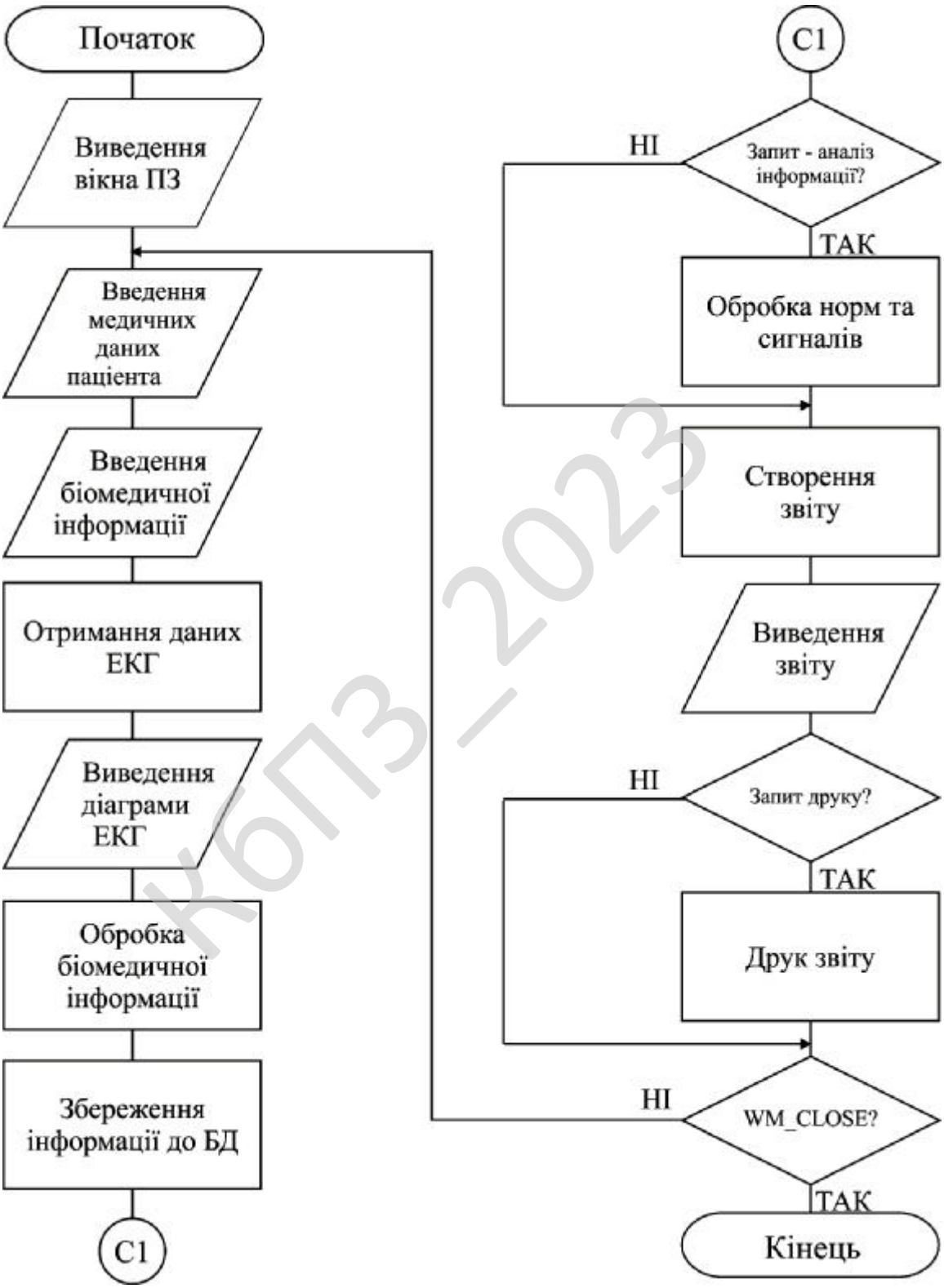


Рисунок 4.1 – Блок схема основної програми

КбПЗ_2023

КБПЗ_2023

Рисунок 4.2 – Блок схема підпрограми

Опис алгоритмів функціонування системи

Розглянемо частину програмного коду, у якому описане призначення процедур та змінних при реалізації створення ЕКГ.

```
unit EKG_V;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Signals, Math, Printers, Matrix;

const
  SNoSignals = 'Сигнали ЕКГ не виявлені!';

type
  TEKGFrame = class(TFrame)
    ScrollBox1: TScrollBox;
    Image: TImage;
    Panel1: TPanel;
    PrintDialog1: TPrintDialog;
    panLine1: TPanel;
    panLine2: TPanel;
    procedure panLine1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
  private
    FGrid: TBitmap;
// Кеш міліметрівки
    FGridColor: Byte;
// Колір сітки (за замовчуванням 50 – рудий)
    FPrintGridColor: TColor;
// Колір роздрукованою сітки
    FDinEKGColor: TColor;
// Цим кольором відображається ЕКГ у процесі знімання
    FEKGColor: TColor;
// Колір графіків
    FPeaksColor: TColor;
// Колір ліній, що виділяють піки ЕКГ

// Параметри вихідних сигналів
    FAmplitude, FFreq: Integer;
// Амплітуда й частота
// Параметри відображення
    Fmm_on_m, Fmm_on_sec: Integer;
```

КБПЗ_2023

```
// Число міліметрів на мілівольт  і міліметрів у секунду
    FInterval: Integer;
// Відстань між сусідніми кривими
    FGridVisible: Boolean;
// Визначає видимість сітки
    FWS: TWorkspace;
// постійна робоча область
    FNames: array of string;
// Імена графіків
    FHead, FFoot, FPageInfo: string;
// Текстова інформація, що виводиться на друк
    FProcessCnt: Integer;
// Лічильник  що використовується при зніманні сигналів ЕКГ
    FPixel: Integer;
// Зберігає номер останнього засвіченого пікселя
    FLimit: Integer;
// Ліміт що використовується при зніманні ЕКГ
    FCoef: Real;
// Коефіцієнт, показуючий, скільки точок проецирується на 1 піксель
    FSignCount: Integer;
// Число сигналів, показуємі у процесі знімання ЕКГ
    Fidx: Integer;
// Індексування масиву Matri
    FIsProcess: Boolean;
// Дорівнює True, якщо виконується процес знімання ЕКГ
    FPosLine1: Integer;
// Позиція 1 лінії
    FPosLine2: Integer;
// Позиція 2 лінії
    FLinesVisible: Boolean;
// Визначає, чи видні лінії
    FStepVisible: Boolean;
// Визначає видимість каліброваного імпульсу
    FList: TStringList;
// Зберігає текст повідомлення MessageText
    procedure SetFont();
// Установлює шрифт тексту
    procedure FindOptimalInterval();
// розраховує інтервал між кривими так щоб їх усі можна було роздрукувати
// мальовування кривих
    procedure _DrawImage(DrawHead: Boolean = True; DrawFoot: Boolean = True);
    procedure DrawGrid(AllCanvas: Boolean = True);
```

КОПЗ_2023

```
    procedure SetPosLine1(const Value: Integer);
    procedure SetPosLine2(const Value: Integer);
    procedure SetSignalsText(const Value: string);
    function GetSignalsText: string;
// Промальовування сітки
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    //---------ФУНКЦІЇ ДЛЯ ВІЗУАЛІЗАЦІЇ КРИВИХ ЕКГ--------
    // Завантаження сигналів ЕКГ із робочою області WS
    procedure LoadSignals(WS: TWorkspace; DataArray: string; PeaksAr: string =
      '');
    // Видаляє сигнали ЕКГ із робочою області. Викликає метод DrawImage().
    // Звичайно служить для виводу повідомлення про помилку  MessageText
    procedure ClearSignals();
    // Текст повідомлення, виведеного на міліметрівку. Текст виводиться тільки при
    // відсутності сигналів ЕКГ. Звичайно це повідомлення про помилку,  при якій
    // має сенс прекратити відображення кривих
    property MessageText: string read GetSignalsText write SetSignalsText;
    // Встановлює імена масивів
    proce dure SetNames(Names: array of string);
    // Промалювання графіків ЕКГ (і міліметрівки при необхідності)
    procedure DrawImage();
    // Печатка кривих на принтері
    procedure Print();
// Підготовляє сітку до відображення що знімається ЕКГ.
 // SignCount - кількість відведень
    procedure StartProcess(SignCount: Integer);
    // Процедура візуалізує процес знімання ЕКГ. Щораз у неї подається
    // масив знятих точок (по одній точці для кожного відведення)
    procedure AddPoints(const NewPoints: array of Real);
    // Процедура зупиняє процес знімання ЕКГ. Інші процедури
    // будуть недоступні, поки ви не зупините процес знімання ЕКГ
    procedure StopProcess();
    // Повертає True, якщо відбувається знімання ЕКГ
    property IsProcess: Boolean read FIsProcess;
    //----------- НАСТРОЮВАННЯ ПАРАМЕТРІВ-------------
// Установка параметрів
    procedure SetParams(Amplitude, Freq, mm_on_m, mm_on_sec: Integer);
// Колір сітки на екрані монітора
    property GridColor: Byte read FGridColor write FGridColor;
// Колір сітки при печатці
```

КБПЗ_2023

```
    property PrintGridColor: TColor read FPrintGridColor write FPrintGridColor;
  // Колір кривих
    property EKGColor: TColor read FEKGColor write FEKGColor;
  // Колір кривих у процесі знімання ЕКГ
    property DinEKGColor: TColor read FDinEKGColor write FDinEKGColor;
  // Колір ліній, що виділяють піки ЕКГ
    property PeaksColor: TColor read FPeaksColor write FPeaksColor;
// Інтервал між кривими
    property Interval: Integer read FInterval write FInterval;
// Включає або виключає міліметрівку
    property GridVisible: Boolean read FGridVisible write FGridVisible;
// Текст заголовка для друку
    property HeadText: string read FHead write FHead;
// Додаткова інформація для друку (унизу екрана)
    property FootText: string read FFoot write FFoot;
// Інформація про сторінку для друку
    property PageInfo: string read FPageInfo write FPageInfo;
// Позиція першою лінії
    property PosLine1: Integer read FPosLine1 write SetPosLine1;
  // Позиція другою лінії
    property PosLine2: Integer read FPosLine2 write SetPosLine2;
   // Визначає видимість лінії (споконвічно лінії не видні)
    property LinesVisible: Boolean read FLinesVisible write FLinesVisible;
  // Визначає видимість каліброваного імпульсу
    property StepVisible: Boolean read FStepVisible write FStepVisible;
  end;
```

При реалізації програмного забезпечення були розглянуті ряд стандартів подання біомедичних даних, оцінені їхні якості на основі оцінки відповідності представленим вимогам з метою визначення необхідності їхньої підтримки.

Самим раннім зі стандартів, що ставляться до першого класу, є специфікація EDF (European Data Format), поява якого відноситься до 1991 року. Перша специфікація, розроблена групою інженерів, що працюють у галузі розробки біомедичних систем, незважаючи на відсутність підтримки з боку якого-небудь інституту стандартизації, залучила значний інтерес із боку розроблювачів систем.

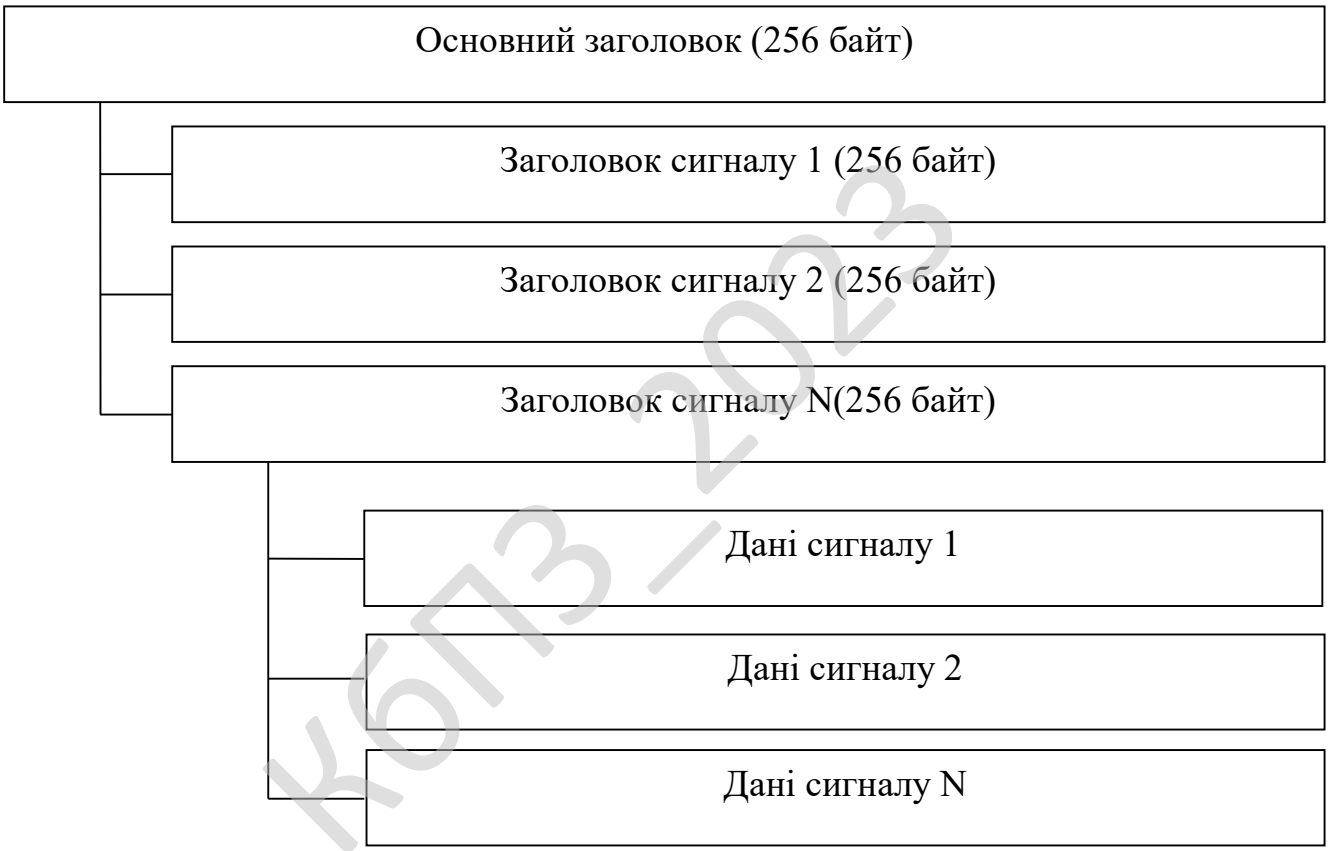
EDF швидко завоював відносну популярність, в основному, в області нейрофізіології, завдяки, насамперед, простоті реалізації. Іншим важливим

КПЗ-2023

прикладом ефективного використання даного формату є можливість його застосування в реферативних базах даних біомедичних сигналів.

Структура формату представлена на рисунку 4.3. Файл даних має деякий ASCII-заголовок, у якому втримуються основні дані про пацієнта, умови знімання й довжину сигналів.

Додатково, для кожного із сигналів створюється відповідний заголовок, у якому втримується інформація про тип сигналу і його фізичних характеристик.



КПЗ – 2023

Рисунок 4.3 – Структура формату EDF

Єдиним стандартом подання нейрофізиологічних сигналів, офіційно підтримуваним організацією по стандартизації, є специфікація ASTM 1467, розроблена відповідно до методології створення повідомлень стандарту обміну медичними записами Health Level-7.

Відповідно до основної специфікації забезпечується підтримка подання основних електрофізіологічних сигналів, а також ЕКГ.

У доповненні до формату багатоканальних записів, стандарт визначає безліч допоміжних елементів: ідентифікаторів каналів, параметрів настроювання фільтрів, каліброваних даних, характеристик електродів, різних форм анотування й інших параметрів.

Для подання цих параметрів ASTM-1467 визначає безліч типів даних, що представляються у вигляді ASCII-символів. Стандартом, орієнтованим, в основному, на завдання електрокардіографії, став формат даних ENV1064 [8], вироблений Європейським Інститутом Стандартизації (CEN), широко відомий серед розроблювачів електрокардіографів за назвою SCP-ECG (Standard Communication Protocol for ECG) [8].

Будучи офіційним стандартом і одержавши підтримку органів стандартизації, SCP-ECG швидко завоював популярність серед виробників кардіологічної техніки. ENVl064 установлює єдиний протокол передачі ЕКГ-даних як між цифровим електрокардіографом і комп'ютеризованою системою керування, так і між комп'ютерними системами різних виробників. Стандарт SCP-ECG не накладає обмежень на фізичний рівень протоколу, а лише визначає мінімально необхідні вимоги для його підтримки.

Він же регламентує деякі угоди по передачі інших даних: відомостей про пацієнта, результатів аналізу ЕКГ, умовах проведення вимірів і т.д.

Документ ENV1064 розбиває логічну послідовність ЕКГ даних (запис) на секції й описує зміст і формат подання кожної секції. Початок кожної секції перебуває по парній адресі (зсуву від 0). Тобто всі секції містять парна кількість байт – при необхідності секція доповнюється нульовим байтом. Всі секції мають ідентифікаційні номери (ID). Номера від 0 до 11 описані в SCP-ECG протоколі (таблиця 4.1), номера від 12 до 127 і номери більше 1024 включно зарезервовані для майбутнього використання. Номера від 128 до 1023 призначені для специфічних секцій виробників ЕКГ-пристроїв.

КОПЗ-2023

Таблиця 4.1 – Структура формату запису SCP-ECG

Наявність	Зміст
Обов'язково	2 байти Контрольна сума CRC16 всіх секцій і розміру всього запису
Обов'язково	4 байти Розмір всього запису в байтах
Обов'язково	Секція 0 Список покажчиків на початок кожної наступної секції
Обов'язково	Секція 1 Дані про пацієнта (ім'я, ідентифікатор, стать, дата народження й т.д.); дані про обстеження (дата, час, умови й т.д.)
Необов'язково	Секція 2 Всі таблиці Хафмена(при кодуванні по Хафмену) для розпакування вихідного сигналу (Секція 6) або репрезентативного комплексу (Секція 5) і різницевого сигналу (Секція 6) у режимі "високої компресії"
Необов'язково	Секція 3 Перерахування відведень, переданих у поточному записі, а також загальної інформації про їхнє кодування
Наявність	Зміст
Необов'язково	Секція 4 Розташування QRS-комплексів (тільки в режимі "високої" компресії)
Необов'язково	Секція 5 Для кожного відведення репрезентативний комплекс
Необов'язково	Секція 6 Для кожного відведення вихідний сигнал або, при використанні медіанного кодування, різницевий сигнал, отриманий шляхом вирахування репрезентативного комплексу з вихідного сигналу
Необов'язково	Секція 7 Загальні для всіх відведень виміру кожного комплексу в записі (тривалості, кути повороту електричних осей і т.д.) і список артефактів від штучного водія ритму.
Необов'язково	Секція 8 Текстовий діагноз від "інтерпретуючого" пристрою
Необов'язково	Секція 9 Діагностичні дані, специфічні для виробника
Необов'язково	Секція 10 Виміри, зроблені для кожного відведення окремо (амплітуди, тривалості й т.д.)
Необов'язково	Секція 11 Уніфікований закодований висновок

КОПЗ-2023

Під репрезентативним комплексом розуміється усереднений або найбільш типовий із всіх знайдених комплексів у всьому записі ЕКГ. Разом з желудочковим QRS комплексом у репрезентативному комплексі присутній зубець Р (крім випадку фибрилляцій передсердь) сегмент S-T і зубець Т.

У секціях 5 і 6 при кодуванні цифрових сигналів допускається подання їх у вигляді оригінальних відрахунків, першого або другого диференціала.

У секції 11 використовується система стандартних кодувань для одержання уніфікованого висновку. На відміну від цього в секції 8 необхідно зберігати висновок у текстовому виді. Секція 9 призначена для запису унікальних діагностичних даних, не специфікованих в описуваному стандарті. Якщо кожна із секцій 8, 9 або 11 є присутнім, то не передбачається, що всі три секції присутні.

Секції з 2 по 11 опціональні. Будь-яка SCP-ECG запис повинна в обов'язковому порядку містити тільки технологічну секцію 0 (список покажчиків на початок кожної наступної секції) і секцію 1 (дані про пацієнта; дані про обстеження). Передбачається наявність у записі й інших секціях, але перевірка їхньої присутності не відбувається.

Таким чином, стандарт SCP-ECG є жорстко орієнтованим на завдання комп'ютерної електрокардіографії і його використання для подання довільних електрофізіологічних сигналів є тяжким.

Специфікація формату CEN-TC251/FEF [8] для біомедичних сигналів опирається на об'єктно-орієнтовану модель і розширену номенклатуру стандарту ENV 13734 [83]. Номенклатура є переліком різноманітних елементів, що представляють собою певні типи медичних даних, кожний з яких однозначно визначається деяким тегом, унікальним у своєму контексті.

Кожний такий елемент отриманий на підставі моделювання предметної області, у цьому випадку біомедичних систем реєстрації й обробки даних. Елементи, маючи набір атрибутів і відповідних бінарних подань, становлять інформаційну базу медичних даних (ІБМД)[7]. Файл у форматі FEF являє собою послідовність секцій, кожна з яких визначається своїм тегом і

КОПІЯ 2023

містить об'єктні елементи, кодування яких здійснюється у відповідності зі специфікацією ENV 13734.

Основні порівняльні характеристики представлених форматів зведені в таблиці 4.2. Крім розглянутих форматів у даний момент доступно ряд інших специфікацій, однак вони не мають широкого поширення й підтримуються вузькими групами фахівців [4,5].

Таблиця 4.2 – Зведена таблиця характеристик основних стандартів подання електрофізіологічних сигналів

Характеристики	Стандарти			
	EDF	ASTM-1467	SCP-ECG	CEN-TC251/FEF
Підтримка сигналів різної природи	Будь-які еквідистантні сигнали	Нейрофізіологічні виміру	Специфічний для ЕКГ	Відповідно до номенклатури ІБМД
Формат кодування даних/можливість стиску даних	ASCII кодування для параметричних даних і бінарна для сигналів/немає	ASCII кодування/немає	Бінарна TLV-кодування/да	Бінарна TLV-кодування/да
Форми анотування	Текстова	Текстова	Текстова	Текстова Голосова Візуальна
Система сумісності	Одноріvнева	Багаторівнева	Багаторівнева	Багаторівнева
Кількість реалізацій	Досить велико	Комерційно доступні реалізації не відомі	Досить велико	Комерційно доступні реалізації рідкі
Складність реалізації	Низька	Відносно висока	Середня	Висока

Таким чином, аналіз різнорідних стандартів показав, що має сенс підтримка стандарту SCP-ECG, як найбільш підготовленого для застосування в

КОПЗ – 2023

електрокардіографії й EDF, через його простоту й наявність великої кількості даних (наприклад, записів медичних баз даних), представлених у даному форматі.

Стандарт CEN-TC251/FEF уважається досить перспективним, тому що містить у собі можливості подання будь-яких даних, представлених в ІБМД, що покриває фактичні потреби ЕКГ, ЕЕГ, МЕГ і супутніх електрофізіологічних досліджень.

Однак стримуючим фактором застосування стандарту CEN-TC251/FEF є складність реалізації, а також мала поширеність. Передбачається, що дана проблема буде дозволена, як тільки з'являться реалізації стандарту у вигляді бібліотек, що вбудовуються. Таким чином, обсяг підтримуваних форматів для розробленого програмного забезпечення був визначений, як EDF, SCP-ECG, а також власного формату подання даних.

Розглянутий механізм інтеграції систем на основі єдиних форматів файлів подання біомедичних сигналів дозволяє в багатьох випадку вирішити проблему несумісності систем. Однак даний підхід не може повною мірою вирішити проблему plug-and-play взаємодії систем, оскільки не забезпечує підтримки динамічного інтерфейсу й тим більше режиму реального часу.

Поставлене завдання здатний вирішити тільки єдиний стандарт взаємодії пристроїв і інформаційних систем.

КОПЗ-2023

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC

проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

КПЗ-2023

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 696, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

КОПІЯ – 2023

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

КбПЗ_2023

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

– Меню з розділами: Файл; Пацієнти; ЕКГ;Параметри; Довідка.

– Налаштування візуального представлення ЕКГ.

– Візуалізація ЕКГ.

Основні можливості розробленої системи: Здійснення процесу знімання ЕКГ; Візуалізація ЕКГ. При цьому можна включити або виключити міліметрівку, настроїти різні параметри відображення; Роздруківка кривих (і при, необхідності, міліметрівки) на принтері.

КбПЗ 2023

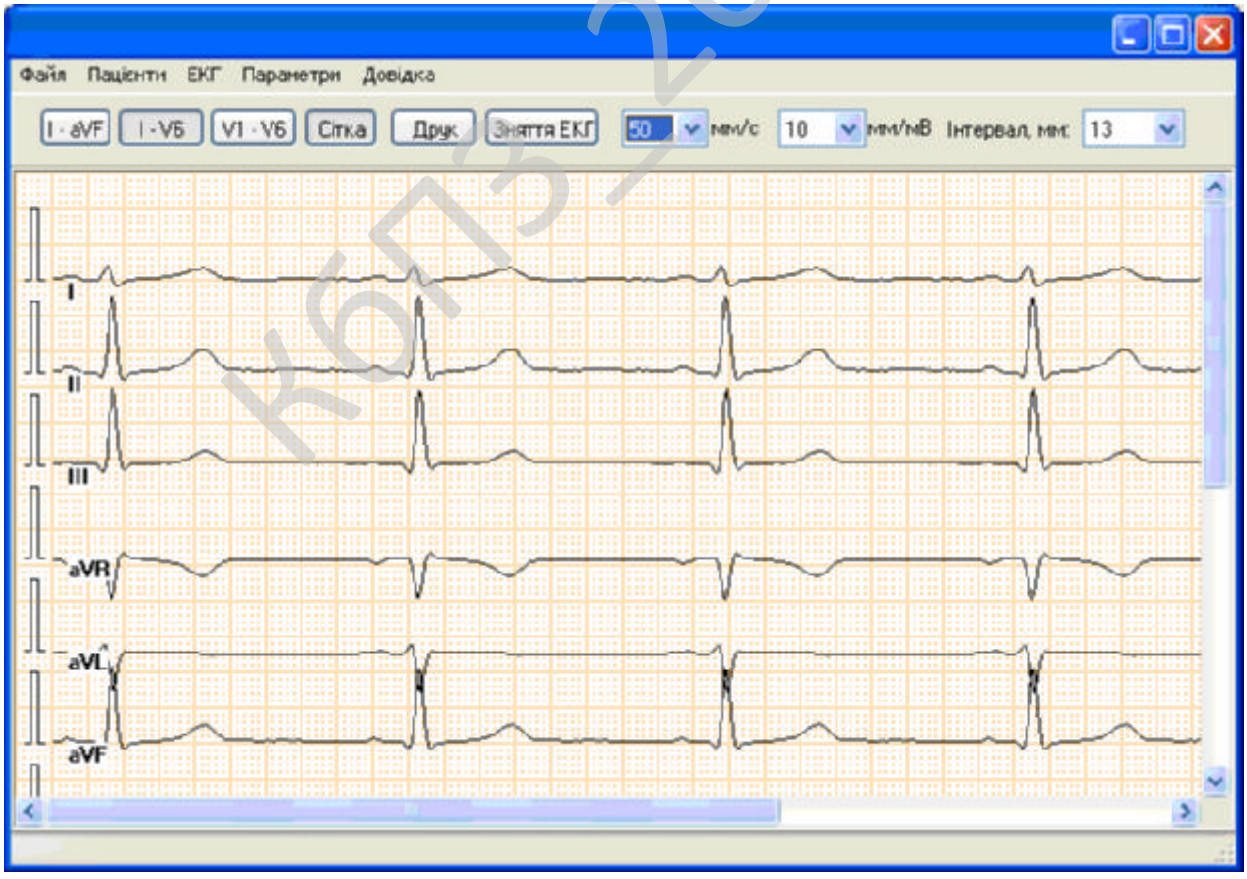


Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

Обрано умови розповсюдження – commercial software. Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником.

Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

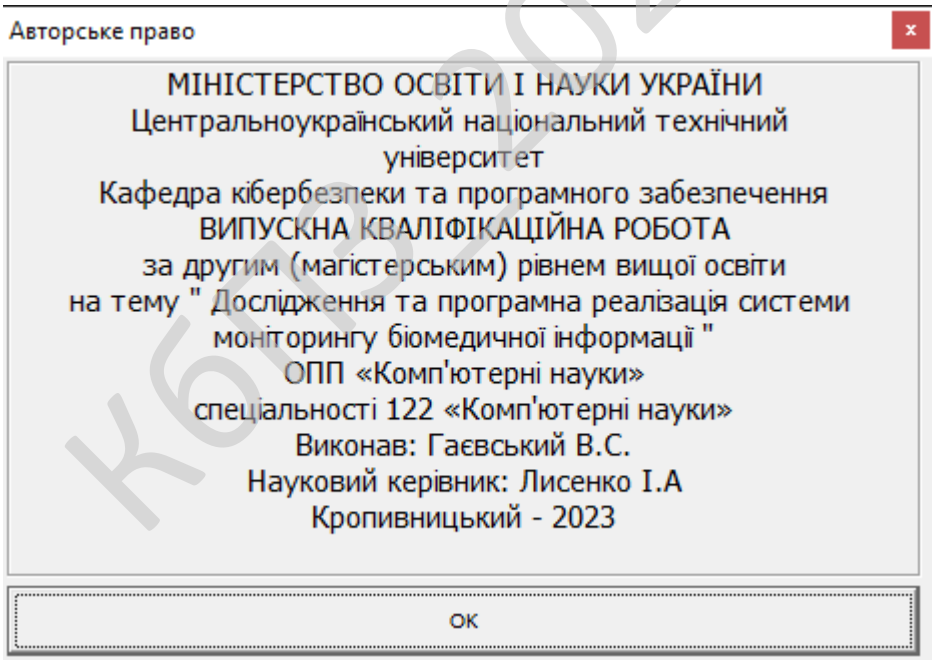


Рисунок 5.2 – Авторське право

КНТУ-2023

6 НАУКОВА НОВИЗНА

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

Метою розробки є дослідження та програмна реалізація системи моніторингу біомедичної інформації.

Об'єктом дослідження є процес моніторингу біомедичної інформації.

Предметом дослідження є методи моніторингу біомедичної інформації.

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

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

– Удосконалено метод моніторингу біомедичної інформації.

– Розроблено вітчизняний продукт моніторингу біомедичної інформації, який має більш широкі можливості, на відміну від існуючих аналогів.

КбПЗ-2023

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 моніторингу біомедичної інформації.

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

КОПЗ_2023

Продовження таблиці 7.1

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

КПІ_2023

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

КОПЗ - 2023

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:

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

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

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

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

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

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

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

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

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

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

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

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

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)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 33 = 56 \text{ люд/день.}$$

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

КОПЗ-2023

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	56	Ф 7.1-7.4
Впровадження	13	Д13
Всього	97	–

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.

$$Ч = \frac{97 \cdot 1}{60 - 5} = 1{,}8 \text{ ставки.}$$

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.

КПЗ_2023

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	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

КбПЗ_2023

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$\Phi_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 567,5/(60 \cdot 8) = 1,2 \text{ ставки.}$$

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 2019 R2, 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	

КБПЗ_2023

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	

КОПЗ_2023

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12060	36180
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	1,8	15300	82620
Інженер - електронщик	1,2	12000	43200
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	12000	18000
Всього за період розробки	$R_{сп}$ = 6,25	-	$Ф_{роб}$ = 243000

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.

$$З_{сд} = \frac{243000}{6,25 \cdot 60} = 648 \; грн.$$

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:

КбПЗ_2023

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

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

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 29000 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:

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 20.10.23 – джерело http://computorg.ua/ru/price.html

КбПЗ – 2023

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	AMD A8-8670E PRO (AD867BAHM44AB) AM4, 4 ядра, 4 потоки, 2.8 GHz, 3.3 GHz, TDP - 35 Вт, 28nm	-
Системна плата	ASUS PRIME A320M-K сокет - AM4, DDR4, 3200 MHz, LAN - 1 Гбіт/с, D-Sub (VGA), HDMI, 1 x M.2, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	AMD Radeon Graphics	-
Жорсткий диск	SSD 2.5" 256GB Mibrand, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 16GB 3200 MHz Dato, 8 ГБ, В наборі - 2	-
DVD-привод	-	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA-489, PSU 350W(FSP Brand: ATX-450PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	-

КОПЗ-2023

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All 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

КОПЗ_2023

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 Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

КПІ – 2023

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.

$$З_о = 648 \cdot 97/100 = 629 \text{ грн.}$$

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.

КОПЗ_2023

Згідно виданих викладачем норм приймаємо одну пачку паперу на три місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 103 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

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

$$З_{М1} = 103 \cdot 3 = 309 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн./шт.

$$З_{М2} = 12 \cdot 100 + 15 = 1215 \text{ грн.}$$

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.):

КбПЗ-2023

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

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:

КОПІЯ 2023

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

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

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – 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.

КПЗ_2023

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)	$З_р$	45628	33550
2. Витрати на електроенергію	$З_{ел}$	711	523
3. Витрати на амортизацію	$З_{ам}$	0	807
Всього витрат за рік	$І$	46339	34880

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 340 годин на рік до 250 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 340 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 45628 \; грн,$$

до:

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

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 ($Ц_{ел}$):

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

$$З_{ел\,баз} = 0{,}55 \cdot 340 \cdot 3{,}8 = 711 \text{ грн.}$$

$$З_{ел\,нов} = 0{,}55 \cdot 250 \cdot 3{,}8 = 523 \text{ грн.}$$

КПІЗ_2023

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.

КПЗ_2023

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

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.	10653
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,28

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

КПЗ_2023

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

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

$К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (46339\text{-}34880)\text{-}0{,}25 \cdot 3226 = 10653 \text{ грн.}$$

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

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

$$T_{сп} = \frac{3226}{46339 - 34880} = 0{,}28 \text{ року}.$$

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.

КбПЗ-2023

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні;

– Положення про охорону праці;

– Накази з охорони праці;

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час

КПЗ – 2023

роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

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

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

КПЗ-2023

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т. ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м'язовий апарат.

КПЗ_2023

8.3 Аналіз умов праці на робочому місці фахівця

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 4,4 м×6,2 м×2,9 м. Одна з її більших стін має шість двостулкових вікон, розмірами 2 м×1,8 м, які виходять на північний захід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита лінолеумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8 м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1255 мм×845 мм. Висота столів 760 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 м2 на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007-98 [3], вона повинна становити 20 м3/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 [4] роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

КПЗ – 2023

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

Відповідно ДСанПіН 3.3.2.007-98 [3] рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

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

Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А)

КОПЗ_2023

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

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

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

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

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

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

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 [4] встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

КОПЗ_2023

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

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

8.4 Розробка заходів з умов поліпшення охорони праці

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

– значення мікроклімату в приміщенні не перевищує норму;

– розрахунки розміру робочого місця на одного працівника відповідають нормі;

– рівень шуму в приміщенні не становить вище норми.

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

Рекомендовані наступні заходи: За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м. Конструкція робочого місця користувача персонального комп’ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має

КОПЗ_2023

регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм). Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим [5].

8.5 Розрахункова частина

Питання охорони праці та правила безпеки при роботі з офісною, комп'ютерною та мережевою технікою розглянуті у працях вітчизняних вчених [5].

Правильна організація експлуатації й обслуговування комп'ютерної та офісної техніки, контрольно-вимірювальної апаратури оговорена "Правилами технічної експлуатації електроустановок споживачів і правил техніки безпеки при експлуатації електроустановок споживачів".

Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлю-вального пристрою $R_{зн}$ = 10 Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з$ = 100 Ом/м;

– тип грунту – суглинок;

– тип заземлювача – труба, діаметром d=0.055 м і довжиною l =1.6 м; – конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [6].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_3$$

КПЗ-2023

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 1.6 м при глибині закладання від вершини h = 0.7 м ϕ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

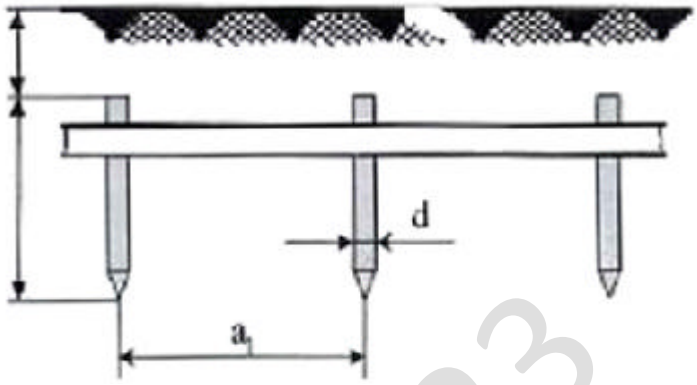


Рисунок 8.1 – Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_B = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де l – довжина заземлювача (l = 1.6 м);

d = 0.055 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.7 + 1.6/2 = 1.5 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 1.6}\left(\ln\left(\frac{2 \cdot 1.6}{0.055}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.5 + 1.6}{4 \cdot 1.5 - 1.6}\right)\right) = 48.28 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_в}{R_{зн}} = \frac{48.28}{10} = 4.8 \approx 5 \text{ шт.}$$

КБПЗ 2023

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 2.3 м. Таким чином коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_В = 0{,}6$.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_В$ приблизно складає

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{48.28}{10 \cdot 0.6} = 8.04 \approx 8 \; шт.$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 2.3 \cdot 8 = 17.4 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h = 0.6 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 17.4} \cdot \ln \frac{2 \cdot 17.4^2}{0.045 \cdot 0.6} = 9.976 \text{ Ом}$$

де b = 0.04 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{48.28 \cdot 9.976}{8 \cdot 9.976 \cdot 0.6 + 48.28 \cdot 0.34} = 7.71 \text{ Ом}$$

де $\eta_Г$ – коефіцієнт використання горизонтального заземлювача ($\eta_Г$ = 0.34).

Маємо 7.71 Ом < 10 Ом, отже нормативне обмеження $R_з < R_{з.норм}$ виконується.

Список використаних джерел інформації

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

2. Наказ Міністерства соціальної політики України 14.02.2018 № 207

КОПЗ_2023

«Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508 (дата звернення 19.10.22).

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

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

5. Катренко Л.А., Катренко А.В. Охорона праці в галузі комп'ютингу: Підручник. За науковою редакцією В.В. Пасічника. – Львів: «Магнолія 2006», 2012. – 544 с.

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

КбПЗ_2023

9 ОСНОВНІ ВИСНОВКИ

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

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 моніторингу біомедичної інформації.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу біомедичної інформації.

– Досліджена система моніторингу біомедичної інформації.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу біомедичної інформації.

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

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.

КПЗ-2023

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Camellia.

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 10653 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,28 роки.

КОПЗ-2023

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Han E, Qian H, Jiang B, Figetakis M, Kosyakova N, Tellides G, *et al*. A therapeutic vascular conduit to support in vivo cell-secreted therapy. NPJ Regen Med. 2021;6:40

2. Yasuda S, Tsuchiya H, Kaiho A, Guo Q, Ikeuchi K, Endo A, *et al*. Stress- and ubiquitylation-dependent phase separation of the proteasome. Nature. 2020;578:296-300

3. Tansley S, Gu N, Guzmán A, Cai W, Wong C, Lister K, *et al*. Microglia-mediated degradation of perineuronal nets promotes pain. Science. 2022;:eabl6773

4. Choi C, Park J, Kim H, Chang K, Park J, MIN K. DSCR1 upregulation enhances dural meningeal lymphatic drainage to attenuate amyloid pathology of Alzheimer's disease. J Pathol. 2021;255:296-310

5. Silva L, Doyle A, Greenwell Wild T, Dutzan N, Tran C, Abusleme L, *et al*. Fibrin is a critical regulator of neutrophil effector function at the oral mucosal barrier. Science. 2021;374:eabl5450

6. Yu H, Lu S, Gasior K, Singh D, Vazquez Sanchez S, Tapia O, *et al*. HSP70 chaperones RNA-free TDP-43 into anisotropic intranuclear liquid spherical shells. Science. 2021;371:

7. Purro S, Farrow M, Linehan J, Nazari T, Thomas D, Chen Z, *et al*. Transmission of amyloid-β protein pathology from cadaveric pituitary growth hormone. Nature. 2018;564:415-419

8. Batie M, Frost J, Frost M, Wilson J, Schofield P, Rocha S. Hypoxia induces rapid changes to histone methylation and reprograms chromatin. Science. 2019;363:1222-1226

9. Peine M, Rausch S, Helmstetter C, Fröhlich A, Hegazy A, Kuhl A, *et al*. Stable T-bet(+)GATA-3(+) Th1/Th2 hybrid cells arise in vivo, can develop directly

КОПЗ_2023

from naive precursors, and limit immunopathologic inflammation. PLoS Biol. 2013;11:e1001633

10. Delloye Bourgeois C, Gibert B, Rama N, Delcros J, Gadot N, Scoazec J, *et al*. Sonic Hedgehog promotes tumor cell survival by inhibiting CDON pro-apoptotic activity. PLoS Biol. 2013;11:e1001623

11. Wu H, Williams J, Nathans J. Morphologic diversity of cutaneous sensory afferents revealed by genetically directed sparse labeling. elife. 2012;1:e00181

12. Fang S, Wei J, Pentinmikko N, Leinonen H, Salven P. Generation of functional blood vessels from a single c-kit+ adult vascular endothelial stem cell. PLoS Biol. 2012;10:e1001407

13. Liu B, Du H, Rutkowski R, Gartner A, Wang X. LAAT-1 is the lysosomal lysine/arginine transporter that maintains amino acid homeostasis. Science. 2012;337:351-4

14. Nancy P, Tagliani E, Tay C, Asp P, Levy D, Erlebacher A. Chemokine gene silencing in decidual stromal cells limits T cell access to the maternal-fetal interface. Science. 2012;336:1317-21

15. Bazzini A, Lee M, Giraldez A. Ribosome profiling shows that miR-430 reduces translation before causing mRNA decay in zebrafish. Science. 2012;336:233-7

16. Brochet M, Collins M, Smith T, Thompson E, Sebastian S, Volkmann K, *et al*. Phosphoinositide metabolism links cGMP-dependent protein kinase G to essential Ca²? signals at key decision points in the life cycle of malaria parasites. PLoS Biol. 2014;12:e1001806

17. Wang T, Yu Y, Govindaiah G, Ye X, Artinian L, Coleman T, *et al*. Circadian rhythm of redox state regulates excitability in suprachiasmatic nucleus neurons. Science. 2012;337:839-42

18. Velasquez S, Ricardi M, Dorosz J, Fernandez P, Nadra A, Pol Fachin L, *et al*. O-glycosylated cell wall proteins are essential in root hair growth. Science. 2011;332:1401-3

КОПІЯ - 2023

19. Taniguchi K, Maeda R, Ando T, Okumura T, Nakazawa N, Hatori R, *et al*. Chirality in planar cell shape contributes to left-right asymmetric epithelial morphogenesis. Science. 2011;333:339-41

20. Janssen A, van der Burg M, Szuhai K, Kops G, Medema R. Chromosome segregation errors as a cause of DNA damage and structural chromosome aberrations. Science. 2011;333:1895-8

21. Dagdas Y, Yoshino K, Dagdas G, Ryder L, Bielska E, Steinberg G, *et al*. Septin-mediated plant cell invasion by the rice blast fungus, Magnaporthe oryzae. Science. 2012;336:1590-5

22. Saito D, Takase Y, Murai H, Takahashi Y. The dorsal aorta initiates a molecular cascade that instructs sympatho-adrenal specification. Science. 2012;336:1578-81

23. Lavieu G, Zheng H, Rothman J. Stapled Golgi cisternae remain in place as cargo passes through the stack. elife. 2013;2:e00558

24. Fiche J, Cattoni D, Diekmann N, Langerak J, Clerte C, Royer C, *et al*. Recruitment, assembly, and molecular architecture of the SpoIIIE DNA pump revealed by superresolution microscopy. PLoS Biol. 2013;11:e1001557

25. Fang M, Xie H, Dougan S, Ploegh H, van Oudenaarden A. Stochastic cytokine expression induces mixed T helper cell States. PLoS Biol. 2013;11:e1001618

26. Menacho Márquez M, García Escudero R, Ojeda V, Abad A, Delgado P, Costa C, *et al*. The Rho exchange factors Vav2 and Vav3 favor skin tumor initiation and promotion by engaging extracellular signaling loops. PLoS Biol. 2013;11:e1001615

27. Hensel Z, Weng X, Lagda A, Xiao J. Transcription-factor-mediated DNA looping probed by high-resolution, single-molecule imaging in live E. coli cells. PLoS Biol. 2013;11:e1001591

28. Zhang M, Wu P, Kelly F, Nurse P, Hang H. Quantitative control of protein S-palmitoylation regulates meiotic entry in fission yeast. PLoS Biol. 2013;11:e1001597

КОПІЯ 2023

29. Hsu P, Devisetty U, Harmer S. Accurate timekeeping is controlled by a cycling activator in Arabidopsis. elife. 2013;2:e00473

30. Cao M, Mao Z, Kam C, Xiao N, Cao X, Shen C, *et al*. PICK1 and ICA69 control insulin granule trafficking and their deficiencies lead to impaired glucose tolerance. PLoS Biol. 2013;11:e1001541

31. Maître J, Berthoumieux H, Krens S, Salbreux G, Jülicher F, Paluch E, *et al*. Adhesion functions in cell sorting by mechanically coupling the cortices of adhering cells. Science. 2012;338:253-6

32. Hu Z, Hom S, Kudze T, Tong X, Choi S, Aramuni G, *et al*. Neurexin and neuroligin mediate retrograde synaptic inhibition in C. elegans. Science. 2012;337:980-4

33. Srivastava D, Woolfrey K, Jones K, Anderson C, Smith K, Russell T, *et al*. An autism-associated variant of Epac2 reveals a role for Ras/Epac2 signaling in controlling basal dendrite maintenance in mice. PLoS Biol. 2012;10:e1001350

34. Goehring N, Trong P, Bois J, Chowdhury D, Nicola E, Hyman A, *et al*. Polarization of PAR proteins by advective triggering of a pattern-forming system. Science. 2011;334:1137-41

35. Luheshi N, Rothwell N, Brough D. The dynamics and mechanisms of interleukin-1alpha and beta nuclear import. Traffic. 2009;10:16-25

36. Gao R, Asano S, Upadhyayula S, Pisarev I, Milkie D, Liu T, *et al*. Cortical column and whole-brain imaging with molecular contrast and nanoscale resolution. Science. 2019;363:

37. Pietzsch T, Saalfeld S, Preibisch S, Tomancak P. BigDataViewer: visualization and processing for large image data sets. Nat Methods. 2015;12:481-3

38. Royer L, Weigert M, Günther U, Maghelli N, Jug F, Sbalzarini I, *et al*. ClearVolume: open-source live 3D visualization for light-sheet microscopy. Nat Methods. 2015;12:480-1

КБПЗ_2023

39. Peng H, Ruan Z, Long F, Simpson J, Myers E. V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. Nat Biotechnol. 2010;28:348-53

40. Kazandjian T, Petras D, Robinson S, van Thiel J, Greene H, Arbuckle K, *et al*. Convergent evolution of pain-inducing defensive venom components in spitting cobras. Science. 2021;371:386-390

41. Huang W, Pérez García P, Pokhilko A, Millar A, Antoshechkin I, Riechmann J, *et al*. Mapping the core of the Arabidopsis circadian clock defines the network structure of the oscillator. Science. 2012;336:75-9

42. Carothers J, Goler J, Juminaga D, Keasling J. Model-driven engineering of RNA devices to quantitatively program gene expression. Science. 2011;334:1716-9

43. Kir S, Beddow S, Samuel V, Miller P, Previs S, Suino Powell K, *et al*. FGF19 as a postprandial, insulin-independent activator of hepatic protein and glycogen synthesis. Science. 2011;331:1621-4

44. Pin P, Benlloch R, Bonnet D, Wremerth Weich E, Kraft T, Gielen J, *et al*. An antagonistic pair of FT homologs mediates the control of flowering time in sugar beet. Science. 2010;330:1397-400

45. Gandhi J, Zhang J, Xie Y, Soh J, Shigematsu H, Zhang W, *et al*. Alterations in genes of the EGFR signaling pathway and their relationship to EGFR tyrosine kinase inhibitor sensitivity in lung cancer cell lines. PLoS ONE. 2009;4:e4576

46. Singh A, Riederer B, Krabbenhöft A, Rausch B, Bonhagen J, Lehmann U, *et al*. Differential roles of NHERF1, NHERF2, and PDZK1 in regulating CFTR-mediated intestinal anion secretion in mice. J Clin Invest. 2009;119:540-50

47. Bassetti M, White J, Kappler J, Marrack P. Transgenic Bcl-3 slows T cell proliferation. Int Immunol. 2009;21:339-48

48. John K, Ragavan N, Pratt M, Singh P, Al Buheissi S, Matanhelia S, *et al*. Quantification of phase I/II metabolizing enzyme gene expression and polycyclic aromatic hydrocarbon-DNA adduct levels in human prostate. Prostate. 2009;69:505-19

КОПЗ_2023

49. Risebro C, Searles R, Melville A, Ehler E, Jina N, Shah S, *et al*. Prox1 maintains muscle structure and growth in the developing heart. Development. 2009;136:495-505

50. Qu X, Metz R, Porter W, Cassone V, Earnest D. Disruption of period gene expression alters the inductive effects of dioxin on the AhR signaling pathway in the mouse liver. Toxicol Appl Pharmacol. 2009;234:370-7

51. Modi B, Neustadter J, Binda E, Lewis J, Filler R, Roberts S, *et al*. Langerhans cells facilitate epithelial DNA damage and squamous cell carcinoma. Science. 2012;335:104-8

52. Smith E, Omerbasic D, Lechner S, Anirudhan G, Lapatsina L, Lewin G. The molecular basis of acid insensitivity in the African naked mole-rat. Science. 2011;334:1557-60

53. Salahudeen A, Thompson J, Ruiz J, Ma H, Kinch L, Li Q, *et al*. An E3 ligase possessing an iron-responsive hemerythrin domain is a regulator of iron homeostasis. Science. 2009;326:722-6

54. Chiu Y, Hornsey M, Klinge L, Jørgensen L, Laval S, Charlton R, *et al*. Attenuated muscle regeneration is a key factor in dysferlin-deficient muscular dystrophy. Hum Mol Genet. 2009;18:1976-89

55. Need A, Ge D, Weale M, Maia J, Feng S, Heinzen E, *et al*. A genome-wide investigation of SNPs and CNVs in schizophrenia. PLoS Genet. 2009;5:e1000373

56. Lesku J, Rattenborg N, Valcu M, Vyssotski A, Kuhn S, Kuemmeth F, *et al*. Adaptive sleep loss in polygynous pectoral sandpipers. Science. 2012;337:1654-8 pubmed

57. Anton A, Teruel R, Corral J, Minano A, Martínez Martínez I, Ordonez A, *et al*. Functional consequences of the prothrombotic SERPINC1 rs2227589 polymorphism on antithrombin levels. Haematologica. 2009;94:589-92

КОПІЯ 2023

58. Friedrichs F, Zugck C, Rauch G, Ivandic B, Weichenhan D, Müller Bardorff M, *et al*. HBEGF, SRA1, and IK: Three cosegregating genes as determinants of cardiomyopathy. Genome Res. 2009;19:395-403

59. Myers S, Bowden R, Tumian A, Bontrop R, Freeman C, MacFie T, *et al*. Drive against hotspot motifs in primates implicates the PRDM9 gene in meiotic recombination. Science. 2010;327:876-9

60. Lachke S, Alkuraya F, Kneeland S, Ohn T, Aboukhalil A, Howell G, *et al*. Mutations in the RNA granule component TDRD7 cause cataract and glaucoma. Science. 2011;331:1571-6

61. Chan Y, Marks M, Jones F, Villarreal G, Shapiro M, Brady S, *et al*. Adaptive evolution of pelvic reduction in sticklebacks by recurrent deletion of a Pitx1 enhancer. Science. 2010;327:302-5

62. Agrawal N, Frederick M, Pickering C, Bettegowda C, Chang K, Li R, *et al*. Exome sequencing of head and neck squamous cell carcinoma reveals inactivating mutations in NOTCH1. Science. 2011;333:1154-7

63. Bettegowda C, Agrawal N, Jiao Y, Sausen M, Wood L, Hruban R, *et al*. Mutations in CIC and FUBP1 contribute to human oligodendroglioma. Science. 2011;333:1453-5

64. Wang L, Gural A, Sun X, Zhao X, Perna F, Huang G, *et al*. The leukemogenicity of AML1-ETO is dependent on site-specific lysine acetylation. Science. 2011;333:765-9

65. Zheng X, Kammerer C, Cox L, Morrison A, Turner S, Ferrell R. Association of SLC34A2 variation and sodium-lithium countertransport activity in humans and baboons. Am J Hypertens. 2009;22:288-93

66. King B, Gillis J, Carlisle H, Dahn R. A natural deletion of the HoxC cluster in elasmobranch fishes. Science. 2011;334:1517

67. Edery P, Marcaillou C, Sahbatou M, Labalme A, Chastang J, Touraine R, *et al*. Association of TALS developmental disorder with defect in minor splicing component U4atac snRNA. Science. 2011;332:240-3

КОПЗ_2023

68. Schmitz R, Schultz M, Lewsey M, O Malley R, Urich M, Libiger O, *et al*. Transgenerational epigenetic instability is a source of novel methylation variants. Science. 2011;334:369-73

69. Rasmussen M, Guo X, Wang Y, Lohmueller K, Rasmussen S, Albrechtsen A, *et al*. An Aboriginal Australian genome reveals separate human dispersals into Asia. Science. 2011;334:94-8

70. McClelland S, Brennan G, DubÃ© C, Rajpara S, Iyer S, Richichi C, *et al*. The transcription factor NRSF contributes to epileptogenesis by selective repression of a subset of target genes. elife. 2014;3:e01267

71. Sidrauski C, Acosta Alvear D, Khoutorsky A, Vedantham P, Hearn B, Li H, *et al*. Pharmacological brake-release of mRNA translation enhances cognitive memory. elife. 2013;2:e00498

72. Kwon H, Koo J, Zufall F, Leinders Zufall T, Margolis F. Ca extrusion by NCX is compromised in olfactory sensory neurons of OMP mice. PLoS ONE. 2009;4:e4260

73. Mukherjee T, Kim W, Mandal L, Banerjee U. Interaction between Notch and Hif-alpha in development and survival of Drosophila blood cells. Science. 2011;332:1210-3

74. Berger K, Lindh R, Wierup N, Zmuda Trzebiatowska E, Lindqvist A, Manganiello V, *et al*. Phosphodiesterase 3B is localized in caveolae and smooth ER in mouse hepatocytes and is important in the regulation of glucose and lipid metabolism. PLoS ONE. 2009;4:e4671

75. Yoshizaki T, Milne J, Imamura T, Schenk S, Sonoda N, Babendure J, *et al*. SIRT1 exerts anti-inflammatory effects and improves insulin sensitivity in adipocytes. Mol Cell Biol. 2009;29:1363-74

76. da Rocha S, Charalambous M, Lin S, Gutteridge I, Ito Y, Gray D, *et al*. Gene dosage effects of the imprinted delta-like homologue 1 (dlk1/pref1) in development: implications for the evolution of imprinting. PLoS Genet. 2009;5:e1000392

КОПЗ-2023

77. Hopkins R, Rausher M. Pollinator-mediated selection on flower color allele drives reinforcement. Science. 2012;335:1090-2

78. Solomon D, Kim T, Diaz Martinez L, Fair J, Elkahloun A, Harris B, *et al*. Mutational inactivation of STAG2 causes aneuploidy in human cancer. Science. 2011;333:1039-43

79. Mäkinen N, Mehine M, Tolvanen J, Kaasinen E, Li Y, Lehtonen H, *et al*. MED12, the mediator complex subunit 12 gene, is mutated at high frequency in uterine leiomyomas. Science. 2011;334:252-5

80. Winzer T, Gazda V, He Z, Kaminski F, Kern M, Larson T, *et al*. A Papaver somniferum 10-gene cluster for synthesis of the anticancer alkaloid noscapine. Science. 2012;336:1704-8

81. Ascunce M, Yang C, Oakey J, Calcaterra L, Wu W, Shih C, *et al*. Global invasion history of the fire ant Solenopsis invicta. Science. 2011;331:1066-8

82. Jenkins D, Davis L, Stafford T, Campos P, Hockett B, Jones G, *et al*. Clovis age Western Stemmed projectile points and human coprolites at the Paisley Caves. Science. 2012;337:223-8

83. Meredith R, Janecka J, Gatesy J, Ryder O, Fisher C, Teeling E, *et al*. Impacts of the Cretaceous Terrestrial Revolution and KPg extinction on mammal diversification. Science. 2011;334:521-4

84. Swan B, Martinez Garcia M, Preston C, Sczyrba A, Woyke T, Lamy D, *et al*. Potential for chemolithoautotrophy among ubiquitous bacteria lineages in the dark ocean. Science. 2011;333:1296-300

85. Szabo P, Dogra P, Gray J, Wells S, Connors T, Weisberg S, *et al*. Longitudinal profiling of respiratory and systemic immune responses reveals myeloid cell-driven lung inflammation in severe COVID-19. Immunity. 2021;54:797-814.e6

86. Dixon G, Pan H, Yang D, Rosen B, Jashari T, Verma N, *et al*. QSER1 protects DNA methylation valleys from de novo methylation. Science. 2021;372:

КОПЗ_2023

87. Yang J, Chen J, Del Carmen Vitery M, Osei Owusu J, Chu J, Yu H, *et al*. PAC, an evolutionarily conserved membrane protein, is a proton-activated chloride channel. Science. 2019;364:395-399

88. Xu Q, Milanez Almeida P, Martins A, Radtke A, Hoehn K, Oguz C, *et al*. Adaptive immune responses to SARS-CoV-2 persist in the pharyngeal lymphoid tissue of children. Nat Immunol. 2023;24:186-199

89. Reinkemeier C, Girona G, Lemke E. Designer membraneless organelles enable codon reassignment of selected mRNAs in eukaryotes. Science. 2019;363:

90. Whitfield Z, Chisholm J, Hawley R, Orr Weaver T. A meiosis-specific form of the APC/C promotes the oocyte-to-embryo transition by decreasing levels of the Polo kinase inhibitor matrimony. PLoS Biol. 2013;11:e1001648

91. Vahl J, Heger K, Knies N, Hein M, Boon L, Yagita H, *et al*. NKT cell-TCR expression activates conventional T cells in vivo, but is largely dispensable for mature NKT cell biology. PLoS Biol. 2013;11:e1001589

92. Kerenyi M, Shao Z, Hsu Y, Guo G, Luc S, O Brien K, *et al*. Histone demethylase Lsd1 represses hematopoietic stem and progenitor cell signatures during blood cell maturation. elife. 2013;2:e00633

93. Kent D, Li J, Tanna H, Fink J, Kirschner K, Pask D, *et al*. Self-renewal of single mouse hematopoietic stem cells is reduced by JAK2V617F without compromising progenitor cell expansion. PLoS Biol. 2013;11:e1001576

94. Demogines A, Abraham J, Choe H, Farzan M, Sawyer S. Dual host-virus arms races shape an essential housekeeping protein. PLoS Biol. 2013;11:e1001571

95. Zhao J, Rao D, O Connell R, Garcia Flores Y, Baltimore D. MicroRNA-146a acts as a guardian of the quality and longevity of hematopoietic stem cells in mice. elife. 2013;2:e00537

96. Lee B, Moon J, Shu J, Yuan L, Newman Z, Schekman R, *et al*. UNC93B1 mediates differential trafficking of endosomal TLRs. elife. 2013;2:e00291

97. McCoy W, Wang X, Yokoyama W, Hansen T, Fremont D. Structural mechanism of ER retrieval of MHC class I by cowpox. PLoS Biol. 2012;10:e1001432

КОПІЯ_2023

98. Silvestre Roig C, Braster Q, Wichapong K, LEE E, Teulon J, Berrebeh N, *et al*. Externalized histone H4 orchestrates chronic inflammation by inducing lytic cell death. Nature. 2019;569:236-240

99. Young M, Mitchell T, Vieira Braga F, Tran M, Stewart B, Ferdinand J, *et al*. Single-cell transcriptomes from human kidneys reveal the cellular identity of renal tumors. Science. 2018;361:594-599

100. Michaud M, Martins I, Sukkurwala A, Adjemian S, Ma Y, Pellegatti P, *et al*. Autophagy-dependent anticancer immune responses induced by chemotherapeutic agents in mice. Science. 2011;334:1573-7

101. Seo J, Yaneva R, Hinson E, Cresswell P. Human cytomegalovirus directly induces the antiviral protein viperin to enhance infectivity. Science. 2011;332:1093-7

102. Ren D, Tu H, Kim H, Wang G, Bean G, Takeuchi O, *et al*. BID, BIM, and PUMA are essential for activation of the BAX- and BAK-dependent cell death program. Science. 2010;330:1390-3

103. Pessina F, Lowndes N. The RSF1 histone-remodelling factor facilitates DNA double-strand break repair by recruiting centromeric and Fanconi Anaemia proteins. PLoS Biol. 2014;12:e1001856

104. Alvarez J, Dodelet Devillers A, Kebir H, Ifergan I, Fabre P, Terouz S, *et al*. The Hedgehog pathway promotes blood-brain barrier integrity and CNS immune quiescence. Science. 2011;334:1727-31

105. Login F, Balmand S, Vallier A, Vincent Monegat C, Vigneron A, Weiss Gayet M, *et al*. Antimicrobial peptides keep insect endosymbionts under control. Science. 2011;334:362-5

106. Wu X, Zhou T, Zhu J, Zhang B, Georgiev I, Wang C, *et al*. Focused evolution of HIV-1 neutralizing antibodies revealed by structures and deep sequencing. Science. 2011;333:1593-602

107. Goritz C, Dias D, Tomilin N, Barbacid M, Shupliakov O, Frisen J. A pericyte origin of spinal cord scar tissue. Science. 2011;333:238-42

КОПЗ_2023

108. Notta F, Doulatov S, Laurenti E, Poeppl A, Jurisica I, Dick J. Isolation of single human hematopoietic stem cells capable of long-term multilineage engraftment. Science. 2011;333:218-21

109. Ekiert D, Friesen R, Bhabha G, Kwaks T, Jongeneelen M, Yu W, *et al*. A highly conserved neutralizing epitope on group 2 influenza A viruses. Science. 2011;333:843-50

110. Beatty G, Chiorean E, Fishman M, Saboury B, Teitelbaum U, Sun W, *et al*. CD40 agonists alter tumor stroma and show efficacy against pancreatic carcinoma in mice and humans. Science. 2011;331:1612-6

111. van Oeffelen L, Peeters E, Nguyen Le Minh P, Charlier D. The 'Densitometric Image Analysis Software' and its application to determine stepwise equilibrium constants from electrophoretic mobility shift assays. PLoS ONE. 2014;9:e85146

112. Nakamura T, Oh C, Liao L, Zhang X, Lopez K, Gibbs D, *et al*. Noncanonical transnitrosylation network contributes to synapse loss in Alzheimer's disease. Science. 2021;371:

113. Trizzino M, Zucco A, Deliard S, Wang F, Barbieri E, Veglia F, *et al*. EGR1 is a gatekeeper of inflammatory enhancers in human macrophages. Sci Adv. 2021;7:

114. Clairfeuille T, Cloake A, Infield D, Llongueras J, Arthur C, Li Z, *et al*. Structural basis of α-scorpion toxin action on Nav channels. Science. 2019;363:

115. Kaelberer M, Buchanan K, Klein M, Barth B, Montoya M, Shen X, *et al*. A gut-brain neural circuit for nutrient sensory transduction. Science. 2018;361:

116. Field C, Baixauli F, Kyle R, Puleston D, Cameron A, Sanin D, *et al*. Mitochondrial Integrity Regulated by Lipid Metabolism Is a Cell-Intrinsic Checkpoint for Treg Suppressive Function. Cell Metab. 2020;31:422-437.e5

117. Szonyi A, Sos K, Nyilas R, Schlingloff D, Domonkos A, Takács V, *et al*. Brainstem nucleus incertus controls contextual memory formation. Science. 2019;364:

КОПІЯ - 2023

118. Schindelin J, Arganda Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, *et al*. Fiji: an open-source platform for biological-image analysis. Nat Methods. 2012;9:676-82

119. Weber C, Zhou Y, Lee J, Looger L, Qian G, Ge C, *et al*. Temperature-dependent sex determination is mediated by pSTAT3 repression of Kdm6b. Science. 2020;368:303-306

120. Kudryashev M, Stenta M, Schmelz S, Amstutz M, Wiesand U, Castaño Diez D, *et al*. In situ structural analysis of the Yersinia enterocolitica injectisome. elife. 2013;2:e00792

121. Szabó A, Papin C, Zorn D, Ponien P, Weber F, Raabe T, *et al*. The CK2 kinase stabilizes CLOCK and represses its activity in the Drosophila circadian oscillator. PLoS Biol. 2013;11:e1001645

122. Andersen K, Onischenko E, Tang J, Kumar P, Chen J, Ulrich A, *et al*. Scaffold nucleoporins Nup188 and Nup192 share structural and functional properties with nuclear transport receptors. elife. 2013;2:e00745

123. Wang X, Kim J, Bazzi M, Robinson S, Collins C, Ye B. Bimodal control of dendritic and axonal growth by the dual leucine zipper kinase pathway. PLoS Biol. 2013;11:e1001572

124. Mitrovic S, Nogueira C, Cantero Recasens G, Kiefer K, Fernández Fernández J, Popoff J, *et al*. TRPM5-mediated calcium uptake regulates mucin secretion from human colon goblet cells. elife. 2013;2:e00658

125. Lim B, Miyazaki R, NEHER S, Siegele D, Ito K, Walter P, *et al*. Heat shock transcription factor ?32 co-opts the signal recognition particle to regulate protein homeostasis in E. coli. PLoS Biol. 2013;11:e1001735

126. Antón Bolaños N, Sempere Ferràndez A, Guillamon Vivancos T, Martini F, Pérez Saiz L, Gezelius H, *et al*. Prenatal activity from thalamic neurons governs the emergence of functional cortical maps in mice. Science. 2019;:

КОПЗ-2023

127. Chiron C, Raynaud C, Tzourio N, Diebler C, Dulac O, Zilbovicius M, *et al*. Regional cerebral blood flow by SPECT imaging in Sturge-Weber disease: an aid for diagnosis. J Neurol Neurosurg Psychiatry. 1989;52:1402-9 pubmed

128. Berg S, Kutra D, Kroeger T, Straehle C, Kausler B, Haubold C, *et al*. ilastik: interactive machine learning for (bio)image analysis. Nat Methods. 2019;:

129. Гаєвський В.С. Дослідження та програмна реалізація системи моніторингу біомедичної інформації // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

130. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.

131. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.

132. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

133. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.

134. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.

135. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.

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

137. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level

КОПЗ-2023

of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

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

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

140. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

141. 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.

142. 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.

143. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

144. 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.

КОПЗ-2023

145. 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.

146. 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.

147. 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.

148. 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.

149. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

150. 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.

151. 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.

КОПЗ_2023

152. 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.

153. 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.

154. 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.

155. 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.

156. 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.

157. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

158. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

КБПЗ – 2023

159. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

160. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

161. 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

162. 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.

163. 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.

164. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

165. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

КОПЗ-2023

166. 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.

167. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

168. 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).

169. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

КБПЗ_2023

Додаток А
(обов'язковий)

Технічне завдання

Зміст

КПЗ–2023

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи моніторингу біомедичної інформації.

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. № 37-13 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 моніторингу біомедичної інформації.

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;

КОПЗ_2023

– 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 моніторингу біомедичної інформації;

– 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.

КБПЗ_2023

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.

КбПЗ – 2023

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

КбПЗ_2023

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 – 130 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 14.12.2023 р.

КПЗ – 2023

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Лисенко І.А

КбПЗ_2023

Дослідження та програмна реалізація системи моніторингу біомедичної інформації

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 85

Літера: РП

Кропивницький – 2023 року

Biomedical_inf.dpr - файл проекту

```
program Biomedical_inf;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1},//основна програма
  EKG in 'EKG.pas' {EKGFrame: TFrame},// створення ЕКГ
  about in 'about.pas' {AboutForm};// довідка

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'Система моніторингу біомедичної інформації';
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TAboutForm, AboutForm);
  Application.Run;
end.
```

КбПЗ_2023

Unit1.pas - основна програма

```
unit Unit1;


interface


uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
about
  Dialogs, EKG, StdCtrls, Matrix, ExtCtrls, Buttons, AppEvnts;


type
  TForm1 = class(TForm)
    EKGFrame1: TEKGFrame;
    Panel1: TPanel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    ComboBox1: TComboBox;
    Label4: TLabel;
    Label5: TLabel;
    ComboBox2: TComboBox;
    Label6: TLabel;
    ComboBox3: TComboBox;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    procedure FormCreate(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure ComboBox1Select(Sender: TObject);
    procedure ComboBox2Select(Sender: TObject);
    procedure ComboBox3Select(Sender: TObject);
    procedure SpeedButton5Click(Sender: TObject);
    procedure SpeedButton6Click(Sender: TObject);

 private
    { Private declarations }

public
    { Public declarations }
  end;


var
  Form1: TForm1;
  Signals: array[1..12] of string =
    ('I', 'II', 'III', 'aVR', 'aVL', 'aVF', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6');

 FIdx: Integer;
implementation


{$R *.dfm}



procedure TForm1.FormCreate(Sender: TObject);
begin
  ChDir(ExtractFilePath(ParamStr(0)));
  Base.LoadFromBinFile('EKG.bin');
  SpeedButton2.Click;
  FIdx := Base.GetIndex('EKG')
end;
```

КОТБ 2023

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
var
  SixOtv: array[1..6] of string;
  I: Integer;
begin
  if Sender = SpeedButton2 then begin
    EKGFrame1.LoadSignals(Base, 'EKG');
    EKGFrame1.SetNames(Signals);
  end;


  if Sender = SpeedButton3 then begin
    for I := 1 to 6 do
    SixOtv[I] := Signals[I];
    Base.CopySubmatrix('EKG', 'EKG1', 1, 6, 1, Base.GetCols('EKG'));
    EKGFrame1.LoadSignals(Base, 'EKG1');
    EKGFrame1.SetNames(SixOtv);
  end;


  if Sender = SpeedButton4 then begin
    for I := 1 to 6 do
    SixOtv[I] := Signals[I + 6];
    Base.CopySubmatrix('EKG', 'EKG1', 7, 6, 1, Base.GetCols('EKG'));
    EKGFrame1.LoadSignals(Base, 'EKG1');
    EKGFrame1.SetNames(SixOtv);
  end;


  if FileExists('PrintInfo.txt') then
  with TStringList.Create do begin
    LoadFromFile('PrintInfo.txt');
    if (Count > 0) and (Pos('HEADINFO', Strings[0]) = 1) then
      EKGFrame1.HeadText := Copy(Strings[0], 10, Length(Strings[0]) - 9);
    if (Count > 1) and (Pos('FOOTINFO', Strings[1]) = 1) then
      EKGFrame1.FootText := Copy(Strings[1], 10, Length(Strings[1]) - 9);
    if (Count > 2) and (Pos('PAGEINFO', Strings[2]) = 1) then
      EKGFrame1.PageInfo := Copy(Strings[2], 10, Length(Strings[2]) - 9);
    Free;
  end;


  EKGFrame1.GridVisible := SpeedButton1.Down;
  EKGFrame1.Interval := StrToInt(ComboBox3.Text);
  EKGFrame1.DrawImage;
end;


procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  EKGFrame1.GridVisible := SpeedButton1.Down;
  EKGFrame1.DrawImage;
end;


procedure TForm1.ComboBox1Select(Sender: TObject);
begin
  if EKGFrame1.IsProcess then begin
    EKGFrame1.StopProcess;
    EKGFrame1.SetParams(0, 0, 0, StrToInt(ComboBox1.Text));
    EKGFrame1.StartProcess(12);
    Exit;
  end;
  EKGFrame1.SetParams(0, 0, 0, StrToInt(ComboBox1.Text));
  EKGFrame1.DrawImage;
end;
```

КОПИЯ 2023

```
procedure TForm1.ComboBox2Select(Sender: TObject);
begin
  EKGFrame1.SetParams(0, 0, StrToInt(ComboBox2.Text), 0);
  EKGFrame1.DrawImage;
end;


procedure TForm1.ComboBox3Select(Sender: TObject);
begin
  EKGFrame1.Interval := StrToInt(ComboBox3.Text);
  EKGFrame1.DrawImage;
end;


procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  EKGFrame1.Print;
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
{
У даній процедурі реалізована всього лише імітація процесу знімання ЕКГ. Однак
цілком можливо виконувати знімання реальних сигналів ЕКГ із реальних
кардіографів
}
var
  I, J, Cols: Integer;
  Ar: Array[1..12] of Real;
begin
  Cols := Base.GetCols(FIdx);
  if SpeedButton6.Down then
    EKGFrame1.StartProcess(12) else EKGFrame1.StopProcess;
  while SpeedButton6.Down do
    for J := 1 to Cols do
    begin
      for I := 1 to 12 do
        Ar[I] := Base.Elem[FIdx, I, J];
      EKGFrame1.AddPoints(Ar);

      // Можливість спостерігати знімання ЕКГ. Чим більше число, тим швидше буде
виконуватися
      if J mod 5 = 0 then Application.ProcessMessages;
    end;
end;

end.
```

КбПЗ_2023

Signals.pas - містить функції для обробки сигналів

```
{Даний модуль використовується для обробки сигналів ЕКГ}

unit Signals;

interface
uses
  Windows, Matrix, Math;

const
  matFastInterp = 'Швидка інтерполяція';
  matIncSizeRows = 'Розтягання сигналу';
  matDecSizeRows = 'Стиск сигналу';
  matNormAmplitude = 'Нормування амплітуди';
  matSmoothSignals = 'Згладжування сигналу';
  matFindIzoline = 'Пошук ізолінії';
  matFindMaxPoints = 'Пошук точок екстремума';

{Функції швидкої інтерполяції}
procedure mFastInterp(Ws: TWorkspace; Xvek, Yvek, X1vek, RESvek: string;
  Faster: Boolean = False);
//procedure mIncSizeRows(Ws: TWorkspace; SourAr, DestAr: string; NumPoints:
Integer);
//procedure mDecSizeRows(Ws: TWorkspace; SourAr, DestAr: string; NumPoints:
Integer);

{Здійснює стиск або розтягання сигналів у рядках масиву}
procedure mScaleSignals(Ws: TWorkspace; SourAr, DestAr: string; DestLen:
Integer);

{Згладжування сигналів}
procedure mSmoothSignals(Ws: TWorkspace; SourAr, DestAr: string; Iter: Integer);

{Нормалізація амплітуди сигналів}
procedure mNormAmplitude(Ws: TWorkspace; SourAr, DestAr: string; MaxValue:
Real);

{Пошук ізолінії}
function mFindIzoline(Ws: TWorkspace; MatName: string; Row: Integer;
  Koef: Integer = 20): Real;

{Пошук точок екстремума}
function mFindMaxPoints(Ws: TWorkspace; SourAr, DestAr: string; Line: Real;
  FindMax: Boolean = True): Boolean;

implementation

procedure mFastInterp(Ws: TWorkspace; Xvek, Yvek, X1vek, RESvek: string;
  Faster: Boolean = False);
var
  I, J, K, Xrows, Xcols, Yrows, Ycols, X1rows, X1cols, ColNum: Integer;
  Xid, Yid, X1id, Rid: Integer;
  Tmp: Real;

  function CalcElem(X0, X1, Y0, Y1, Y2, X: Real): Real;
  var E: Real;
  begin
    E := (X - X0) / (X1 - X0);
    if (E < 0) or (E > 2) then Ws.DoError('Intepr Error: calc elem');
    Result := Y0 + (Y1 - Y0) * E + (Y2 - 2 * Y1 + Y0) / 2 * E * (E - 1);
  end;
begin
  with TWorkspace.Create(matFastInterp, Ws) do
  begin
    CheckResAr(RESvek);
    Xid := GetSize(LoadArray(Xvek, 'X'), Xrows, Xcols);
```

КоПЗ_2023

```
    Yid := GetSize(LoadArray(Yvek, 'Y'), Yrows, Ycols);
    X1id := GetSize(LoadArray(X1vek, 'X1'), X1rows, X1cols);

    if (Xrows > 1) or (Xcols <> Ycols) or (X1rows > 1) then
      DoError(matArraysNotAgree);

    Rid := NewArray('R', Yrows, X1cols);
    ColNum := Xcols;
    for I := X1cols downto 1 do
    begin
      Tmp := Elem[X1id, 1, I];
      // Шукаємо точку в масиві вихідних відліків
      for J := ColNum downto 1 do
        if Tmp >= Elem[Xid, 1, J] then
        begin
          for K := 1 to Yrows do
          begin
            if J = Xcols then Elem[Rid, K, I] := Elem[Yid, K, J] else
              if J = Xcols - 1 then
                Elem[Rid, K, I] :=
                  CalcElem(Elem[Xid, 1, Xcols - 2], Elem[Xid, 1, Xcols - 1],
                  Elem[Yid, K, Xcols - 2], Elem[Yid, K, Xcols - 1], Elem[Yid,
                  K, Xcols], Elem[X1id, 1, I])
            else
              Elem[Rid, K, I] :=
                CalcElem(Elem[Xid, 1, J], Elem[Xid, 1, J + 1],
                Elem[Yid, K, J], Elem[Yid, K, J + 1], Elem[Yid, K, J + 2],
                Elem[X1id, 1, I]);
          end;
          if Faster then ColNum := J;
          Break;
        end;

    end; // for I
    ReturnArray('R', RESvek);
    Free();
  end;
end;

procedure mIncSizeRows(Ws: TWorkspace; SourAr, DestAr: string; NumPoints:
Integer);
var
  Cols: Integer;
begin
  with TWorkspace.Create(matIncSizeRows, Ws) do
  begin
    Cols := GetCols(LoadArray(SourAr, 'Sour'));
    if NumPoints < Cols then DoError(matBadInputData);
    NewArray('X', 1, Cols, 1); // Створюємо масив відліків
    NewFillAr('X1', 1, Cols, NumPoints);
    mFastInterp(SelfWS, 'X', 'Sour', 'X1', 'Y', True);
    ReturnArray('Y', DestAr);
    Free();
  end;
end;

procedure mDecSizeRows(Ws: TWorkspace; SourAr, DestAr: string; NumPoints:
Integer);
var
  Interv, Num, yIdx, tIdx, I, J, Rows, C, Cols: Integer;
begin
  with TWorkspace.Create(matDecSizeRows, Ws) do
  begin
    CheckResAr(DestAr);
    GetSize(LoadArray(SourAr, 'Sour'), Rows, Cols);
    if NumPoints > Cols then DoError(matBadInputData);
    // Створюємо масив зі збільшеним числом стовпців
    Interv := Cols div (NumPoints - 1) + 1;
    Num := Interv * (NumPoints - 1) + 1;
```

КбПЗ_2023

```
    mIncSizeRows(SelfWS, 'Sour', 'Temp', Num);
    tIdx := GetIndex('Temp');
    yIdx := NewArray('Y', Rows, NumPoints);
    C := 0;
    for J := 1 to Num do
    begin
      if (J mod Interv) <> 1 then Continue;
      Inc(C);
      for I := 1 to Rows do Elem[yIdx, I, C] := Elem[tIdx, I, J];
    end;
    ReturnArray('Y', DestAr);
    Free();
  end;
end;

procedure mScaleSignals(Ws: TWorkspace; SourAr, DestAr: string; DestLen:
Integer);
begin
  Ws.CheckResAr(DestAr);
  if DestLen < Ws.GetCols(SourAr) then
    mDecSizeRows(WS, SourAr, DestAr, DestLen) else
    mIncSizeRows(WS, SourAr, DestAr, DestLen);
end;

procedure mNormAmplitude(Ws: TWorkspace; SourAr, DestAr: string; MaxValue:
Real);
var
  sIdx, tIdx, Rows, Cols, I, J: Integer;
  Koef: Real;
begin
  with TWorkspace.Create(matNormAmplitude, Ws) do
  begin
    CheckResAr(DestAr);
    sIdx := GetSize(LoadArray(SourAr, 'A', False), Rows, Cols);
    GetMinMax('A', 'Temp'); // Визначаємо  min і max для кожного рядка
    CalcFunc('Temp', 'Temp', fncAbs); // Міняємо мінуси на плюси
    GetMax('Temp', 'Temp', '', dimRows); // Визначаємо max елементи
    // Тепер у векторі Temp зберігаються максимальні елементи для кожного рядка
    tIdx := GetIndex('Temp');
    for I := 1 to Rows do
    begin
      Koef := MaxValue / Elem[tIdx, I, 1]; // Коефіцієнт шкалювання
      for J := 1 to Cols do
        Elem[sIdx, I, J] := Elem[sIdx, I, J] * Koef;
    end;
    ReturnArray('A', DestAr); // Передаємо результат
    Free();
  end;
end;


procedure mSmoothSignals(Ws: TWorkspace; SourAr, DestAr: string; Iter: Integer);
var
  RowCnt, ColCnt, A, rv: Integer;
  I, Z, II: Integer;
  q, w, k: extended;
begin
  with TWorkspace.Create(matSmoothSignals, Ws) do
  begin
    CheckResAr(DestAr); // Перевірка імені результату
    A := GetSize(LoadArray(SourAr, 'A', False), RowCnt, ColCnt);
    // створюємо змінну для тимчасового зберігання результату
    rv := NewArray('rv', RowCnt, ColCnt);
    I := 0;
    for z := 1 to iter do
    begin
      for II := 1 to RowCnt do
      begin
        I := 1;
```

КПІ 2023

```
        while I <> ColCnt do
        begin
          q := Elem[A, II, I];
          w := Elem[A, II, I + 1];
          k := (q - w) / 4;
          Elem[rv, II, I] := (q + w) / 2 + k;
          Elem[rv, II, I + 1] := (q + w) / 2 - k;
          Inc(I);
        end;
      end;
      A := CopyArray('rv', 'A');
      for II := 1 to RowCnt do
      begin
        while i <> 1 do
        begin
          q := Elem[A, II, I];
          w := Elem[A, II, I - 1];
          k := (w - q) / 4;
          Elem[rv, II, I] := (q + w) / 2 - k;
          Elem[rv, II, I - 1] := (q + w) / 2 + k;
          I := I - 1;
        end;
      end;
      A := CopyArray('rv', 'A');
    end;
    ReturnArray('A', DestAr);
    Free();
  end;
end;

{Спеціалізована функція, застосовувана для обробки сигналів ЕКГ}
function mFindIzoline(Ws: TWorkspace; MatName: string; Row: Integer;
  Koef: Integer = 20): Real;
var
  Cols, I, J, Leng: Integer;
  AbsMax, AbsMin, MaxEl, MinEl: Real;
  Tmp, C: Integer;
begin
  with TWorkspace.Create(matFindIzoline, Ws) do
  begin
    LoadArray(MatName, MatName); // Копіювання посилання
    C := 0;
    Cols := GetCols(MatName);
    CopyCutRows(MatName, 'Temp1', Row, 1);
    CopyArray('Temp1', 'SomeArray');

    mNormAmplitude(SelfWS, 'Temp1', 'Temp1', Koef);
    CalcFunc('Temp1', 'Temp1', fncRound);
    GetMinMax('Temp1', MinEl, MaxEl);
    Leng := Round(MaxEl) - Round(MinEl) + 1;

    GetMinMax('SomeArray', AbsMin, AbsMax);
    AbsMax := Max(abs(AbsMin), abs(AbsMax));

    NewArray('Temp2', 2, Leng, True);
    for I := 1 to Cols do
    begin
      Tmp := Round(Elem['Temp1', 1, I]);
      if Tmp = 10000000 then Continue;
      Inc(C);
      for J := I to Cols do
      begin
        if Round(Elem['Temp1', 1, J]) = Tmp then
        begin
          Elem['Temp2', 1, C] := Elem['Temp2', 1, C] + 1;
          Elem['Temp2', 2, C] := Tmp;
          Elem['Temp1', 1, J] := 10000000;
        end;
      end;
```

КПЗ_2023

```
    end;
    GetMinOrMax('Temp2', 1, dimRows, True, I);
    Result := Elem['Temp2', 2, I] * AbsMax / Koef;
    Free();
  end;
end;

{Функція знаходження екстремумів.}
function mFindMaxPoints(Ws: TWorkspace; SourAr, DestAr: string; Line: Real;
  FindMax: Boolean = True): Boolean;
var
  I, J, CC: Integer;
  El1, El2, MaxEl: Real;
  MaxInd: Integer;
  sIdx, kIdx, IResArray: Integer;
begin
  with TWorkspace.Create(matFindMaxPoints, Ws) do
  begin
    CheckResAr(DestAr);
    sIdx := LoadArray(SourAr, 'A');
    kIdx := 0;
    Result := False;
    if GetRows(sIdx) <> 1 then DoError(matIsNotVektor);
    CC := GetCols(sIdx);
     // У циклі перебираємо всі елементи вхідного масиву
    for I := 1 to CC - 1 do
    begin
      El1 := Elem[sIdx, 1, I];
      El2 := Elem[sIdx, 1, I + 1];
       // Якщо Line лежить у межах [El1; El2]
      if ((Line >= El1) and (Line < El2)) or ((Line > El2) and (Line <= El1))
then
      begin
        if not ArrayExists('B') then
          kIdx := NewArray('B', 1, 1) else
          Resize('B', 1, GetCols(kIdx) + 1);
        Elem[kIdx, 1, GetCols(kIdx)] := I
      end;
    end;
    if (not ArrayExists('B')) or (GetCols(kIdx) < 2) then
    begin
      Free;
      Exit;
    end;
    // Визначаємо позицію першого елемента
    I := Round(Elem[kIdx, 1, 1]);
    if (I > 1) and FindMax and (Elem[sIdx, 1, I - 1] > Elem[sIdx, 1, I]) then
      CopyCutCols('B', '', 1, 1, True);
    if (not ArrayExists('B')) or (GetCols(kIdx) < 2) then
    begin
      Free;
      Exit;
    end;
    if (I > 1) and (not FindMax) and (Elem[sIdx, 1, I - 1] < Elem[sIdx, 1, I])
then
      CopyCutCols('B', '', 1, 1, True);
    if (Find('B') = -1) or (GetCols(kIdx) < 2) then
    begin
      Free;
      Exit;
    end;
    if (GetCols(kIdx) mod 2) <> 0 then // Видалення останього елемента
      CopyCutCols('B', '', GetCols(kIdx) - 1, 1, True);
    // Створюємо вихідний масив
    CC := GetCols(kIdx);
    IResArray := NewArray('X', 1, CC div 2);
    for I := 1 to (CC div 2) do
    begin
      El1 := Elem[kIdx, 1, I * 2 - 1];
```

КбПЗ - 2023

```
      El2 := Elem[kIdx, 1, I * 2];
      if FindMax then
        MaxEl := -MaxDouble else
        MaxEl := MaxDouble;
      MaxInd := 0;
      for J := Round(El1) to Round(El2) do
      begin
        if ((Elem[sIdx, 1, J] > MaxEl) and FindMax) or
          ((Elem[sIdx, 1, J] < MaxEl) and (not FindMax)) then
        begin
          MaxEl := Elem[sIdx, 1, J];
          MaxInd := J;
        end;
      end;
      Elem[IResArray, 1, I] := MaxInd;
    end;
    if Find('X') = -1 then
    begin
      Free;
      Exit;
    end;
    ReturnArray('X', DestAr);
    Result := True;
    Free();
  end;
end;

end.
```

КбПЗ_2023

Matrix.pas - ядро системи матричних обчислень

```
{

Це засіб, призначений для роботи із двомірними масивами речовинних чисел.
Підтримується всього один формат - Real (або Double). Ключовим моментом у даній
розробці є використання робочої області TWorkspace, що є сховищем практично
необмеженого числа масивів. Кількість робочих областей і розміри масивів також
ніщо не обмежує. Іншим важливим моментом є зручність роботи з окремими масивами.
Вам навіть не потрібно їх повідомляти. Кожний масив має своє унікальне імені (у
межах однієї робочої області), і саме по цьому імені й здійснюється звертання до
даного масиву. Приклад створення нульового масиву розміром 6 x 5:
Base.NewZeros('MyFirstArray', 6, 5); Кожна функція, оголошена в m-файлі
виконується у своїй робочій області, причому одні функції можуть викликати інші
функції й вкладеність таких викликів необмежена. Якщо в який-небудь із
виконуваних функцій відбудеться помилка, то обчислення перервуться й пам'ять,
виділена вкладеним функціям, повністю звільниться. Реалізовані практично всі
операції, які можна виконати над двомірними масивами. Ще один ключовий момент -
можливість гнучкого розширення функціональних можливостей системи. Мається на
увазі наступне: ви можете створювати цілі бібліотеки підпрограм в окремих
модулях, не торкаючи модуль ядра (Matrix.pas).
Як приклад ви можете переглянути модуль Signals.pas. У ньому перебуває кілька
функцій, які автор використовував у своїх розробках. Парсинг рядків виконується
вкрай рідко, а всі операції продумані дуже ретельно, тому Matrix є дуже швидкою
системою матричних обчислень.
}

                                                                              {
 Для використання системи Matrix.pas необхідно підключити модуль Matrix до
проекту й використовувати потрібні функції. Після підключення вам стає доступною
робоча область Base, яка створюється завжди (код створення перебуває в розділі
Initialization). Із системою поставляється документація, у якій дається
призначення кожної функції Matrix (коротке призначення дається в коментарях до
функції)
                                                                              }

{.$define CheckRange}

unit Matrix;

interface

uses Windows, SysUtils, Classes, Math, ComObj, Variants;

var
  {Визначає, потрібно чи сповіщати користувача про помилки (рекомендується)}
  MatrixShowErrorMessages: Boolean = True;

const
  MaxArrayCount = 1000000; // Максимальне число масивів у робочій області

{Конвертує рядок у число. Як роздільник може стояти як
 точка, так і кома}
function AnyStrToFloat(S: string): Extended;

{Конвертує число в рядок фіксованого формату (23 символу). У якості
 роздільника ставиться точка}
function FloatToStrExp(Value: Extended): string;

{Конвертує число в рядок. Довжина рядка змінна. Як роздільник
 ставиться точка}
function FloatToStrGen(Value: Extended): string;

{Передає команду Str}
procedure SendMatlabCommand(Str: string);

type    {Повідомляємо всі необхідні типи}
```

КбПЗ_2023

```
{Використовується для вказівки виміру масиву}
TDim = (dimRows, dimCols);

{Псевдоніми для стандартних числових типів}
TStandartType =
   // У першому рядку порядок елементів повинен бути фіксованим. Це потрібно
   // для правильної роботи функції GetElemType
  (stReal, stByte, stShortint, stWord, stShort, stDword, stInteger,
   stSmallint, stLongint, stCardinal, stLongword, stInt64, stDouble, stReal48,
   stSingle, stExtended, stComp, stCurrency);

{Опис масиву}
TMatrixData = record
  NameArray: string;
  Rows, Cols, Address, Hash: Integer;
end;

TNameArray = array[1..32] of Char;
TRealFunc = function(const Value: Real): Real;
TRealFunc2 = function(const Value1, Value2: Real): Real;
TExtendFunc = function(const Value: Extended): Extended;

{Клас "Список масивів". Для кожного масиву зберігає саму необхідну
 інформацію: Імені, Адресу, Число рядків, Число стовпців. Використовується
 винятково в класі TWorkspace }
TMatrixList = class(TObject)
private
  FList: array of ^TMatrixData;
  FCount: Integer;
  FCapacity: Integer;
  FData: ^TMatrixData;
  FLastIndexes: array[1..10] of Integer;
  function GetItemsName(Index: Integer): string;
  procedure SetItemsName(Index: Integer; const Value: string);
  function GetItemsCols(Index: Integer): Integer;
  procedure SetItemsCols(Index: Integer; const Value: Integer);
  function GetItemsRows(Index: Integer): Integer;
  procedure SetItemsRows(Index: Integer; const Value: Integer);
  function GetItemsAddr(Index: Integer): Integer;
  procedure SetItemsAddr(Index: Integer; const Value: Integer);
  procedure Grow;
  procedure SetCapacity(NewCapacity: Integer);
public
  {Деструктор}
  destructor Destroy(); override;

  {Додає елемент у список}
  function Add(Item: TMatrixData): Integer;

  {Фактично не видаляє, а встановлює префікс $ у рядків і обнуляє числа}
  procedure Delete(Index: Integer);

  {Використовується для видалення елементів, починаючи з кінця}
  procedure DeleteFull(Index: Integer);

  {Повністю очищає список і обнуляє довжину масиву покажчиків}
  procedure Clear;

  {Повертає число елементів списку}
  function Count: Integer;

  {Здійснює пошук масиву із заданим іменем. Виконує індексацію
   10 масивів для прискорення пошуку}
  function FindName(const ArrayName: string): Integer;

  {Повертає індекс масиву (у змінної Idx) OldName. Якщо масив не
   знайдено, то вертається індекс першого вилученого масиву (вилучені
   масиви позначені символом '$') або -1, якщо нічого не було знайдено}
  function FindOldName(const OldName: string; var Idx: Integer): Boolean;
```

КОПІЯ 2023

```
    {Визначає, чи існує масив, заданий індексом Idx}
    function ArrayExistsForIndex(const Idx: Integer): Boolean;

    {Установлює значення Index у перший осередок масиву FLastIndexes}
    procedure SetIndex(Index: Integer);

    {Наступні властивості забезпечують швидкий доступ до потрібного поля}
    property ItemsName[Index: Integer]: string read GetItemsName write
SetItemsName; default;
    property ItemsRows[Index: Integer]: Integer read GetItemsRows write
SetItemsRows;
    property ItemsCols[Index: Integer]: Integer read GetItemsCols write
SetItemsCols;
    property ItemsAddr[Index: Integer]: Integer read GetItemsAddr write
SetItemsAddr;
  end;
  {Клас TMatrixList закінчений}

  {Клас TWorkspace. Основний клас системи Matrix. Ви можете використовувати
   тільки функції, які оголошені в розділі PUBLIC, і їхнє призначення
   ви повинні знати й повинні вміти ними користуватися}
  TWorkspace = class(TObject)
  private
    FArrayList: TMatrixList;      // Список масивів даної робочої області
    FRefList: TList;              // Список масивів, переданих по посиланню
    FWorkspaceList: TList;        // Список дочірніх робочих областей
    FParentWorkspace: TWorkspace; // Якщо не NIL, то батько - робоча область
    FUniqueNameCounter: Int64;    // Лічильник унікальних імен
    FElemCount: Integer;          // Число елементів у робочій області
    FName: string;                // Імені робочої області
    FSelf: TWorkspace;            // Посилання на робочу область
    FMStream: TMemoryStream;      // Потік пам'яті для роботи з файлами

    {Процедури й функції, що здійснюють доступ до окремого елемента масиву}
    function GetElem(const Name: string; Row, Col: Integer): Real;
    procedure SetElem(const Name: string; Row, Col: Integer; const Value: Real);
    function GetElem(Idx, Row, Col: Integer): Real;
    procedure SetElem(Idx, Row, Col: Integer; const Value: Real);
    function GetElemFast(Adr, Row, Col, ColCount: Integer): Real;
    procedure SetElemFast(Adr, Row, Col, ColCount: Integer; const Value: Real);

    {Визначає, чи є масив з індексом Index переданим по посиланню}
    function FIsRef(Index: Integer): Boolean;

    {Витягає посилання на масив}
    function FGetRef(Name: string): TMatrixData;

    {Установлює посилання на масив}
    procedure FSetRef(Ref: TMatrixData);

    {Видаляє посилання на масив}
    procedure FDelRef(Name: string);

    {Записує зазначений масив у двійковий файл. Якщо AddToEnd=True,
     то масив запишеться в кінець файлу, інакше процедура буде перевіряти
     файл на наявність у ньому масиву NameAr}
    procedure FSaveArrayToBinFile(const FileName: string; ArIdx: Integer;
      AddToEnd: Boolean = True);

    {Зберігає робочу область у текстовому файлі}
    procedure FSaveToTextFile(FileName: string);

    {Зберігає зазначений масив у текстовий файл}
    procedure FSaveArrayToTextFile(const FileName: string; Name: string);

    {Завантажує робочу область із текстового файлу}
    procedure FLoadFromTextFile(const FileName: string);
```

КоПЗ-2023

```
    {Завантажує зазначений масив з текстового файлу}
    procedure FLoadArrayFromTextFile(const FileName: string; Name: string);

    {Очищає тимчасові робочі області й генерує виключення}
    procedure FDoError(Msg: string);
  public
    {Імені робочої області}
    property wsName: string read FName;

    {Посилання на дану робочу область (типізований Self)}
    property SelfWS: TWorkSpace read FSelf;

    {Посилання на батьківську робочу область}
    property ParentWS: TWorkSpace read FParentWorkspace;

    {Створює робочу область із іменем Name. Якщо ParentWorkspace заданий, то
     створена робоча область уважається тимчасовий, і буде автоматично
     знищуватися при знищенні ParentWorkspace}
    constructor Create(const Name: string; ParentWorkspace: TWorkspace = nil);
overload;

    {Знищує робочу область}
    destructor Destroy; override;

    {Генерує виключення (відбувається очищення стека й вивід текстового
повідомлення)
     Тимчасові робочі області каскадно знищуються доти, поки не буде
     знайдено основну робочу область, у якої ParentWorkspace=nil}
    procedure DoError(Msg: string);

    {Шукає масив "Name" у робочій області й повертає його індекс. Якщо масиву
     немає, то функція поверне значення -1}
    function Find(const Name: string): Integer;

    {Визначає, чи існує зазначений масив}
    function ArrayExists(const Name: string): Boolean; overload;

    {Визначає, чи існує масив із зазначеним індексом}
    function ArrayExists(Idx: Integer): Boolean; overload;

    {Створює масив "Name" розміром Rows x Cols. Забиває масив нулями, якщо
     параметр Init=True. Повертає індекс створеного масиву. Якщо масив
     є вектором, то виділяється додаткова пам'ять, що дозволяє швидко
     змінити розмір вектора за допомогою функції Resize}
    function NewArray(const Name: string; Rows, Cols: Integer;
      Init: Boolean = False): Integer; overload;

    {Створення нового вектора-рядка (Name=StartValue:Step:FinishValue)}
    function NewArray(const Name: string;
      StartValue, FinishValue: Real; Step: Extended): Integer; overload;

    {Повне очищення робочої області}
    procedure Clear; overload;

    {Видалення масиву "Name" з робочої області}
    function Clear(const Name: string): Integer; overload;

    {Властивості - для читання й запису елемента масиву по імені й індексу}
    property Elem[const Name: string; Row, Col: Integer]: Real read GetElem
write SetElem;
    property Elem[Idx, Row, Col: Integer]: Real read GetElem write SetElem;

    {Найшвидший спосіб доступу до елемента масиву. Необхідно додатково
     указувати адресу початку масиву й число стовпців у масиві}
    property ElemFast[Adr, Row, Col, ColCount: Integer]: Real read GetElemFast
write SetElemFast;

    {Установлює нове значення Value елемента масиву, заданого іменем.
     При AutoSize=True розміри масиву змінюються автоматично
```

КПЗ_2023

```
     При AutoCreate=True масив створюється, якщо він раніше не існував}
    procedure SetEl(const Name: string; const Value: Real; Row, Col: Integer;
      AutoSize: Boolean = False; AutoCreate: Boolean = False); overload;

    {Установлює нове значення елемента масиву, заданого індексом
     При AutoSize=True розміри масиву змінюються автоматично}
    procedure SetEl(Idx: Integer; const Value: Real; Row, Col: Integer;
      AutoSize: Boolean = False); overload;

    {Копіює масив SourAr у масив DestAr у межах даної робочої області}
    function CopyArray(const SourAr, DestAr: string): Integer; overload;

    {Копіює масив SourAr в DestAr з робочої області SourWS}
    function CopyArray(SourWS: TWorkspace; const SourAr, DestAr: string):
      Integer; overload;

    {Копіює посилання на масив із зазначеної робочої області. Тобто, фактично
     ви будете працювати з масивом SourAr, що перебуває в робочій області
     SourWS. Масив, переданий по посиланню, повинен використовуватися тільки для
     читання}
    function CopyRef(SourWS: TWorkspace; const SourAr, DestAr: string): Integer;

    {Переміщає масив у зазначену робочу область. Масив SourAr у робочої
     області DestWS буде мати імені DestAr}
    procedure MoveArray(DestWS: TWorkspace; const SourAr, DestAr: string);

    {Завантаження масиву в підпрограму з батьківської робочої області.
     Передача масиву може відбуватися по посиланню (ByRef=True) або за допомогою
     копіювання (ByRef=False). Переданий по посиланню масив повинен
використовуватися
     тільки для читання}
    function LoadArray(const SourAr, DestAr: string; ByRef: Boolean = True):
Integer;

    {Повертає результат роботи підпрограми (масив) у батьківську робочу
     область. Відбувається переміщення масиву, тому після роботи даної
     функції масив SourAr буде вилучений з даної робочої області}
    procedure ReturnArray(const SourAr, DestAr: string);

    {Перейменовує масив}
    function RenameArray(const SourAr, DestAr: string): Integer;

    {Транспонує матрицю (заміна рядків матриці її стовпцями)}
    function Transpose(const SourAr: string; DestAr: string): string;

    {Додає рядка масиву SourAr2 до масиву SourAr1. У результаті виходить
     масив DestAr. Дозволяється в якості одного із вхідних масивів указувати
     вихідний масив DestAr. Це дозволяє організувати додавання рядків до
     масиву DestAr у циклі, причому до початку циклу масив DestAr може
     не існувати}
    function AddRows(const SourAr1, SourAr2: string; DestAr: string): string;

    {Додає стовпці до масиву}
    function AddCols(const SourAr1, SourAr2: string; DestAr: string): string;

    {Вставляє рядки масиву RowsAr перед рядком Row масиву DestAr}
    function InsertRows(const RowsAr: string; DestAr: string; Row: Integer):
string;

    {Вставляє стовпці масиву ColsAr перед стовпцем Col масиву DestAr}
    function InsertCols(const ColsAr: string; DestAr: string; Col: Integer):
string;

    {Копіює або вирізає в масив DestAr рядка з масиву SourAr}
    function CopyCutRows(SourAr, DestAr: string;
      RowNumber, Rows: Integer; Cut: Boolean = False): string;

    {Копіює або вирізає в масив DestAr стовпці з масиву SourAr}
    function CopyCutCols(SourAr, DestAr: string;
```

Копія 2023

```
      ColNumber, Cols: Integer; Cut: Boolean = False): string;

    {Копіює зазначену частину масиву SourAr у масив DestAr}
    function CopySubmatrix(SourAr, DestAr: string;
      FirstRow, Rows, FirstCol, Cols: Integer): string;

    {Вставляє масив SourAr усередину масиву DestAr}
    function PasteSubmatrix(SourAr, DestAr: string;
      FirstPastRow, FirstPastCol: Integer): string;

    {Змінює розміри масиву (вихідний масив розглядається у вигляді вектора)}
    function Resize(const Name: string; Rows, Cols: Integer): Integer;

    {Змінює розміри матриці, зберігаючи внутрішню структуру}
    function ResizeMatrix(const Name: string; Rows, Cols: Integer): Integer;

    {Обчислює математичні функції з одним параметром.
     Приклад. Нехай є масив A = [1 2 3 4 5];
     Виконаємо зведення у квадрат: CalcFunc('A', 'A', fncSqr);
     Одержимо масив A = [1 4 9 16 25]
     Замість fncSqr можна вказувати інші функції, у тому числі й свої.
     Функція CalcFunc проганяє послідовно кожний елемент масиву SourAr
     через задану функцію}
    function CalcFunc(const SourAr: string; DestAr: string; Func: TRealFunc):
string; overload;
    function CalcFunc(const SourAr: string; DestAr: string; Func: TExtendFunc):
string; overload;

    {Функція, на відміну від попередньої, використовує два вхідних масиви.
     Нехай є вхідні масиви A і B. Їх можна скласти в такий спосіб:
     CalcFunc2('A', 'B', 'C', fncSum);
     Замість назви масиву ви можете вказати числове значення. Якщо ви
     вкажете змінну типу, відмінного від строкового й речовинного, те
     правильність роботи функції не гарантується. Функція зможе працювати
     тільки якщо обидва масиви мають однаковий розмір, або кожний з них
     містить тільки один елемент}
    function CalcFunc2(const SourArray1, SourArray2: Variant; DestAr: string;
      Func: TRealFunc2): string;

    {Заповнює масив випадковими числами від 0 до 1. Якщо розміри не зазначені,
     то використовується наявний масив. У противному випадку буде створений
     новий масив с розмірами Rows x Cols}
    function RandomAr(const Name: string; Rows: Integer = 0; Cols: Integer = 0):
string;

    {Заповнює масив діапазоном чисел із кроком Step}
    function FillAr(const Name: string; BaseValue: Real; Step: Extended):
string;

    {Створює масив і заповнює діапазоном чисел. Перше число - BaseValue, а
     інші виходять шляхом додатка значення Step}
    function NewFillAr(const Name: string; Rows, Cols: Integer;
      const BaseValue: Real; Step: Extended): string; overload;

    {Створює масив (вектор) і заповнює діапазоном чисел. Перше число -
BaseValue,
     останнє - LastValue. Крок розраховується залежно від кількості елементів}
    function NewFillAr(const Name: string; const FirstValue, LastValue: Real;
      ElemCount: Integer): string; overload;

    {Створює масив, всі елементи якого - нулі}
    function NewZeros(const Name: string; Rows, Cols: Integer): string;

    {Створює масив, всі елементи якого - одиниці}
    function NewOnes(const Name: string; Rows, Cols: Integer): string;

    {Перемножує масиви за правилом матричного множення}
    function MulMatrix(const Matrix1, Matrix2: string; MatResult: string):
string;
```

КПЗ – 2023

```
    {Визначає, чи рівні масиви між собою чи ні. Результат може бути
     записаний у масив ResArray (це "1" або "0")}
    function IsEqual(const Array1, Array2: string; ResArray: string = ''):
Boolean;

    {Якщо масив складається з одного елемента, то функція поверне True}
    function IsSingle(const Name: String): Boolean; overload;
    function IsSingle(Idx: Integer): Boolean; overload;

    {Визначає, чи є матриця квадратною}
    function IsSquare(const Name: String): Boolean; overload;
    function IsSquare(Idx: Integer): Boolean; overload;

    {Визначає, чи однакові розміри в масивів чи ні}
    function SizeEqual(const Array1, Array2: string): Boolean; overload;
    function SizeEqual(Idx1, Idx2: Integer): Boolean; overload;

    {Визначає мінімальні елементи (і при необхідності їхні індекси) в
     рядках або стовпцях матриці SourAr}
    procedure GetMin(SourAr, MinElems, Indexes: string;
      Dim: TDim = dimCols);

    {Визначає максимальні елементи (і при необхідності їхні індекси) в
     рядках або стовпцях матриці SourAr}
    procedure GetMax(SourAr, MaxElems, Indexes: string; Dim: TDim = dimCols);

    {Повертає величину найменшого й найбільшого елемента масиву Name}
    procedure GetMinMax(Name: string; var MinVal, MaxVal: Real); overload;

    {Повертає найбільші й найменші елементи кожного рядка або стовпця}
    function GetMinMax(SourAr, DestAr: string; Dim: TDim = dimRows): string;
overload;

    {Повертає мінімальний або максимальний елемент зазначеного рядка
     або стовпця масиву Name}
    function GetMinOrMax(Name: string; Num: Integer; Dim: TDim;
      ReturnMax: Boolean; var Index: Integer): Real;

    {Повертає середні арифметичні значення для стовпців або для рядків
     матриці SourAr}
    function GetMean(SourAr, DestAr: string; Dim: TDim = dimCols): string;

    {Множення елементів рядків або стовпців матриці}
    function CalcProd(SourAr, DestAr: string; Dim: TDim = dimRows): string;

    {Підсумує рядки масиву між собою}
    function SumRows(SourAr, DestAr: string): string;

    {Підсумує стовпці масиву між собою}
    function SumCols(SourAr, DestAr: string): string;

    {Завантажує масив з рядка, наприклад SLoad('A=[1 2 3; 4 5 6]');}
    function SLoad(const Str: string): string; overload;

    {Створює масив Name з елементами Values, наприклад SLoad('A', '[1 2 3]');}
    function SLoad(const Name, Values: string): string; overload;

    {Створює масив Name з елементом Value, наприклад SLoad('A', 100);}
    function SLoad(const Name: string; const Value: Real): string; overload;

    {Повертає розміри зазначеного масиву і його індекс - при бажанні}
    function GetSize(const Name: string; var Rows, Cols: Integer): Integer;
overload;

    {Повертає розміри зазначеного масиву і його індекс - при бажанні}
    function GetSize(Idx: Integer; var Rows, Cols: Integer): Integer; overload;

    {Повертає число елементів масиву}
```

КПІ-2023

```
function GetSize(const Name: string): Integer; overload;
function GetSize(Idx: Integer): Integer; overload;

{Повертає індекс масиву. Генерує виключення, якщо масиву немає}
function GetIndex(const Name: string): Integer;

{Повертає адреса масиву по імені у форматі Integer}
function GetAddress(const Name: string): Integer; overload;

{Повертає адреса масиву по індексу у форматі Integer}
function GetAddress(Idx: Integer): Integer; overload;

{Повертає імені масиву по його індексу}
function GetName(Idx: Integer): string; overload;

{Повертає число рядків}
function GetRows(const Name: string): Integer; overload;

{Повертає число рядків}
function GetRows(Idx: Integer): Integer; overload;

{Повертає число стовпців}
function GetCols(const Name: string): Integer; overload;

{Повертає число стовпців}
function GetCols(Idx: Integer): Integer; overload;

{Повертає список масивів даної робочої області}
procedure GetNameList(List: TStrings);

{Повертає кількість масивів}
function GetArrayCount(): Integer;

{Повертає загальне число елементів у робочій області}
function GetElemCount(): Integer;

{Перевіряє, чи правильно імені вихідного масиву. Якщо імені не задане,
 то може змінити його на "ans"}
function CheckResAr(var Name: string; DoANS: Boolean = True): string;

{Визначає, чи правильно задане імені масиву. Імені може мати довжину до 32
 символів. Першим символом повинна бути буква латинського алфавіту. Далі
 можуть стояти наступні символи: ['A'..'Z', 'a'..'z', '0'..'9', '_']}
function IsTrueName(const Name: string): Boolean;

{Зберігає масив у текстовий рядок}
function SaveArrayToString(const Name: string): string;

{Завантажує масив з текстового рядка}
function LoadArrayFromString(Str: string): string;

{Зберігає елементи масиву в рядок. Як роздільник рядків
 служить конструкція #13#10}
function SaveArrayToValues(const Name: string): string;

{Зберігає масив в ascii-файл}
procedure SaveToAscii(const FileName, Name: string; Fixed: Boolean = False);

{Завантажує масив з ascii-файлу}
procedure LoadFromAscii(const FileName: string);

{Зберігає робочу область або масив у текстовий файл}
procedure SaveToTextFile(const FileName: string; ArName: string = '');

{Завантажує робочу область або масив з текстового файлу}
procedure LoadFromTextFile(const FileName: string; ArName: string = '');

{Зберігає робочу область або масив у двійковий файл}
procedure SaveToBinFile(const FileName: string; ArName: string = '');
```

КПІ - 2023

```
    {Зчитує робочу область або масив із двійкового файлу}
    procedure LoadFromBinFile(const FileName: string; NameAr: string = '');

    {Копіює масив у зазначену область пам'яті}
    procedure SaveArrayToMemory(ArrayName: string; ArrayAddress: Pointer;
      ElemType: TStandartType);

    {Завантажує масив із зазначеної області пам'яті}
    procedure LoadArrayFromMemory(ArrayName: string; ArrayAddress: Pointer;
      RowCount, ColCount: Integer; ElemType: TStandartType);

    {Зберігає масив у потік (дочірній від TStream)}
    procedure SaveArrayToStream(ArrayName: string; Stream: TStream;
      ElemType: TStandartType; StartByte: Integer = 0);

    {Завантажує масив з потоку (дочірнього від TStream)}
    procedure LoadArrayFromStream(ArrayName: string; Stream: TStream;
      Rows, Cols: Integer; ElemType: TStandartType; StartByte: DWORD = 0);

    {Передає зазначений масив у MATLAB (в основну робочу область)}
    procedure PutArrayToMatlab(const Name: string);

    {Зчитує зазначений масив з основної робочої області MATLABa
     при помилці читання поверне False}
    function LoadArrayFromMatlab(const Name: string): Boolean;

    {Передача середовищу MATLAB команди Str}
    procedure SendMatlabCommand(Str: string);

    {Прикладів роботи даної функції зараз немає, однак вона допоможе вам у тому
     випадку, якщо ви хочете, щоб ваша підпрограма підтримувала одночасно
     і передачу масиву, заданого іменем і масиву, заданого рядком чисел.
     Тут InputArray - імені масиву, переданого в підпрограму,
     TempArray - імені тимчасового масиву. Логіка роботи функції така, що якщо
     InputArray - рядок чисел, то буде автоматично створений масив TempArray, в
     який завантажаться елементи рядка InputArray. Далі рядку InputArray буде
     привласнено імені тимчасового масиву TempArray.
     Таким чином, після роботи даної функції ви можете вважати InputArray
     іменем вхідного масиву й працювати з ним як ні в чому не бувало}
    function HandInput(var InputArray: string; var TempArray: string): Integer;

    {Генерація унікального імені масиву}
    function GenName(Add: string = ''): string;

    {Повертає тип елементів масиву, закодований у вигляді числа}
    function GetElemType(const Name: string): TStandartType;
  end;


var
  {Тут оголошена робоча область Base. Ви не повинні створювати або знищувати її.
   Як тільки ви підключите даний модуль до свого проекту, вам відразу стає
   доступною робоча область Base. Змінна оголошена винятково для вашого
   зручності}
  Base: TWorkspace;


// Наступні функції використовуються для передачі у функцію CalcFunc2

function fncSum(const Value1, Value2: Real): Real;   // Додавання
function fncSub(const Value1, Value2: Real): Real;   // Вирахування
function fncMul(const Value1, Value2: Real): Real;   // Множення
function fncDiv(const Value1, Value2: Real): Real;   // Ділення
function fncPower(const Value1, Value2: Real): Real; // Зведення в ступінь

function fncEQ(const Value1, Value2: Real): Real; // Дорівнює
function fncNE(const Value1, Value2: Real): Real; // Не дорівнює
function fncLT(const Value1, Value2: Real): Real; // Менше
```

КПЗ-2023

```
function fncGT(const Value1, Value2: Real): Real; // Більше
function fncLE(const Value1, Value2: Real): Real; // Менше або дорівнює
function fncGE(const Value1, Value2: Real): Real; // Більше або дорівнює

function fncAnd(const Value1, Value2: Real): Real; // I
function fncOr(const Value1, Value2: Real): Real; // Або
function fncXor(const Value1, Value2: Real): Real; // Xor

// Наступні функції використовуються для передачі у функцію CalcFunc

function fncNot(const Value: Real): Real; // Not

function fncNone(const Value: Real): Real; // Result := Value
function fncSin(const Value: Real): Real;
function fncCos(const Value: Real): Real;
function fncRound(const Value: Real): Real;
function fncTrunc(const Value: Real): Real;
function fncInt(const Value: Real): Real;
function fncFrac(const Value: Real): Real;
function fncSqrt(const Value: Real): Real;
function fncSqr(const Value: Real): Real;
function fncAbs(const Value: Real): Real;
function fncExp(const Value: Real): Real;

// Наступні змінні використовуються для передачі у функцію CalcFunc
var
         fncTan: TExtendFunc = Tan;
         fncCotan: TExtendFunc = Cotan;
         fncArcsin: TExtendFunc = Arcsin;
         fncArccos: TExtendFunc = Arccos;
         fncSecant: TExtendFunc = Secant;
         fncCosecant: TExtendFunc = Cosecant;
         fncLog10: TExtendFunc = Log10;
         fncLog2: TExtendFunc = Log2;
         fncLnXP1: TExtendFunc = LnXP1;
         fncCosh: TExtendFunc = Cosh;
         fncSinh: TExtendFunc = Sinh;
         fncTanh: TExtendFunc = Tanh;
         fncCot: TExtendFunc = Cot;
         fncSec: TExtendFunc = Sec;
         fncCsc: TExtendFunc = Csc;
         fncArcCot: TExtendFunc = ArcCot;
         fncArcSec: TExtendFunc = ArcSec;
         fncArcCsc: TExtendFunc = ArcCsc;
         fncArcCosh: TExtendFunc = ArcCosh;
         fncArcSinh: TExtendFunc = ArcSinh;
         fncArcTanh: TExtendFunc = ArcTanh;
         fncArcCot: TExtendFunc = ArcCot;
         fncArcSec: TExtendFunc = ArcSec;
         fncArcCsc: TExtendFunc = ArcCsc;

const
  // Числові константи
  matrixPi = Pi;
  matrixExp = 2.7182818284590455;
  matrixEps = 1.0842021724855 E-19;

  // Текстові константи
  matBaseWorkspace = 'Основна робоча область';
  matWrongIndex = 'Масив із зазначеним індексом не існує';
  matWrongCapacity = 'Перевищене максимально припустиме число масивів у робочій
області';
  matArrayNotFound = 'Зазначений масив відсутній у робочій області';
  matBadCoords = 'Зазначені неприпустимі координати елемента';
  matBadRow = 'Зазначеного рядка не існує';
  matBadCol = 'Зазначеного стовпця не існує';
  matBadRowCol = 'Зазначеного стовпця/рядка не існує';
  matBadSize = 'Розміри масиву задані невірно';
  matBadInputData = 'Неприпустимі вхідні дані';
```

КСПЗ_2023

```
  matErrorCreateArray = 'Неможливо створити масив із зазначеними розмірами';
  matIsNotVektor = 'Масив не є вектором';
  matDivsionByZero = 'Ділення на нуль';
  matBadOperation = 'Задана операція не підтримується';
  matCalcFuncError = 'Відбулася помилка при розрахунку функції';
  matErrorLoadingArray = 'Помилка при спробі завантажити масив';
  matErrorPast = 'Неможливо вставити масив сам у себе';
  matInvalidFloat = '"%s" не є правильним дробовим числом';
  matOutOfMemory = 'Недостатньо пам'яті для виконання операції';
  matBadName = 'Зазначене неприпустиме імені масиву';
  matDirNotFound = 'Зазначена директорія не знайдена';
  matBadType = 'Зазначений тип не підтримується';
  matArraysNotAgree = 'Розміри масивів не погоджені';
  matBadParams = 'Неприпустимі параметри створення масиву';
  matBadFileFormat = 'Неправильний формат файлу ';
  matFileNotFound = 'Файл не знайдений';
  matBinaryHeader = 'Matrix '; // 30 байт
  matWorkspaceError = 'Робоча область задана помилково';
  matReturnError = 'Спроба повернення масиву, переданого по посиланню';
  matCalcLimit =
    'Генератор імен масивів досяг останнього можливого значення.' +
    'Тепер він буде обнулений. При поганій організації обчислень це може
привести' +
    'до помилок. Ви хочете продовжити роботу програми?';

implementation

var
  Matlab: Variant;
  MatlabArray, MatlabVar: OleVariant;

function fncSum(const Value1, Value2: Real): Real;
begin
  Result := Value1 + Value2;
end;

function fncSub(const Value1, Value2: Real): Real;
begin
  Result := Value1 - Value2;
end;

function fncMul(const Value1, Value2: Real): Real;
begin
  Result := Value1 * Value2;
end;

function fncDiv(const Value1, Value2: Real): Real;
begin
  Result := Value1 / Value2;
end;

function fncPower(const Value1, Value2: Real): Real;
begin
  Result := Power(Value1, Value2);
end;

function fncEQ(const Value1, Value2: Real): Real;
begin
  Result := Integer(Value1 = Value2);
end;

function fncNE(const Value1, Value2: Real): Real;
begin
  Result := Integer(Value1 <> Value2);
end;

function fncLT(const Value1, Value2: Real): Real;
begin
  Result := Integer(Value1 < Value2);
```

КОПЗ – 2023

```
end;

function fncGT(const Value1, Value2: Real): Real;
begin
  Result := Integer(Value1 > Value2);
end;

function fncLE(const Value1, Value2: Real): Real;
begin
  Result := Integer(Value1 <= Value2);
end;

function fncGE(const Value1, Value2: Real): Real;
begin
  Result := Integer(Value1 >= Value2);
end;

function fncAnd(const Value1, Value2: Real): Real;
begin
  Result := Integer((Value1 <> 0) and (Value2 <> 0));
end;

function fncOr(const Value1, Value2: Real): Real;
begin
  Result := Integer((Value1 <> 0) or (Value2 <> 0));
end;

function fncXor(const Value1, Value2: Real): Real;
begin
  Result := Integer((Value1 <> 0) xor (Value2 <> 0));
end;

function fncNot(const Value: Real): Real;
begin
  Result := Integer(not (Value <> 0));
end;

function fncNone(const Value: Real): Real;
begin
  Result := Value;
end;

function fncSin(const Value: Real): Real;
begin
  Result := Sin(Value);
end;

function fncCos(const Value: Real): Real;
begin
  Result := Cos(Value);
end;

function fncRound(const Value: Real): Real;
begin
  Result := Round(Value);
end;

function fncTrunc(const Value: Real): Real;
begin
  Result := Trunc(Value);
end;

function fncInt(const Value: Real): Real;
begin
  Result := Int(Value);
end;

function fncFrac(const Value: Real): Real;
begin
```

КоПЗ_2023

```
  Result := Frac(Value);
end;

function fncSqrt(const Value: Real): Real;
begin
  Result := Sqrt(Value);
end;

function fncSqr(const Value: Real): Real;
begin
  Result := Sqr(Value);
end;

function fncAbs(const Value: Real): Real;
begin
  Result := Abs(Value);
end;

function fncExp(const Value: Real): Real;
begin
  Result := Exp(Value);
end;

procedure MsGrow(Ms: TMemoryStream);
var
  Pos: Integer;
begin
  if (Ms.Position < Ms.Size - 1) or (Ms.Size < 10000) then Exit;
  Pos := Ms.Position;
  // Збільшуємо розмір потоку на 20 %
  Ms.SetSize(Ms.Size + Ms.Size div 5);
  Ms.Position := Pos;
end;

function StrToName(S: string): TNameArray;
var
  I, J: Integer;
begin
  FillChar(Result, 32, 0);
  if S <> '' then
  begin
    I := Min(32, Length(S));
    for J := 1 to I do
      Result[J] := S[J];
  end;
end;

function AvailMemory: Real;
var
  MemInfo: TMemoryStatus;
begin
  MemInfo.dwLength := Sizeof(MemInfo);
  GlobalMemoryStatus(MemInfo);
  Result := MemInfo.dwAvailPhys + MemInfo.dwAvailPageFile;
end;

function FileSize(FileName: string): Integer;
var
  F: HFile;
begin
  Result := 0;
  if not FileExists(FileName) then Exit;
  F := _lopen(PChar(FileName), OF_READ);
  Result := _llseek(F, 0, FILE_END);
  _lclose(F);
end;

function MatrixData(const NameArray: string;
  const Rows, Cols, Address: Integer): TMatrixData;
```

КБПЗ_2023

```
begin
  Result.NameArray := NameArray;
  Result.Rows := Rows;
  Result.Cols := Cols;
  Result.Address := Address;
  Result.Hash := 0;
end;

function AnyStrToFloat(S: string): Extended;
var
  C: Char;
  I: Byte;
begin
  if DecimalSeparator = '.' then C := ',' else C := '.';
  I := Pos(C, S);
  if I > 0 then S[I] := DecimalSeparator;
  if not TextToFloat(PChar(S), Result, fvExtended) then
    raise Exception.CreateFmt(matInvalidFloat, [S]);
end;

function FloatToStrExp(Value: Extended): string;
var
  Buffer: array[0..63] of Char;
  I: Byte;
begin
  SetString(Result, Buffer, FloatToText(Buffer, Value, fvExtended,
    ffExponent, 17, 3));
  I := Pos(',', Result);
  if I > 0 then Result[I] := '.';
end;

function FloatToStrGen(Value: Extended): string;
var
  Buffer: array[0..63] of Char;
  I: Byte;
begin
  SetString(Result, Buffer, FloatToText(Buffer, Value, fvExtended,
    ffGeneral, 15, 0));
  I := Pos(',', Result);
  if I > 0 then Result[I] := '.';
end;

function GenerateHash(const Str: string): Integer;
var
  Len, I: Integer;
  Flag: Boolean;
begin
  Result := 0;
  if Str = '' then Result := -1 else
    if Str[1] = '$' then Result := 0 else
    begin
      Len := Length(Str);
      Flag := False;
      for I := 1 to Len do
      begin
        if Flag then
          Result := Result + not Ord(Str[I])
        else
          Result := Result + Ord(Str[I]);
        Flag := not Flag;
      end;
    end;
end;

procedure SendMatlabCommand(Str: string);
begin
  Matlab := CreateOleObject('Matlab.Application');
  Matlab.Execute(Str);
end;
```

КоПЗ - 2023

```
function TMatrixList.Add(Item: TMatrixData): Integer;
begin
  if FCount > MaxArrayCount then
    raise Exception.Create(matWrongCapacity);
  Result := FCount;
  if Result = FCapacity then Grow;
  New(FData);
  FData^ := Item;
  FData^.Hash := GenerateHash(FData^.NameArray);
  Inc(FCount);
  FList[FCount - 1] := Pointer(FData);
end;

function TMatrixList.ArrayExistsForIndex(const Idx: Integer): Boolean;
begin
  Result := False;
  if (Idx < 0) or (Idx >= FCount) or (FList[Idx]^.NameArray[1] = '$') then Exit;
  Result := True;
end;

procedure TMatrixList.Clear;
var
  I: Integer;
begin
  if FCount > 0 then
    for I := FCount - 1 downto 0 do
      DeleteFull(I);
  FCount := 0; // Про всякий випадок
  // Довжину списку не обнуляємо, тому що ефективності це не підвищить
end;

function TMatrixList.Count: Integer;
begin
  Result := FCount;
end;

procedure TMatrixList.Delete(Index: Integer);
begin
  if (Index < 0) or (Index >= FCount) then
    raise Exception.Create(matWrongIndex);
  with FList[Index]^ do begin
    NameArray := '$' + NameArray;
    Hash := 0;
    Rows := 0;
    Cols := 0;
    Address := 0;
  end;
end;

procedure TMatrixList.DeleteFull(Index: Integer);
begin
  if (Index <> FCount - 1) then
    raise Exception.Create('TMatrixList: You must delete only last elem!');
  if FList[Index] = nil then Exit;
  FList[Index]^.NameArray := '';
  FData := Pointer(FList[Index]);
  Dispose(FData);
  FList[Index] := nil;
  Dec(FCount);
end;

destructor TMatrixList.Destroy;
begin
  Clear;
  FList := nil;
  inherited Destroy;
end;
```

КСПЗ_2023

```
function TMatrixList.FindName(const ArrayName: string): Integer;
var
  I, J, Buf, Hash: Integer;
begin
  Hash := GenerateHash(ArrayName);
  Result := -1;
  if FCount = 0 then Exit;
  // Шукаємо спочатку в масиві FLastIndexes
  for I := 1 to 10 do begin
    if (FLastIndexes[I] < FCount) and (FList[FLastIndexes[I]].Hash = Hash) and
       (FList[FLastIndexes[I]].NameArray = ArrayName) then
      begin
        Result := FLastIndexes[I];
        if I > 1 then
        begin // Переміщає індекс до початку масиву на одну позицію
          Buf := FLastIndexes[I - 1];
          FLastIndexes[I - 1] := FLastIndexes[I];
          FLastIndexes[I] := Buf;
        end;
        Exit;
      end;
  end;
  // Якщо не знайшли в FLastIndexes, то шукаємо в самому списку
  for I := 0 to FCount - 1 do begin
    if (FList[I]^.Hash = Hash) and (FList[I]^.NameArray = ArrayName) then
    begin
      Result := I;
      for J := 9 downto 1 do
        FLastIndexes[J + 1] := FLastIndexes[J];
      FLastIndexes[1] := I;
      Exit;
    end;
  end;
end;

function TMatrixList.FindOldName(const OldName: string;
  var Idx: Integer): Boolean;
var
  I: Integer;
begin
  Result := False;
  Idx := -1;
  if FCount = 0 then Exit;
  for I := 0 to FCount - 1 do
    if (FList[I]^.Hash = 0) and (FList[I]^.NameArray[1] = '$') then
    begin
      if Idx = -1 then Idx := I;
      if FList[I]^.NameArray = OldName then
      begin
        Idx := I;
        Result := True;
        Exit;
      end;
    end;
end;

function TMatrixList.GetItemsAddr(Index: Integer): Integer;
begin
  Result := FList[Index]^.Address;
end;

function TMatrixList.GetItemsCols(Index: Integer): Integer;
begin
  Result := FList[Index]^.Cols;
end;

function TMatrixList.GetItemsName(Index: Integer): string;
begin
  Result := FList[Index]^.NameArray;
```

КбПЗ_2023

```
end;

function TMatrixList.GetItemsRows(Index: Integer): Integer;
begin
  Result := FList[Index]^.Rows;
end;

procedure TMatrixList.Grow;
begin
  if FCapacity < 50 then SetCapacity(50) else
    // Збільшує розмір списку на 20 %
    SetCapacity(FCapacity + FCapacity div 5);
end;

procedure TMatrixList.SetCapacity(NewCapacity: Integer);
begin
  if (NewCapacity < FCount) or (NewCapacity > MaxArrayCount) then
    raise Exception.Create(matWrongCapacity);
  if NewCapacity <> FCapacity then
  begin
    SetLength(FList, NewCapacity);
    FCapacity := NewCapacity;
  end;
end;

procedure TMatrixList.SetIndex(Index: Integer);
var
  I: Integer;
begin
  for I := 1 to 10 do if FLastIndexes[I] = Index then Exit;
  for I := 9 downto 1 do FLastIndexes[I + 1] := FLastIndexes[I];
  FLastIndexes[1] := Index;
end;

procedure TMatrixList.SetItemsAddr(Index: Integer; const Value: Integer);
begin
  FList[Index]^.Address := Value;
end;

procedure TMatrixList.SetItemsCols(Index: Integer; const Value: Integer);
begin
  FList[Index]^.Cols := Value;
end;

procedure TMatrixList.SetItemsName(Index: Integer; const Value: string);
begin
  FList[Index]^.NameArray := Value;
  FList[Index]^.Hash := GenerateHash(Value);
end;

procedure TMatrixList.SetItemsRows(Index: Integer; const Value: Integer);
begin
  FList[Index]^.Rows := Value;
end;

function TWorkspace.Resize(const Name: string; Rows, Cols: Integer): Integer;
var
  C, OldSize, NewSize, RC, CC, I, I1: Integer;
  Tmp: string;
begin
  I := Find(Name);
  if I = -1 then begin
    Result := NewArray(Name, Rows, Cols, True);
    Exit;
  end;
  GetSize(I, RC, CC);
  OldSize := RC * CC;
  NewSize := Rows * Cols;
  // Якщо оброблюваний масив є вектором, по намагаємося застосувати
```

КОПЗ_2023

```
  // що залишилася виділена для даного масиву пам'ять
  if (((RC = 1) and (CC > 1)) or ((RC > 1) and (CC = 1))) and
     (NewSize div 100 = OldSize div 100) then
  begin
    // Коректуємо число елементів у робочій області
    FElemCount := FElemCount + NewSize - OldSize;
    FArrayList.ItemsRows[I] := Rows;
    FArrayList.ItemsCols[I] := Cols;
    // Якщо брали додаткову пам'ять, то обнуляємо її, починаючи з OldSize
    if NewSize > OldSize then
      FillChar(Pointer(FArrayList.ItemsAddr[I] + OldSize * 8)^, (NewSize -
OldSize) * 8, 0);
    Result := I;
    Exit;
  end;
  C := Min(NewSize, OldSize) * 8;
  Tmp := GenName();
  I1 := NewArray(Tmp, Rows, Cols, True);
  Move(Pointer(FArrayList.ItemsAddr[I])^, Pointer(FArrayList.ItemsAddr[I1])^,
C);
  Result := RenameArray(Tmp, Name);
end;

procedure TWorkspace.Clear;
var
  I: Integer;
begin
  FElemCount := 0;
  FUniqueNameCounter := 0;
  with FArrayList do begin
    if Count = 0 then Exit;
    for I := 0 to Count - 1 do
    try
      if not FIsRef(I) then
        FreeMem(Pointer(ItemsAddr[I]));
    except
      // Мовчачи ігноруємо дане виключення
    end;
    Clear;
  end;
  if FMStream <> nil then FMStream.Clear;
end;

function TWorkspace.Clear(const Name: string): Integer;
var
  I, C, Rows, Cols: Integer;
begin
  Result := Find(Name);
  if Result = -1 then Exit;
  if Name[1] = '$' then Exit;
  if FIsRef(Result) then FDelRef(Name) else
  begin
    GetSize(Result, Rows, Cols);
    Dec(FElemCount, Rows * Cols);
  end;

  try
    FreeMem(Pointer(FArrayList.ItemsAddr[Result]));
  except
    // Видалення неіснуючого масиву
    FArrayList.Delete(Result);
    Exit;
  end;
  // Відзначаємо масив як вилучений
  FArrayList.Delete(Result);

  // Видаляємо старі імена, якщо вони усе ще перебувають у робочій області
  C := FArrayList.Count;
  if C > Result + 1 then
```

КбПЗ_2023

```
    for I := C - 1 downto Result + 1 do
    begin
      if FArrayList.ItemsName[I][1] <> '$' then Break else
        FArrayList.DeleteFull(I);
    end;
end;

constructor TWorkspace.Create(const Name: string; ParentWorkspace: TWorkspace =
nil);
begin
  inherited Create;
  FArrayList := TMatrixList.Create;
  FName := Name;
  FRefList := TList.Create;
  FWorkspaceList := TList.Create;
  FSelf := Self;
  // Додає посилання на робочу область до батьківської робочої області
  if ParentWorkspace <> nil then
  begin
    ParentWorkspace.FWorkspaceList.Add(Self);
    FParentWorkspace := ParentWorkspace;
  end;
end;

destructor TWorkspace.Destroy;
var
  I: Integer;
begin
  // Видаляємо посилання з батька
  if FParentWorkspace <> nil then
    FParentWorkspace.FWorkspaceList.Remove(Self);

  // Видаляємо всі дочірні робочі області
  if FWorkspaceList.Count > 0 then
    for I := FWorkspaceList.Count - 1 downto 0 do
      TWorkspace(FWorkspaceList.Items[I]).Free;

  Clear;
  FWorkspaceList.Free;
  FRefList.Free;
  FArrayList.Free;
  if FMStream <> nil then FMStream.Free;
  inherited;
end;

procedure TWorkspace.DoError(Msg: string);
begin
  Msg := wsName + ':'#13#10#13#10 + Msg;
  FDoError(Msg);
end;

procedure TWorkspace.FDoError(Msg: string);
var
  I: Integer;
begin
  // Знищуємо всі дочірні робочі області
  if FWorkspaceList.Count > 0 then
    for I := FWorkspaceList.Count - 1 downto 0 do
      TWorkspace(FWorkspaceList.Items[I]).Free;

  // Якщо дійшли до головної робочої області, то генеруємо помилку
  if FParentWorkspace = nil then
  begin
    if MatrixShowErrorMessages then
      raise Exception.Create(Msg)
    else
      Abort;
  end else
    FParentWorkspace.FDoError(Msg);
```

КОПЗ 2023

```
end;

function TWorkspace.Find(const Name: string): Integer;
begin
  Result := FArrayList.FindName(Name);
end;

function TWorkspace.GetAddress(const Name: string): Integer;
begin
  Result := FArrayList.ItemsAddr[GetIndex(Name)];
end;

function TWorkspace.GetIndex(const Name: string): Integer;
begin
  Result := FArrayList.FindName(Name);
  if Result = -1 then DoError(matArrayNotFound);
end;

function TWorkspace.GetSize(const Name: string; var Rows, Cols: Integer):
Integer;
begin
  Result := Find(Name);
  if Result = -1 then DoError(matArrayNotFound);
  Rows := FArrayList.ItemsRows[Result];
  Cols := FArrayList.ItemsCols[Result];
end;

function TWorkspace.LoadArrayFromString(Str: string): string;
var
  I, Len, NextPos, Cols, Rows, ElCount, Adr: Integer;
  ArName, S: string;
  EquExists, ColsKnown: Boolean;
  R: Real;
  E: Extended;
begin
  if FMStream = nil then FMStream := TMemoryStream.Create;
  if Length(Str) < 3 then DoError(matErrorLoadingArray);
  if (Str[Length(Str) - 1] <> ']') and (Str[Length(Str)] <> ']') then
    Insert(']', Str, Length(Str) + 1);
  Len := Length(Str);
  NextPos := 0;

  // Визначаємо ім'я масиву
  for I := 1 to Len do
  begin
    if (ArName <> '') and (not (Str[I] in ['A'..'Z', 'a'..'z', '0'..'9', '_']))
then
    begin
      NextPos := I;
      Break;
    end;
    if (Str[I] in ['A'..'Z', 'a'..'z']) and (ArName = '') then
    begin
      ArName := Str[I];
      Continue;
    end;
    if (Str[I] in ['A'..'Z', 'a'..'z', '0'..'9', '_']) and (ArName <> '') then
      ArName := ArName + Str[I];
  end;

  if ArName = '' then DoError(matErrorLoadingArray);
  Result := ArName;

  // Шукаємо символ '='
  EquExists := False;
  for I := NextPos to Len do
    if Str[I] = '=' then
    begin
      EquExists := True;
```

КОПІЯ 2023

```
      NextPos := I + 1;
      Break;
    end;

  if not EquExists then DoError(matErrorLoadingArray);
  ColsKnown := False;
  ElCount := 0;
  Cols := 0;
  FMStream.Clear;

  // Зчитуємо елементи масиву
  for I := NextPos to Len do
  begin
    if Str[I] in ['0'..'9', '.', '-', '+', 'e', 'E'] then
    begin
      S := S + Str[I];
      Continue;
    end else
    begin
      if S <> '' then
      begin
        E := AnyStrToFloat(S);
        S := '';
        if E < -MaxDouble then E := -MaxDouble;
        if E >  MaxDouble then E :=  MaxDouble;
        R := E;
        MsGrow(FMStream);
        FMStream.Write(R, 8);
        if not ColsKnown then Inc(Cols);
        Inc(ElCount);
      end;

      if Str[I] = ']' then Break;
      if Str[I] = ';' then ColsKnown := True;
    end
  end;
  // Копіює елементи в масив
  if FMStream.Size = 0 then DoError(matErrorLoadingArray);
  if (ElCount mod Cols) <> 0 then DoError(matErrorLoadingArray);

  Rows := ElCount div Cols;
  Adr := GetAddress(NewArray(ArName, Rows, Cols));
  FMStream.Seek(0, soFromBeginning);
  FMStream.Read(Pointer(Adr)^, ElCount * 8);
  FMStream.Clear;
end;

function TWorkspace.NewArray(const Name: string; Rows, Cols: Integer;
  Init: Boolean): Integer;
var
  f, InitCnt, Idxold: Integer;
  eRows, eCols, Size: Integer;
  P: Pointer;
begin
  eRows := 0;
  eCols := 0;
  if (Rows < 1) or (Cols < 1) or (Int64(Rows) * Cols * 8 > High(Integer)) then
    DoError(matErrorCreateArray);
  if not IsTrueName(Name) then DoError(matBadName);
  // Якщо змінна з таким іменем є й має ті ж розміри, то
  // використовуємо її
  f := Find(Name);
  Size := Rows * Cols;
  if f <> -1 then begin
    GetSize(f, eRows, eCols);
    if ((eRows = Rows) and (eCols = Cols)) or
       (((eRows * eCols) = Size) and (Rows <> 1) and (Cols <> 1)) then
    begin
      P := Pointer(GetAddress(f));
```

КПВ_2023

```
      if Init then FillChar(P^, Size shl 3, 0);
      Result := f;
      FArrayList.ItemsRows[f] := Rows;
      FArrayList.ItemsCols[f] := Cols;
      Exit;
    end else
      Clear(Name);
  end;
  FArrayList.FindOldName('$' + Name, Idxold);
  InitCnt := Size * 8;
  if InitCnt > AvailMemory then DoError(matOutOfMemory);
  // Якщо масив - вектор, то ініціалізуємо пам'ять кратною 100
  if ((Rows = 1) and (Cols > 1)) or ((Rows > 1) and (Cols = 1)) then
    GetMem(P, (Size div 100 + 1) * 100 * 8)
  else
    GetMem(P, Size shl 3);

  Inc(FElemCount, Size);

  if Idxold = -1 then
    FArrayList.Add(MatrixData(Name, Rows, Cols, Integer(P)))
  else begin
    FArrayList.ItemsName[Idxold] := Name;
    FArrayList.ItemsRows[Idxold] := Rows;
    FArrayList.ItemsCols[Idxold] := Cols;
    FArrayList.ItemsAddr[Idxold] := Integer(P);
  end;
  if Idxold <> -1 then Result := Idxold else Result := FArrayList.Count - 1;
  FArrayList.SetIndex(Result);
  if Init then FillChar(P^, InitCnt, 0);
end;

function TWorkspace.SaveArrayToString(const Name: string): string;
var
  I, J, K, Rows, Cols, Idx: Integer;
  Len, L2, N: Integer;
  S: string;
begin
  Idx := GetSize(Name, Rows, Cols);
  Result := #13#10 + Name + '=' + '[';
  N := Length(Result);
  // Установлюємо максимально можливу довжину рядка
  Len := Rows * (Cols + 1) * 25 + 100;
  SetLength(Result, Len);
  for I := 1 to Rows do
  begin
    for J := 1 to Cols do begin
      S := FloatToStrGen(Elem[Idx, I, J]);
      L2 := Length(S);
      for K := 1 to L2 do
        Result[N + K] := S[K];
      Inc(N, L2 + 1);
     Result[N] := ' ';
    end;
    Dec(N);
    Result[N + 1] := ';';
    Result[N + 2] := #13;
    Result[N + 3] := #10;
    Inc(N, 3);
  end;
  Result[N - 2] := ']';
  Result[N - 1] := ';';
  Result[N + 0] := #13;
  Result[N + 1] := #10;
  SetLength(Result, N + 1);
end;

procedure TWorkspace.SetEl(const Name: string; const Value: Real; Row, Col:
Integer;
```

КПЗ_2023

```
  AutoSize: Boolean = False; AutoCreate: Boolean = False);
var
  I: Integer;
  Rows, Cols, NewRows, NewCols: Integer;
begin
  if (Row < 1) or (Col < 1) then DoError(matBadCoords);
  I := Find(Name);
  if AutoCreate and (I = -1) then I := NewArray(Name, Row, Col, True);
  GetSize(I, Rows, Cols);
  if (Row > Rows) or (Col > Cols) then
  begin
    if not AutoSize then DoError(matBadCoords) else
      begin
        NewRows := Max(Row, Rows);
        NewCols := Max(Col, Cols);
        if (NewRows = 1) or (NewCols = 1) then
          I := Resize(Name, NewRows, NewCols)
        else
          I := ResizeMatrix(Name, NewRows, NewCols);
      end;
  end;
  Real(Pointer((FArrayList.ItemsAddr[I] + ((Row - 1) * Cols + Col) shl 3 - 8))^)
:= Value;
end;

function TWorkspace.NewArray(const Name: string; StartValue,
  FinishValue: Real; Step: Extended): Integer;
var
  I, Mem: Integer;
  ColCount: Integer;
  ETemp: Extended;
  Adr: Integer;
  eps1, eps2: Real;
  P: Pointer;
begin
  if not IsTrueName(Name) then DoError(matBadName);
  if (Step = 0) or ((FinishValue - StartValue) / Step < 0) then
DoError(matBadParams);

  // Визначаємо кількість стовпців у масиві
  ColCount := 0;
  ETemp := StartValue;
  eps1 := abs(Step) / 2; // Для запобігання зациклення
  eps2 := matrixEps;  // Машинне епсілон

  if StartValue <= FinishValue then
    while ETemp <= FinishValue + eps1 do
    begin
      Inc(ColCount);
      ETemp := ETemp + Step;
      if (ETemp > FinishValue) and
        (abs((ETemp - FinishValue) / FinishValue) > eps2) then Break;
    end else
    while ETemp >= FinishValue - eps1 do
    begin
      Inc(ColCount);
      ETemp := ETemp + Step;
      if (ETemp < FinishValue) and
        (abs((FinishValue - ETemp) / FinishValue) > eps2) then Break;
    end;
  if ColCount = 0 then DoError(matBadCoords);
  // Якщо змінна з таким іменем уже є, то знищуємо її:
  Clear(Name);
  if (ColCount < 1) or (Int64(ColCount) * 8 > High(Integer)) then
DoError(matErrorCreateArray);
  if AvailMemory() < (Int64(ColCount) * 8) then DoError(matOutOfMemory);
  // Шукаємо, немає чи в списку змінною з іменем '$'+Name
  FArrayList.FindOldName('$' + Name, Mem);
  // Якщо масив - вектор, то ініціалізуємо пам'ять кратною 100
```

КПЗ - 2023

```
  if ColCount > 1 then
    GetMem(P, (ColCount div 100 + 1) * 100 * 8)
  else
    GetMem(P, ColCount * 8);

  Inc(FElemCount, ColCount);

  if Mem = -1 then
    FArrayList.Add(MatrixData(Name, 1, ColCount, Integer(P)))
  else
    with FArrayList do begin
      ItemsName[Mem] := Name;
      ItemsRows[Mem] := 1;
      ItemsCols[Mem] := ColCount;
      ItemsAddr[Mem] := Integer(P);
    end;

  if Mem <> -1 then Result := Mem else Result := FArrayList.Count - 1;
  FArrayList.SetIndex(Result);
  Adr := Integer(P);
  ETemp := StartValue;
  for I := 1 to ColCount do begin
    Real(Pointer(Adr)^) := ETemp;
    Inc(Adr, 8);
    ETemp := ETemp + Step;
  end;
end;

procedure TWorkspace.SaveToAscii(const FileName, Name: string;
  Fixed: Boolean = False);
var
  FullFileName: string;
  I, J, Rows, Cols, Adr: Integer;
  TF: TextFile;
  Str, Path, fName: string;
begin
  Path := ExtractFilePath(FileName);
  fName := ExtractFileName(FileName);
  if Pos('.', fName) = 0 then fName := fName + '.asc';
  if Path = '' then Path := ExtractFilePath(ParamStr(0));
  if not DirectoryExists(Path) then DoError(matDirNotFound);

  Adr := GetAddress(GetSize(Name, Rows, Cols));
  FullFileName := Path + fName;
  AssignFile(TF, FullFileName);
  Rewrite(TF);
    for I := 1 to Rows do begin
      for J := 1 to Cols do begin
        if Fixed then begin
          Str := FloatToStrExp(ElemFast[Adr, I, J, Cols]);
          while Length(Str) < 24 do Insert(' ', Str, 1);
        end else
          Str := FloatToStrGen(ElemFast[Adr, I, J, Cols]);
        if J < Cols then Str := Str + ' ';
        Write(TF, Str);
      end;
      Writeln(TF);
    end;
  CloseFile(TF);
end;


procedure TWorkspace.LoadFromAscii(const FileName: string);
var
  R: Real;
  S, Name: string;
  ColCount, ElCount: Integer;
  RowCount: Integer;
  Adr: Integer;
```

KGTB_2023

```
  ColKnown: Boolean;
  E: Extended;
  C: Char;
begin
  if FMStream = nil then FMStream := TMemoryStream.Create;
  FMStream.Clear;
  ColCount := 0;
  ElCount := 0;
  ColKnown := False;

  with TMemoryStream.Create do
  try
    LoadFromFile(FileName);
    Seek(0, soFromBeginning);
    while Position < Size do
    begin
      Read(C, 1);
      if C in ['0'..'9', '.', '-', '+', 'e', 'E'] then S := S + C else
      begin
        if (S <> '') or (Position = Size - 1) then
        begin
          E := AnyStrToFloat(S);
          if E > MaxDouble then E := MaxDouble;
          if E < -MaxDouble then E := -MaxDouble;
          R := E;
          S := '';
          if not ColKnown then Inc(ColCount);
          if (C in [#13, #10]) then ColKnown := True;
          MsGrow(FMStream);
          FMStream.Write(R, 8);
          Inc(ElCount);
        end;
      end;
    end;
    Free;
  except
    Free;
    DoError(matErrorLoadingArray);
  end;
  FMStream.Seek(0, soFromBeginning);

  if (ElCount mod ColCount) <> 0 then DoError(matErrorLoadingArray);
  RowCount := ElCount div ColCount;

  Name := ExtractFileName(FileName);
  while Pos('.', Name) > 0 do Delete(Name, Length(Name), 1);

  Adr := GetAddress(NewArray(Name, RowCount, ColCount));
  FMStream.Read(Pointer(Adr)^, RowCount * ColCount * 8);
  FMStream.Clear;
end;

function TWorkspace.SaveArrayToValues(const Name: string): string;
var
  I, J, Rows, Cols, Adr: Integer;
begin
  Result := '';
  Adr := GetAddress(GetSize(Name, Rows, Cols));
  for I := 1 to Rows do
    for J := 1 to Cols do
    begin
      if J < Cols then
        Insert(FloatToStrGen(ElemFast[Adr, I, J, Cols]) + ' ', Result,
Length(Result) + 1)
      else
        Insert(FloatToStrGen(ElemFast[Adr, I, J, Cols]) + #13#10, Result,
Length(Result) + 1)
    end;
end;
```

КПЗ_2023

```
procedure TWorkspace.LoadArrayFromMemory(ArrayName: string;
  ArrayAddress: Pointer; RowCount, ColCount: Integer;
  ElemType: TStandartType);
var
  Adr, I, ElCount: Integer;
  Bextended: Extended;
begin
  Adr := GetAddress(NewArray(ArrayName, RowCount, ColCount));
  ElCount := RowCount * ColCount - 1;

  case ElemType of

    stByte:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Byte(Pointer(Integer(ArrayAddress) +
I)^);

    stShortint:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := ShortInt(Pointer(Integer(ArrayAddress)
+ I)^);

    stWord:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Word(Pointer(Integer(ArrayAddress) + I
shl 1)^);

    stSmallint, stShort:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Smallint(Pointer(Integer(ArrayAddress)
+ I shl 1)^);

    stCardinal, stLongword, stDword:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Cardinal(Pointer(Integer(ArrayAddress)
+ I shl 2)^);

    stInteger, stLongint:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Integer(Pointer(Integer(ArrayAddress) +
I shl 2)^);

    stInt64:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Int64(Pointer(Integer(ArrayAddress) + I
shl 3)^);

    stReal, stDouble:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Real(Pointer(Integer(ArrayAddress) + I
shl 3)^);

    stReal48:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Real48(Pointer(Integer(ArrayAddress) +
I * 6)^);

    stSingle:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Single(Pointer(Integer(ArrayAddress) +
I shl 2)^);

    stExtended:
      for I := 0 to ElCount do
      begin
        Bextended := Extended(Pointer(Integer(ArrayAddress) + I * 10)^);
        if Bextended > MaxDouble then Bextended := MaxDouble;
        if Bextended < -MaxDouble then Bextended := -MaxDouble;
```

КОПЗ_2023

```
        Real(Pointer(Adr + I shl 3)^) := Bextended;
      end;

    stComp:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Comp(Pointer(Integer(ArrayAddress) + I
shl 3)^);

    stCurrency:
      for I := 0 to ElCount do
        Real(Pointer(Adr + I shl 3)^) := Currency(Pointer(Integer(ArrayAddress)
+ I shl 3)^);

    else
      DoError(matBadType);
  end;
end;

procedure TWorkspace.LoadArrayFromStream(ArrayName: string;
  Stream: TStream; Rows, Cols: Integer; ElemType: TStandartType;
  StartByte: DWORD = 0);
var
  I, ElCount, Adr: Integer;
  Buf: array[1..10] of Byte;
  Bextended: Extended;
  Pos: DWORD;
begin
  Pos := Stream.Position;
  Stream.Seek(StartByte, soFromBeginning);
  Adr := GetAddress(NewArray(ArrayName, Rows, Cols));
  ElCount := Rows * Cols - 1;

  case ElemType of

    stByte:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 1);
        Real(Pointer(Adr + I shl 3)^) := Byte(Pointer(@Buf)^);
      end;

    stShortint:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 2);
        Real(Pointer(Adr + I shl 3)^) := ShortInt(Pointer(@Buf)^);
      end;

    stWord:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 2);
        Real(Pointer(Adr + I shl 3)^) := Word(Pointer(@Buf)^);
      end;

    stSmallint, stShort:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 2);
        Real(Pointer(Adr + I shl 3)^) := Smallint(Pointer(@Buf)^);
      end;

    stCardinal, stLongword, stDword:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 4);
        Real(Pointer(Adr + I shl 3)^) := Cardinal(Pointer(@Buf)^);
      end;
```

КбТЗ_2023

```
    stInteger, stLongint:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 4);
        Real(Pointer(Adr + I shl 3)^) := Integer(Pointer(@Buf)^);
      end;

    stInt64:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 8);
        Real(Pointer(Adr + I shl 3)^) := Int64(Pointer(@Buf)^);
      end;

    stReal, stDouble:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 8);
        Real(Pointer(Adr + I shl 3)^) := Real(Pointer(@Buf)^);
      end;

    stReal48:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 6);
        Real(Pointer(Adr + I shl 3)^) := Real48(Pointer(@Buf)^);
      end;

    stSingle:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 4);
        Real(Pointer(Adr + I shl 3)^) := Single(Pointer(@Buf)^);
      end;

    stExtended:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 10);
        Bextended := Extended(Pointer(@Buf)^);
        if Bextended > MaxDouble then Bextended := MaxDouble;
        if Bextended < -MaxDouble then Bextended := -MaxDouble;
        Real(Pointer(Adr + I shl 3)^) := Bextended;
      end;

    stComp:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 8);
        Real(Pointer(Adr + I shl 3)^) := Comp(Pointer(@Buf)^);
      end;

    stCurrency:
      for I := 0 to ElCount do
      begin
        Stream.Read(Buf, 8);
        Real(Pointer(Adr + I shl 3)^) := Currency(Pointer(@Buf)^);
      end;

    else
      DoError(matBadType);
  end;

  Stream.Position := Pos;
end;

procedure TWorkspace.SaveArrayToStream(ArrayName: string; Stream: TStream;
  ElemType: TStandartType; StartByte: Integer = 0);
var
```

КБТЗ_2023

```
  Adr, I, ElCount, Cols, Rows: Integer;
  Buf: array[1..10] of Byte;
  Pos: DWORD;
begin
  if StartByte < 0 then StartByte := 0;
  Pos := Stream.Position;
  Adr := GetAddress(GetSize(ArrayName, Rows, Cols));
  ElCount := Rows * Cols - 1;

  case ElemType of

    stByte:
      begin
        Stream.Size := StartByte + ElCount;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Byte(Pointer(@Buf)^) := Byte(Round(Real(Pointer(Adr + I shl 3)^)));
          Stream.Write(Buf, 1);
        end;
      end;

    stShortint:
      begin
        Stream.Size := StartByte + ElCount;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          ShortInt(Pointer(@Buf)^) := ShortInt(Round(Real(Pointer(Adr + I shl
3)^)));
          Stream.Write(Buf, 1);
        end;
      end;

    stWord:
      begin
        Stream.Size := StartByte + ElCount * 2;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Word(Pointer(@Buf)^) := Word(Round(Real(Pointer(Adr + I shl 3)^)));
          Stream.Write(Buf, 2);
        end;
      end;

    stSmallint, stShort:
      begin
        Stream.Size := StartByte + ElCount * 2;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Smallint(Pointer(@Buf)^) := Smallint(Round(Real(Pointer(Adr + I shl
3)^)));
          Stream.Write(Buf, 2);
        end;
      end;

    stInteger, stLongint:
      begin
        Stream.Size := StartByte + ElCount * 4;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Integer(Pointer(@Buf)^) := Integer(Round(Real(Pointer(Adr + I shl
3)^)));
          Stream.Write(Buf, 4);
        end;
      end;

    stCardinal, stLongword, stDWord:
      begin
        Stream.Size := StartByte + ElCount * 4;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
```

K6IFS_2023

```
          Cardinal(Pointer(@Buf)^) := Cardinal(Round(Real(Pointer(Adr + I shl
3)^)));
          Stream.Write(Buf, 4);
        end;
      end;

    stInt64:
      begin
        Stream.Size := StartByte + ElCount * 8;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Int64(Pointer(@Buf)^) := Int64(Round(Real(Pointer(Adr + I shl 3)^)));
          Stream.Write(Buf, 8);
        end;
      end;

    stReal, stDouble:
      begin
        Stream.Size := StartByte + ElCount * 8;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Real(Pointer(@Buf)^) := Real(Pointer(Adr + I shl 3)^);
          Stream.Write(Buf, 8);
        end;
      end;

    stReal48:
      begin
        Stream.Size := StartByte + ElCount * 6;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Real48(Pointer(@Buf)^) := Real(Pointer(Adr + I shl 3)^);
          Stream.Write(Buf, 6);
        end;
      end;

    stSingle:
      begin
        Stream.Size := StartByte + ElCount * 4;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Single(Pointer(@Buf)^) := Real(Pointer(Adr + I shl 3)^);
          Stream.Write(Buf, 4);
        end;
      end;

    stExtended:
      begin
        Stream.Size := StartByte + ElCount * 10;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Extended(Pointer(@Buf)^) := Real(Pointer(Adr + I shl 3)^);
          Stream.Write(Buf, 10);
        end;
      end;

    stComp:
      begin
        Stream.Size := StartByte + ElCount * 8;
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Comp(Pointer(@Buf)^) := Real(Pointer(Adr + I shl 3)^);
          Stream.Write(Buf, 8);
        end;
      end;

    stCurrency:
      begin
        Stream.Size := StartByte + ElCount * 8;
```

КСПЗ_2023

```
        Stream.Seek(StartByte, soFromBeginning);
        for I := 0 to ElCount do begin
          Currency(Pointer(@Buf)^) := Real(Pointer(Adr + I shl 3)^);
          Stream.Write(Buf, 8);
        end;
      end;

    else
      DoError(matBadType);
  end;

  Stream.Seek(Pos, soFromBeginning);
end;


procedure TWorkspace.SaveArrayToMemory(ArrayName: string; ArrayAddress: Pointer;
  ElemType: TStandartType);
var
  Adr, I, ElCount, Rows, Cols: Integer;
begin
  Adr := GetAddress(GetSize(ArrayName, Rows, Cols));
  ElCount := Rows * Cols - 1;

  case ElemType of

    stByte:
      for I := 0 to ElCount do
        Byte(Pointer(Integer(ArrayAddress) + I)^) :=
          Byte(Round(Real(Pointer(Adr + I shl 3)^)));

    stShortint:
      for I := 0 to ElCount do
        ShortInt(Pointer(Integer(ArrayAddress) + I)^) :=
          ShortInt(Round(Real(Pointer(Adr + I shl 3)^)));

    stWord:
      for I := 0 to ElCount do
        Word(Pointer(Integer(ArrayAddress) + I shl 1)^) :=
          Word(Round(Real(Pointer(Adr + I shl 3 )^)));

    stSmallint, stShort:
      for I := 0 to ElCount do
        Smallint(Pointer(Integer(ArrayAddress) + I shl 1)^) :=
          Smallint(Round(Real(Pointer(Adr + I shl 3 )^)));

    stInteger, stLongint:
      for I := 0 to ElCount do
        Integer(Pointer(Integer(ArrayAddress) + I shl 2)^) :=
          Integer(Round(Real(Pointer(Adr + I shl 3)^)));

    stCardinal, stLongword, stDword:
      for I := 0 to ElCount do
        Cardinal(Pointer(Integer(ArrayAddress) + I shl 2)^) :=
          Cardinal(Round(Real(Pointer(Adr + I shl 3)^)));

    stInt64:
      for I := 0 to ElCount do
        Int64(Pointer(Integer(ArrayAddress) + I shl 3)^) :=
          Int64(Round(Real(Pointer(Adr + I shl 3)^)));

    stReal, stDouble:
      for I := 0 to ElCount do
        Real(Pointer(Integer(ArrayAddress) + I shl 3)^) := Real(Pointer(Adr + I
shl 3)^);

    stReal48:
      for I := 0 to ElCount do
        Real48(Pointer(Integer(ArrayAddress) + I * 6)^) := Real(Pointer(Adr + I
shl 3)^);
```

RTB_2023

```
    stSingle:
      for I := 0 to ElCount do
        Single(Pointer(Integer(ArrayAddress) + I shl 2)^) := Real(Pointer(Adr +
I shl 3)^);

    stExtended:
      for I := 0 to ElCount do
        Extended(Pointer(Integer(ArrayAddress) + I * 10)^) := Real(Pointer(Adr +
I shl 3)^);

    stComp:
      for I := 0 to ElCount do
        Comp(Pointer(Integer(ArrayAddress) + I shl 3)^) := Real(Pointer(Adr + I
shl 3)^);

    stCurrency:
      for I := 0 to ElCount do
        Currency(Pointer(Integer(ArrayAddress) + I shl 3)^) := Real(Pointer(Adr
+ I shl 3)^);

    else
      DoError(matBadType);

  end;
end;

procedure TWorkspace.FLoadFromTextFile(const FileName: string);
var
  C: Char;
  FormStr: Boolean;
  tStr, tStr2: string;
  fSize, Pos: Cardinal;
  MS: TMemoryStream;
begin
  if not FileExists(FileName) then DoError(matFileNotFound);
  fSize := FileSize(FileName);
  Pos := 0;

  tStr := ' ';
  tStr2 := '';
  FormStr := False;

  MS := TMemoryStream.Create;

  // Завантажуємо весь файл у потік
  MS.LoadFromFile(FileName);
  MS.Seek(0, soFromBeginning);
  with MS do begin
    while Position < Size do begin
      Read(C, 1);
      Inc(Pos);

      tStr := tStr[Length(tStr)] + C;
      if (C in ['a'..'z', 'A'..'Z']) and ((tStr[Length(tStr) - 1] <= ' ') or
        (tStr[Length(tStr) - 1] in [']', ';'])) then FormStr := not FormStr;

      if FormStr then Insert(C, tStr2, Length(tStr2) + 1);

      if ((not FormStr) or (Pos >= fSize)) and (tStr2 <> '') then
      begin
        try
          LoadArrayFromString(tStr2);
        except
          MS.Free;
          DoError(matErrorLoadingArray);
        end;
        FormStr := True;
        tStr2 := C;
```

КбПЗ – 2023

```
      end;
    end;
  end;
  MS.Free;
end;

procedure TWorkspace.FLoadArrayFromTextFile(const FileName: string;
  Name: string);
var
  TF: TextFile;
  C: Char;
  FormStr: Boolean;
  tStr, tStr2: string;
  fSize, Pos: Cardinal;
begin
  if not FileExists(FileName) then DoError(matFileNotFound);

  fSize := FileSize(FileName);
  Pos := 0;

  AssignFile(TF, FileName);
  Reset(TF);

  tStr := ' ';
  tStr2 := '';
  FormStr := False;

  while not EOF(TF) do begin
    Read(TF, C);
    Inc(Pos);

    tStr := tStr[Length(tStr)] + C;
    if (C in ['a'..'z', 'A'..'Z']) and ((tStr[Length(tStr) - 1] <= ' ')
      or (tStr[Length(tStr) - 1] in [']', ';'])) then FormStr := not FormStr;

    if FormStr then Insert(C, tStr2, Length(tStr2) + 1);

    if ((not FormStr) or (Pos >= fSize)) and (tStr2 <> '') then
    begin
      try
        if (Length(tStr2) > Length(Name)) and
           (Copy(tStr2, 1, Length(Name)) = Name) and
           (tStr2[Length(Name) + 1] in [#13, #10, ' ', '='])
        then
          LoadArrayFromString(tStr2);
      except
        CloseFile(TF);
        DoError(matErrorLoadingArray);
      end;
      FormStr := True;
      tStr2 := C;
    end;
  end;

  CloseFile(TF);
end;

procedure TWorkspace.SaveToTextFile(const FileName: string; ArName: string =
'');
begin
  if ArName = '' then FSaveToTextFile(FileName) else
    FSaveArrayToTextFile(FileName, ArName);
end;

procedure TWorkspace.FSaveArrayToTextFile(const FileName: string;
  Name: string);
var
  Ws: TWorkspace;
begin
```

КПЗ_2023

```
  GetSize(Name);
  Ws := SelfWS;
  with TWorkspace.Create('FSaveArrayToTextFile', Ws) do
  begin
    if FileExists(FileName) then FLoadFromTextFile(FileName);
    CopyRef(Ws, Name, Name);
    FSaveToTextFile(FileName);
    Free;
  end;
end;

procedure TWorkspace.FSaveToTextFile(FileName: string);
var
  TF: TextFile;
  I: Integer;
  Str: string;
begin
  if FArrayList.Count = 0 then Exit;
  if ExtractFileExt(FileName) = '' then FileName := FileName + '.dat';
  AssignFile(TF, FileName);
  Rewrite(TF);
  for I := 0 to FArrayList.Count - 1 do
    if FArrayList[I][1] <> '$' then
    begin
      Str := SaveArrayToString(FArrayList[I]);
      Write(TF, Str);
    end;
  CloseFile(TF);
end;

procedure TWorkspace.LoadFromBinFile(const FileName: string; NameAr: string);
var
  FS: TFileStream;
  cHead, cHead1: array[1..30] of Char;
  ArName: TNameArray;
  Name: string;
  J, ArSize: Integer;
  Rang: TPoint;
  Adr: Integer;
  TByte, ByteCnt: Byte;
  TWord, DataType: Word;
  TDWord: DWord;

begin
  if FMStream = nil then FMStream := TMemoryStream.Create;
  cHead := matBinaryHeader; //30 b

  if not FileExists(FileName) then DoError(matFileNotFound);

  FS := TFileStream.Create(FileName, fmOpenRead);
  FS.Seek(0, soFromBeginning);
  FS.Read(cHead1, 30); // Зчитуємо заголовну інформацію
  if cHead <> cHead1 then begin
    FS.Free;
    DoError(matBadFileFormat);
  end;

  while FS.Position < FS.Size do
  begin
    FS.Read(ArName, 32); // Зчитуємо ім'я масиву
    Name := ArName;
    Name := Trim(Name);
    FS.Read(Rang, 8);     // Зчитуємо розміри
    ArSize := Rang.X * Rang.Y;
    FS.Read(DataType, 2); // Зчитуємо тип елементів

    case TStandartType(DataType) of
      stByte, stShortint : ByteCnt := 1;
      stWord, stShort    : ByteCnt := 2;
```

КбПЗ_2023

```
  stDWord, stInteger : ByteCnt := 4;
  else ByteCnt := 8;
end;

// Якщо ім'я неприпустиме або не те, що потрібно, то пропускаємо
if (not IsTrueName(Name)) or (ArSize = 0) or
   ((NameAr <> '') and (Name <> NameAr)) then
begin
  FS.Seek(ArSize * ByteCnt, soFromCurrent); // Ігноруємо всі елементи
  Continue;
end;
// Створюємо масив Name
Adr := GetAddress(NewArray(Name, Rang.X, Rang.Y));

// Якщо елементи речовинні, то відразу завантажуємо з файлу
if DataType = 0 then FS.Read(Pointer(Adr)^, ArSize * 8) else
begin
  // У противному випадку зчитуємо масив у потік
  FMStream.SetSize(ArSize * ByteCnt); // Фіксуємо розміри
  FMStream.Seek(0, soFromBeginning);
  // Зчитуємо весь масив
  FS.Read(Pointer(FMStream.Memory)^, ArSize * ByteCnt);
  FMStream.Seek(0, soFromBeginning);

  case TStandartType(DataType) of
    stByte:
       for J := 1 to ArSize do begin
         FMStream.Read(TByte, 1);
         Real(Pointer(Adr)^) := TByte;
         Inc(Adr, 8);
       end;

    stShortint:
       for J := 1 to ArSize do begin
         FMStream.Read(TByte, 1);
         Real(Pointer(Adr)^) := Shortint(TByte);
         Inc(Adr, 8);
       end;

    stWord:
       for J := 1 to ArSize do begin
         FMStream.Read(TWord, 2);
         Real(Pointer(Adr)^) := TWord;
         Inc(Adr, 8);
       end;

    stShort:
       for J := 1 to ArSize do begin
         FMStream.Read(TWord, 2);
         Real(Pointer(Adr)^) := Short(TWord);
         Inc(Adr, 8);
       end;

    stDWord:
       for J := 1 to ArSize do begin
         FMStream.Read(TDWord, 4);
         Real(Pointer(Adr)^) := TDWord;
         Inc(Adr, 8);
       end;

    stInteger:
       for J := 1 to ArSize do begin
         FMStream.Read(TDWord, 4);
         Real(Pointer(Adr)^) := Integer(TDWord);
         Inc(Adr, 8);
       end;
  end; // of case
end;
if NameAr <> '' then Break; // Потрібний масив зчитали, виходимо із циклу
```

КПЗ – 2023

```
  end;
  FS.Free;
  FMStream.Clear;
end;

procedure TWorkspace.PutArrayToMatlab(const Name: string);
var
  I, J, aIdx: Integer;
  Rows, Cols: Integer;
begin
  aIdx := GetSize(Name, Rows, Cols);
  Matlab := CreateOleObject('Matlab.Application');
  if (Rows = 1) and (Cols = 1) then
    MatlabArray := VarArrayCreate([1, 1, 1, 2], varDouble)
  else
    MatlabArray := VarArrayCreate([1, Rows, 1, Cols], varDouble);

  MatlabVar := VarArrayCreate([1, 2], varDouble);

  if (Rows = 1) and (Cols = 1) then
  begin
    MatlabArray[1, 1] := Elem[aIdx, 1, 1];
    MatlabArray[1, 2] := 0;
  end else
    for I := 1 to Rows do
      for J := 1 to Cols do
        MatlabArray[I, J] := Elem[aIdx, I, J];

  Matlab.PutFullMatrix(Name, 'base', VarArrayRef(MatlabArray),
    VarArrayRef(MatlabVar));
   // Скорочуємо розмір переданого масиву вдвічі
  if (Rows = 1) and (Cols = 1) then
    Matlab.Execute(Name + ' = ' + Name + '(1)') else
    Matlab.Execute(Name + ' = ' + Name + '(:,:)');
end;


function TWorkspace.LoadArrayFromMatlab(const Name: string): Boolean;
var
  I, J, aIdx: Integer;
  Rows, Cols: Integer;
begin
  Result := False;
  Matlab := CreateOleObject('Matlab.Application');

  MatlabArray := VarArrayCreate([1, 1], varDouble);
  MatlabVar := VarArrayCreate([1, 2], varDouble);

  Matlab.Execute('Matrix_tmp=size(' + Name + ')');
  Matlab.Execute('Matrix_tmp=length(Matrix_tmp)');

  try
    Matlab.GetFullMatrix('Matrix_tmp', 'base', VarArrayRef(MatlabArray),
VarArrayRef(MatlabVar));
  except
    Exit;
  end;

  if MatlabArray[1] = 1 then
  begin // Одне число
    MatlabArray := VarArrayCreate([1, 1], varDouble);
    Matlab.GetFullMatrix(Name, 'base', VarArrayRef(MatlabArray),
VarArrayRef(MatlabVar));
    NewArray(Name, 1, 1);
    Elem[Name, 1, 1] := MatlabArray[1]
  end;

  if MatlabArray[1] = 2 then
  begin // Матриця
```

КоПЗ_2023

```
    // Запитуємо кількість рядків і стовпців
    MatlabArray := VarArrayCreate([1, 1, 1, 2], varDouble);
    Matlab.Execute('Matrix_tmp=size(' + Name + ')');
    Matlab.GetFullMatrix('Matrix_tmp', 'base', VarArrayRef(MatlabArray),
VarArrayRef(MatlabVar));
    Rows := MatlabArray[1, 1];
    Cols := MatlabArray[1, 2];
    if Rows * Cols = 0 then Exit;
    MatlabArray := VarArrayCreate([1, Rows, 1, Cols], varDouble);
    Matlab.GetFullMatrix(Name, 'base', VarArrayRef(MatlabArray),
VarArrayRef(MatlabVar));
    aIdx := NewArray(Name, Rows, Cols);
    for I := 1 to Rows do
      for J := 1 to Cols do
        Elem[aIdx, I, J] := MatlabArray[I, J];
  end;
  Matlab.Execute('clear Matrix_tmp');
  Result := True;
end;

function TWorkspace.GetAddress(Idx: Integer): Integer;
begin
  if not ArrayExists(Idx) then DoError(matArrayNotFound);
  Result := FArrayList.ItemsAddr[Idx];
end;

function TWorkspace.GetName(Idx: Integer): string;
begin
  if Idx > FArrayList.Count - 1 then DoError(matArrayNotFound);
  Result := FArrayList[Idx];
  if Result[1] = '$' then DoError(matArrayNotFound);
end;

function TWorkspace.GetRows(const Name: string): Integer;
begin
  Result := FArrayList.ItemsRows[GetIndex(Name)];
end;

function TWorkspace.GetCols(const Name: string): Integer;
begin
  Result := FArrayList.ItemsCols[GetIndex(Name)];
end;

function TWorkspace.GetRows(Idx: Integer): Integer;
begin
  if not ArrayExists(Idx) then DoError(matArrayNotFound);
  Result := FArrayList.ItemsRows[Idx];
end;

function TWorkspace.GetCols(Idx: Integer): Integer;
begin
  if not ArrayExists(Idx) then DoError(matArrayNotFound);
  Result := FArrayList.ItemsCols[Idx];
end;

procedure TWorkspace.SetEl(Idx: Integer; const Value: Real; Row, Col: Integer;
  AutoSize: Boolean);
var
  Rows, Cols, NewRows, NewCols: Integer;
begin
  GetSize(Idx, Rows, Cols);
  if (Row < 1) or (Col < 1) then DoError(matBadCoords);
  if (Row > Rows) or (Col > Cols) then
  begin
    if not AutoSize then DoError(matBadCoords) else
      begin
        NewRows := Max(Row, Rows);
        NewCols := Max(Col, Cols);
        if (NewRows = 1) or (NewCols = 1) then
```

КОПЗ_2023

```
          Idx := Resize(FArrayList[Idx], NewRows, NewCols)
        else
          Idx := ResizeMatrix(FArrayList[Idx], NewRows, NewCols);
      end;
  end;
  Real(Pointer((FArrayList.ItemsAddr[Idx] + ((Row - 1) * Cols  + Col) shl 3 -
8))^) := Value;
end;

function TWorkspace.FGetRef(Name: string): TMatrixData;
var
  I: Integer;
begin
  I := Find(Name);
  if I = -1 then begin
    Result.NameArray := '';
    Exit;
  end;
  Result.NameArray := Name;
  Result.Address := FArrayList.ItemsAddr[I];
  Result.Rows := FArrayList.ItemsRows[I];
  Result.Cols := FArrayList.ItemsCols[I];
end;

procedure TWorkspace.FSetRef(Ref: TMatrixData);
var
  f, I: Integer;
begin
  if Ref.NameArray = '' then Exit;
  f := Find(Ref.NameArray);
  if (f = -1) and (FArrayList.Count <> 0) then
    for I := 0 to FArrayList.Count - 1 do
      if FArrayList[I][1] = '$' then
      begin
        f := I;
        Break;
      end;

  if f <> -1 then
  begin
    // Якщо область у пам'яті інша, то видаляємо масив
    if (FArrayList.ItemsAddr[f] <> 0) and
      (FArrayList.ItemsAddr[f] <> Ref.Address)
      then Clear(Ref.NameArray);
    with FArrayList do
    begin
      ItemsName[f] := Ref.NameArray;
      ItemsRows[f] := Ref.Rows;
      ItemsCols[f] := Ref.Cols;
      ItemsAddr[f] := Ref.Address;
    end;
    Exit;
  end;
  FArrayList.Add(MatrixData(Ref.NameArray, Ref.Rows, Ref.Cols, Ref.Address));
end;

procedure TWorkspace.FDelRef(Name: string);
var
  f: Integer;
begin
  f := Find(Name);
  if f = -1 then Exit;
  FArrayList.Delete(f);
  FRefList.Remove(Pointer(f));
end;

function TWorkspace.CopyArray(const SourAr, DestAr: string): Integer;
var
  I, K, C, Rows, Cols: Integer;
```

КПЗ_2023

```
begin
  I := GetSize(SourAr, Rows, Cols);
  Result := I;
  if SourAr = DestAr then Exit;
  C := Rows * Cols * 8;
  K := NewArray(DestAr, Rows, Cols);
  Result := K;
  Move(Pointer(FArrayList.ItemsAddr[I])^, Pointer(FArrayList.ItemsAddr[K])^, C);
end;

function TWorkspace.RenameArray(const SourAr, DestAr: string): Integer;
var
  I, ResInd: Integer;
begin
  if not IsTrueName(DestAr) then DoError(matBadName);
  I := GetIndex(SourAr);
  Result := I;
  if SourAr = DestAr then Exit;
  ResInd := Clear(DestAr);
  if ResInd <> -1 then
  begin
    with FArrayList do
    begin
      ItemsName[ResInd] := DestAr;
      ItemsRows[ResInd] := ItemsRows[I];
      ItemsCols[ResInd] := ItemsCols[I];
      ItemsAddr[ResInd] := ItemsAddr[I];
    end;
    FArrayList.Delete(I);
    Result := ResInd;
  end else
    FArrayList[I] := DestAr;
  FArrayList.SetIndex(Result);
end;

function TWorkspace.GetElem(const Name: string; Row, Col: Integer): Real;
var
  Idx: Integer;
begin
  Idx := GetIndex(Name);
{$ifdef CheckRange}
  if (Row < 1) or (Col < 1) or (Row > FArrayList.ItemsRows[Idx])
    or (Col > FArrayList.ItemsCols[Idx]) then DoError(matBadRowCol);
{$endif}
  Result := Real(Pointer((FArrayList.ItemsAddr[Idx] + ((Row - 1) *
   FArrayList.ItemsCols[Idx] + Col - 1) shl 3))^);
end;

procedure TWorkspace.SetElem(const Name: string; Row, Col: Integer; const Value:
Real);
var
  Idx: Integer;
begin
  Idx := GetIndex(Name);
{$ifdef CheckRange}
  if (Row < 1) or (Col < 1) or (Row > FArrayList.ItemsRows[Idx])
    or (Col > FArrayList.ItemsCols[Idx]) then DoError(matBadRowCol);
{$endif}
  Real(Pointer((FArrayList.ItemsAddr[Idx] + ((Row - 1) *
    FArrayList.ItemsCols[Idx] + Col - 1) shl 3))^) := Value;
end;

function TWorkspace.GetElem(Idx, Row, Col: Integer): Real;
begin
{$ifdef CheckRange}
  if (Row < 1) or (Col < 1) or (Row > FArrayList.ItemsRows[Idx])
    or (Col > FArrayList.ItemsCols[Idx]) then DoError(matBadRowCol);
{$endif}
  Result := Real(Pointer((FArrayList.ItemsAddr[Idx] +
```

КПЗ – 2023

```
    ((Row - 1) * FArrayList.ItemsCols[Idx] + Col - 1) shl 3))^);
end;

procedure TWorkspace.SetElem(Idx, Row, Col: Integer; const Value: Real);
begin
{$ifdef CheckRange}
  if (Row < 1) or (Col < 1) or (Row > FArrayList.ItemsRows[Idx])
    or (Col > FArrayList.ItemsCols[Idx]) then DoError(matBadRowCol);
{$endif}
  Real(Pointer((FArrayList.ItemsAddr[Idx] +
    ((Row - 1) * FArrayList.ItemsCols[Idx] + Col - 1) shl 3))^) := Value;
end;

function TWorkspace.Transpose(const SourAr: string; DestAr: string): string;
var
  NewRowCount, NewColCount, Id: Integer;
  AdrMain, AdrRes, IRes, MainShift, ResShift, NewColCount8: Integer;
  I, J: Integer;
  TempVar: string;
begin
  Result := CheckResAr(DestAr);
  Id := GetSize(SourAr, NewColCount, NewRowCount);
  AdrMain := GetAddress(Id);
  IRes := Find(DestAr);
  if (IRes <> -1) and (DestAr <> SourAr) and (GetRows(IRes) = NewRowCount) and
    (GetCols(IRes) = NewColCount) then AdrRes := GetAddress(IRes)
  else begin
    TempVar := GenName();
    AdrRes := GetAddress(NewArray(TempVar, NewRowCount, NewColCount));
  end;
  if (NewColCount = 1) or (NewRowCount = 1) then
    Move(Pointer(AdrMain)^, Pointer(AdrRes)^, NewRowCount * NewColCount * 8)
  else

  begin
    NewColCount8 := NewColCount * 8;
    for I := 0 to NewColCount - 1 do
    begin
      MainShift := I * NewRowCount * 8;
      ResShift := I * 8;
      for J := 1 to NewRowCount do
      begin
        Real(Pointer(AdrRes + ResShift)^) := Real(Pointer(AdrMain +
MainShift)^);
        Inc(ResShift, NewColCount8);
        Inc(MainShift, 8);
      end;
    end;
  end;

  if TempVar <> '' then RenameArray(TempVar, DestAr);
end;

function TWorkspace.CheckResAr(var Name: string; DoANS: Boolean = True): string;
begin
  if DoANS and (Name = '') then Name := 'ans' else
    if not IsTrueName(Name) then DoError(matBadName);
  Result := Name;
end;

function TWorkspace.IsTrueName(const Name: string): Boolean;
var
  I, Len: Integer;
begin
  Result := False;
  Len := Length(Name);
  if (Len = 0) or (Len > 32) then Exit;
  if not (Name[1] in ['A'..'Z', 'a'..'z']) then Exit;
  for I := 2 to Len do
```

КОТЗ_2023

```
    if not (Name[I] in ['A'..'Z', 'a'..'z', '0'..'9', '_']) then Exit;
  Result := True;
end;

function TWorkspace.HandInput(var InputArray,
  TempArray: string): Integer;
var
  I: Integer;
begin
  I := Find(InputArray);
  if I <> -1 then
  begin
    Result := I;
    Exit;
  end;
  if InputArray <> '' then
    if InputArray[1] in ['0'..'9', '.', '-', '+', '['] then
    begin
      TempArray := GenName();
      SLoad(TempArray, InputArray);
      InputArray := TempArray;
      Result := GetIndex(InputArray);
      Exit;
    end;
  Result := -1;
  DoError(matArrayNotFound);
end;

function TWorkspace.GenName(Add: string = ''): string;
begin
  if Add <> '' then Result := 'I_' + Add + '_TV' else begin
    Inc(FUniqueNameCounter);
    Result := 'I_' + IntToStr(FUniqueNameCounter) + '_TV';
    if FUniqueNameCounter >= MaxComp then
    begin
      FUniqueNameCounter := 0;
      if MessageBox(GetActiveWindow, matCalcLimit, 'Warning!',
        MB_ICONWARNING or MB_OKCANCEL) = IDCANCEL then DoError('');
    end;
  end;
end;

function TWorkspace.AddRows(const SourAr1, SourAr2: string; DestAr: string):
string;
var
  ColCnt1, ColCnt2, ColCntRes: Integer;
  RowCnt1, RowCnt2, RowCntRes: Integer;
  Adr1, Adr2, AdrRes: Integer;
  TempRes: string;
begin
  Result := CheckResAr(DestAr);
  if (SourAr1 = DestAr) and (Find(DestAr) = -1) then
  begin
    CopyArray(SourAr2, DestAr);
    Exit;
  end;
  if (SourAr2 = DestAr) and (Find(DestAr) = -1) then
  begin
    CopyArray(SourAr1, DestAr);
    Exit;
  end;
  TempRes := GenName();
  Adr1 := GetAddress(GetSize(SourAr1, RowCnt1, ColCnt1));
  Adr2 := GetAddress(GetSize(SourAr2, RowCnt2, ColCnt2));
  if ColCnt1 <> ColCnt2 then DoError(matArraysNotAgree);
  ColCntRes := ColCnt1;
  RowCntRes := RowCnt1 + RowCnt2;
  AdrRes := GetAddress(NewArray(TempRes, RowCntRes, ColCntRes));
  Move(Pointer(Adr1)^, Pointer(AdrRes)^, RowCnt1 * ColCnt1 * 8);
```

КбПЗ – 2023

```
  Move(Pointer(Adr2)^, Pointer(AdrRes + RowCnt1 * ColCnt1 * 8)^, RowCnt2 *
ColCnt2 * 8);
  RenameArray(TempRes, DestAr);
end;

function TWorkspace.AddCols(const SourAr1, SourAr2: string; DestAr: string):
string;
var
  ColCnt1, ColCnt2, ColCntRes: Integer;
  RowCnt1, RowCnt2, RowCntRes: Integer;
  Adr1, Adr2, AdrRes: Integer;
  I: Integer;
  CurrentAddress: Integer;
  TempRes: string;
begin
  Result := CheckResAr(DestAr);
  if (SourAr1 = DestAr) and (Find(DestAr) = -1) then
  begin
    CopyArray(SourAr2, DestAr);
    Exit;
  end;
  if (SourAr2 = DestAr) and (Find(DestAr) = -1) then
  begin
    CopyArray(SourAr1, DestAr);
    Exit;
  end;
  TempRes := GenName();
  Adr1 := GetAddress(GetSize(SourAr1, RowCnt1, ColCnt1));
  Adr2 := GetAddress(GetSize(SourAr2, RowCnt2, ColCnt2));
  if RowCnt1 <> RowCnt2 then DoError(matArraysNotAgree);
  RowCntRes := RowCnt1;
  ColCntRes := ColCnt1 + ColCnt2;
  AdrRes := GetAddress(NewArray(TempRes, RowCntRes, ColCntRes));
  CurrentAddress := AdrRes;
  ColCnt1 := ColCnt1 shl 3;
  ColCnt2 := ColCnt2 shl 3;
  for I := 0 to RowCntRes-1 do
  begin
    Move(Pointer(Adr1)^, Pointer(CurrentAddress)^, ColCnt1);
    Inc(CurrentAddress, ColCnt1);
    Move(Pointer(Adr2)^, Pointer(CurrentAddress)^, ColCnt2);
    Inc(CurrentAddress, ColCnt2);
    Inc(Adr1, ColCnt1);
    Inc(Adr2, ColCnt2);
  end;
  RenameArray(TempRes, DestAr);
end;

function TWorkspace.CopyCutRows(SourAr, DestAr: string;
  RowNumber, Rows: Integer; Cut: Boolean = False): string;
var
  AdrMat1, AdrMat2, AdrMatRes, Idx, iTmp: Integer;
  ColCnt, RowCnt: Integer;
  TempRes, Temp2: string;
begin
  Result := CheckResAr(DestAr);
  Temp2 := GenName();
  TempRes := GenName();
  AdrMat1 := GetAddress(GetSize(SourAr, RowCnt, ColCnt));
  if (RowCnt < RowNumber) or (RowNumber < 1) then DoError(matBadRow);
  if RowNumber + Rows > RowCnt + 1 then DoError(matBadRow);
  // Створюємо тимчасовий масив
  // Якщо результуючий масив уже є й має необхідну розмірність, то
  // немає необхідності створювати його заново
  Idx := Find(DestAr);
  if (Idx <> -1) and (GetRows(Idx) = Rows) and (GetCols(Idx) = ColCnt) then
    AdrMatRes := GetAddress(RenameArray(DestAr, TempRes)) else
    AdrMatRes := GetAddress(NewArray(TempRes, Rows, ColCnt));
   // Копіюємо рядок у тимчасовий масив
```

КОПІЯ_2023

```
  Move(Pointer(AdrMat1 + (RowNumber - 1) * ColCnt shl 3)^,
    Pointer(AdrMatRes)^, (ColCnt shl 3) * Rows);
   // Якщо потрібно було вирізати, то ...
  if Cut and (Rows < RowCnt) then
  begin
    AdrMat2 := GetAddress(NewArray(Temp2, RowCnt - Rows, ColCnt));
    ColCnt := ColCnt * 8;
    Dec(RowNumber);
    iTmp := ColCnt * RowNumber;
    Move(Pointer(AdrMat1)^, Pointer(AdrMat2)^, iTmp);
    Move(Pointer(AdrMat1 + iTmp + ColCnt * Rows)^, Pointer(AdrMat2 + iTmp)^,
      ColCnt * (RowCnt - Rows) - iTmp);
    RenameArray(Temp2, SourAr);
  end;
  // Якщо він вихідного масиву нічого не залишається, то видаляємо його
  if Cut and (Rows >= RowCnt) then Clear(SourAr);
  RenameArray(TempRes, DestAr);
  Clear(Temp2);
end;

function TWorkspace.CopyCutCols(SourAr, DestAr: string; ColNumber,
  Cols: Integer; Cut: Boolean): string;
var
  IndMat2, Idx: Integer;
  AdrMat1, AdrMat2, AdrMatRes, tmpCols, tmpCNumber, tmpAdr, tmpCCnt: Integer;
  ColCnt, RowCnt: Integer;
  Mat2CC: Integer;
  I: Integer;
  Temp2, TempRes: string;
begin
  Result := CheckResAr(DestAr);
  AdrMat1 := GetAddress(GetSize(SourAr, RowCnt, ColCnt));
  if (ColCnt < ColNumber) or (ColNumber < 1) then DoError(matBadCol);
  if ColNumber + Cols > ColCnt + 1 then DoError(matBadCol);
  Temp2 := GenName();
  TempRes := GenName();
  // Створюємо тимчасовий масив, куди буде копіюватися стовпець
  Idx := Find(DestAr);
  if (Idx <> -1) and (GetRows(Idx) = RowCnt) and (GetCols(Idx) = Cols) then
    AdrMatRes := GetAddress(RenameArray(DestAr, TempRes))
  else
    AdrMatRes := GetAddress(NewArray(TempRes, RowCnt, Cols));

  // Копіюємо стовпець у тимчасовий масив
  tmpCols := Cols shl 3;
  tmpCCnt := ColCnt shl 3;
  tmpCNumber := (ColNumber - 1) shl 3;
  tmpAdr := AdrMat1 + tmpCNumber;
  for I := 0 to RowCnt-1 do
  begin
    Move(Pointer(tmpAdr)^, Pointer(AdrMatRes)^, tmpCols);
    Inc(tmpAdr, tmpCCnt);
    Inc(AdrMatRes, tmpCols);
  end;

  // Якщо потрібно було вирізати, то ...
  if Cut and (Cols < ColCnt) then
  begin
    // Створюємо нову тимчасову змінну
    IndMat2 := NewArray(Temp2, RowCnt, ColCnt - Cols);
    AdrMat2 := GetAddress(IndMat2);
    Mat2CC := GetCols(IndMat2);

    ColNumber := (ColNumber - 1) shl 3;
    tmpCols := Cols shl 3 + ColNumber;
    tmpCCnt := ColCnt shl 3;
    Mat2CC := Mat2CC shl 3;
    for I := 1 to RowCnt do
    begin
```

КОТЗ - 2023

```
      Move(Pointer(AdrMat1)^, Pointer(AdrMat2)^, ColNumber);
      Move(Pointer(AdrMat1 + tmpCols)^, Pointer(AdrMat2 + ColNumber)^, Mat2CC -
ColNumber);
      Inc(AdrMat1, tmpCCnt);
      Inc(AdrMat2, Mat2CC);
    end;
    RenameArray(Temp2, SourAr);
  end;
  if Cut and (Cols >= ColCnt) then Clear(SourAr);
  RenameArray(TempRes, DestAr);
  Clear(Temp2);
end;

function TWorkspace.CopySubmatrix(SourAr, DestAr: string; FirstRow, Rows,
  FirstCol, Cols: Integer): string;
var
  RowCnt, ColCnt, I, Idx: Integer;
  Adr1, Adr2: Integer;
  Temp1: string;
begin
  if (FirstRow < 1) or (FirstCol < 1) then DoError(matBadRowCol);
  Result := CheckResAr(DestAr);
  Adr1 := GetAddress(GetSize(SourAr, RowCnt, ColCnt)) +
    ((FirstRow - 1) * ColCnt + FirstCol - 1) shl 3;

  if (FirstRow > RowCnt) or (FirstCol > ColCnt) then DoError(matBadRowCol);
  if (FirstRow + Rows - 1 > RowCnt) or (FirstCol + Cols - 1 > ColCnt) then
    DoError(matBadRowCol);
  Temp1 := GenName();

  Idx := Find(DestAr);
  if (Idx <> -1) and (GetRows(Idx) = Rows) and (GetCols(Idx) = Cols)
    then Temp1 := DestAr else NewArray(Temp1, Rows, Cols, True);
  Adr2 := GetAddress(Temp1);

  ColCnt := ColCnt shl 3;
  Cols := Cols shl 3;
  for I := 1 to Rows do
  begin
    Move(Pointer(Adr1)^, Pointer(Adr2)^, Cols);
    Inc(Adr1, ColCnt);
    Inc(Adr2, Cols);
  end;
  // Перейменовуємо тимчасовий масив у результуючий
  if Temp1 <> DestAr then RenameArray(Temp1, DestAr);
end;

function TWorkspace.PasteSubmatrix(SourAr, DestAr: string; FirstPastRow,
  FirstPastCol: Integer): string;
var
  RowCnt, ColCnt, Id1, Id2: Integer;
  MinDestRowCnt, MinDestColCnt: Integer;
  ResRowCnt, ResColCnt: Integer;
  Adr1, Adr2, I: Integer;
begin
  if SourAr = DestAr then DoError(matErrorPast);
  Result := CheckResAr(DestAr);
  if (FirstPastRow < 1) or (FirstPastCol < 1) then DoError(matBadRowCol);
  Id1 := GetSize(SourAr, RowCnt, ColCnt);
  // Визначаємо мінімально припустимі розміри результуючою матриці
  MinDestRowCnt := RowCnt + FirstPastRow - 1;
  MinDestColCnt := ColCnt + FirstPastCol - 1;
  Id2 := Find(DestAr);
  if Id2 <> -1 then
  begin
    GetSize(Id2, ResRowCnt, ResColCnt);
    if (ResRowCnt < MinDestRowCnt) or (ResColCnt < MinDestColCnt) then
      DoError(matArraysNotAgree);
  end else
```

КоПЗ_2023

```
  begin
    ResRowCnt := MinDestRowCnt;
    ResColCnt := MinDestColCnt;
    Id2 := NewArray(DestAr, ResRowCnt, ResColCnt, True);
  end;
  Adr1 := GetAddress(Id1);
  Adr2 := GetAddress(Id2) +
   ((FirstPastRow - 1) * ResColCnt + FirstPastCol - 1) shl 3;

  ColCnt := ColCnt shl 3;
  ResColCnt := ResColCnt shl 3;
  for I := 1 to RowCnt do
  begin
    Move(Pointer(Adr1)^, Pointer(Adr2)^, ColCnt);
    Inc(Adr1, ColCnt);
    Inc(Adr2, ResColCnt);
  end;
end;

function TWorkspace.ResizeMatrix(const Name: string; Rows, Cols: Integer):
Integer;
var
  Temp: string;
  RC, CC, Ind: Integer;
begin
  if (Rows < 1) or (Cols < 1) then DoError(matBadSize);
  Ind := Find(Name);
  if Ind = -1 then begin
    Result := NewArray(Name, Rows, Cols, True);
    Exit;
  end;
  GetSize(Ind, RC, CC);
  Result := Ind;
  if Int64(Point(RC, CC)) = Int64(Point(Rows, Cols)) then Exit;
  Temp := GenName();
  CopySubmatrix(Name, Temp, 1, Min(Rows, RC), 1, Min(Cols, CC));
  Result := NewArray(Name, Rows, Cols, True);
  PasteSubmatrix(Temp, Name, 1, 1);
  Clear(Temp);
end;

function TWorkspace.RandomAr(const Name: string; Rows, Cols: Integer): string;
var
  Adr, I: Integer;
begin
  if (Rows = 0) or (Cols = 0) then
    Adr := GetAddress(GetSize(Name, Rows, Cols))
  else
    Adr := GetAddress(NewArray(Name, Rows, Cols));

  for I := 1 to Rows * Cols do
  begin
    Real(Pointer(Adr)^) := Random;
    Inc(Adr, 8);
  end;
  Result := Name;
end;

function TWorkspace.FillAr(const Name: string; BaseValue: Real; Step: Extended):
string;
var
  Rows, Cols, Adr, I: Integer;
  TmpValue: Extended;
begin
  Adr := GetAddress(GetSize(Name, Rows, Cols));
  TmpValue := BaseValue;
  for I := 1 to Rows * Cols do
  begin
    Real(Pointer(Adr)^) := TmpValue;
```

КОПІЯ 2023

```
    TmpValue := TmpValue + Step;
    Inc(Adr, 8);
  end;
  Result := Name;
end;

function TWorkspace.NewFillAr(const Name: string; Rows, Cols: Integer;
  const BaseValue: Real; Step: Extended): string;
begin
  NewArray(Name, Rows, Cols);
  Result := FillAr(Name, BaseValue, Step);
end;

function TWorkspace.MulMatrix(const Matrix1, Matrix2: string; MatResult:
string): string;
var
  M1Rows, M2Rows, M1Cols, M2Cols, I, J, K, Adr1, Adr2, AdrRes, IRes: Integer;
  Adr2Temp, M2Cols8, M2Rows8: Integer;
  Tmp: Real;
  TempRes: string;
begin
  Result := CheckResAr(MatResult);
  Adr1 := GetAddress(GetSize(Matrix1, M1Rows, M1Cols));
  Adr2 := GetAddress(GetSize(Matrix2, M2Rows, M2Cols));
  if M1Cols <> M2Rows then DoError(matArraysNotAgree);
  IRes := Find(MatResult);
  if (IRes <> -1) and (MatResult <> Matrix1) and (MatResult <> Matrix2) and
    (GetRows(IRes) = M1Rows) and (GetCols(IRes) = M2Cols) then
    AdrRes := GetAddress(IRes)
  else begin
    TempRes := GenName();
    AdrRes := GetAddress(NewArray(TempRes, M1Rows, M2Cols));
  end;
  M1Cols := M1Cols shl 3;
  M2Cols8 := M2Cols shl 3;
  M2Rows8 := M2Rows shl 3;
  for I := 1 to M1Rows do
  begin
    for J := 1 to M2Cols do
    begin
      Tmp := 0;
      Adr2Temp := Adr2;
      for K := 1 to M2Rows do
      begin
        Tmp := Tmp + Real(Pointer(Adr1)^) * Real(Pointer(Adr2)^);
        Inc(Adr1, 8);
        Inc(Adr2, M2Cols8);
      end;
      Adr2 := Adr2Temp + 8;
      Real(Pointer(AdrRes)^) := Tmp;
      Dec(Adr1, M2Rows8);
      Inc(AdrRes, 8);
    end;
    Inc(Adr1, M1Cols);
    Dec(Adr2, M2Cols8);
  end;
  if TempRes <> '' then RenameArray(TempRes, MatResult);
end;

function TWorkspace.IsEqual(const Array1, Array2: string; ResArray: string =
''):
  Boolean;
var
  I, Adr1, Adr2, Rows1, Rows2, Cols1, Cols2: Integer;
begin
  if ResArray <> '' then
    if not IsTrueName(ResArray) then DoError(matBadName);
  if (ResArray = Array1) or (ResArray = Array2) then DoError(matBadName);
  if ResArray <> '' then NewZeros(ResArray, 1, 1);
```

КБПЗ_2023

```
  Result := False;
  Adr1 := GetAddress(GetSize(Array1, Rows1, Cols1));
  Adr2 := GetAddress(GetSize(Array2, Rows2, Cols2));
  if not (Int64(Point(Rows1, Cols1)) = Int64(Point(Rows2, Cols2))) then Exit;

  for I := 1 to Rows1 * Cols1 do
  begin
    if Real(Pointer(Adr1)^) <> Real(Pointer(Adr2)^) then Exit;
    Inc(Adr1, 8);
    Inc(Adr2, 8);
  end;

  Result := True;
  if ResArray <> '' then NewOnes(ResArray, 1, 1);;
end;

function TWorkspace.SLoad(const Str: string): string;
begin
  Result := LoadArrayFromString(Str);
end;

function TWorkspace.SLoad(const Name, Values: string): string;
begin
  Result := LoadArrayFromString(Name + '=' + Values);
end;

function TWorkspace.CopyArray(SourWS: TWorkspace; const SourAr,
  DestAr: string): Integer;
var
  Rows, Cols, DestAdr: Integer;
  SourAdr: Pointer;
begin
  // Визначаємо розміри масиву
  if not SourWS.ArrayExists(SourAr) then DoError(matArrayNotFound);
  SourAdr := Pointer(SourWS.GetAddress(SourAr));
  SourWS.GetSize(SourAr, Rows, Cols);
  // Створюємо в поточній робочій області масив
  Result := NewArray(DestAr, Rows, Cols);
  DestAdr := GetAddress(Result);
  // Копіюємо масив
  Move(SourAdr^, Pointer(DestAdr)^, Rows * Cols shl 3);
end;

procedure TWorkspace.MoveArray(DestWS: TWorkspace; const SourAr, DestAr:
string);
var
  Ref: TMatrixData;
  Idx: Integer;
begin
  if SelfWS = DestWS then DoError(matWorkspaceError);
  Idx := GetIndex(SourAr);
  if not IsTrueName(DestAr) then DoError(matBadName);
  if (FRefList.IndexOf(Pointer(Idx)) <> -1) then DoError(matReturnError);
  Ref := FGetRef(SourAr);
  Ref.NameArray := DestAr;
  DestWS.FSetRef(Ref);
  FDelRef(SourAr);
  DestWS.FElemCount := DestWS.FElemCount + Ref.Rows * Ref.Cols;
end;

function TWorkspace.CopyRef(SourWS: TWorkspace; const SourAr,
  DestAr: string): Integer;
var
  Ref: TMatrixData;
begin
  if SelfWS = SourWS then DoError(matWorkspaceError);
  if (not IsTrueName(DestAr)) then DoError(matBadName);
  if not SourWS.ArrayExists(SourAr) then DoError(matArrayNotFound);
  Ref := SourWS.FGetRef(SourAr);
```

КбП-2023

```
  Ref.NameArray := DestAr;
  FSetRef(Ref);
  Result := GetIndex(DestAr);
  if FRefList.IndexOf(Pointer(Result)) = -1 then
    FRefList.Add(Pointer(Result));
end;

function TWorkspace.LoadArray(const SourAr, DestAr: string;
  ByRef: Boolean = True): Integer;
begin
  if ByRef then
    Result := CopyRef(ParentWS, SourAr, DestAr)
  else
    Result := CopyArray(ParentWS, SourAr, DestAr);
end;

procedure TWorkspace.ReturnArray(const SourAr, DestAr: string);
begin
  MoveArray(ParentWS, SourAr, DestAr);
end;

function TWorkspace.SumRows(SourAr, DestAr: string): string;
var
  sAdr, Idx, Rows, Cols, I, J: Integer;
  Temp: string;
  R: Extended;
begin
  sAdr := GetAddress(GetSize(SourAr, Rows, Cols));
  Result := CheckResAr(DestAr);
  Temp := GenName();
  Idx := NewArray(Temp, 1, Cols);
  for I := 1 to Cols do
  begin
    R := 0;
    for J := 1 to Rows do
      R := R + ElemFast[sAdr, J, I, Cols];
    Elem[Idx, 1, I] := R;
  end;
  RenameArray(Temp, DestAr);
end;

function TWorkspace.SumCols(SourAr, DestAr: string): string;
var
  Idx, Rows, Cols, I, J, sAdr: Integer;
  Temp: string;
  R: Extended;
begin
  Result := CheckResAr(DestAr);
  sAdr := GetAddress(GetSize(SourAr, Rows, Cols));
  Temp := GenName();
  Idx := NewArray(Temp, Rows, 1);
  for I := 1 to Rows do
  begin
    R := 0;
    for J := 1 to Cols do
      R := R + ElemFast[sAdr, I, J, Cols];
    Elem[Idx, I, 1] := R;
  end;
  RenameArray(Temp, DestAr);
end;

function TWorkspace.CalcFunc(const SourAr: string; DestAr: string; Func:
TRealFunc): string;
var
  ElCount, I, sIdx, dIdx: Integer;
  Rows, Cols: Integer;
  sAdr, dAdr: Integer;
  Temp: string;
begin
```

КБГЭ 2023

```
  Result := CheckResAr(DestAr);
  sIdx := GetSize(SourAr, Rows, Cols);
  ElCount := Rows * Cols;
  sAdr := GetAddress(sIdx);
  dIdx := Find(DestAr);
  if (dIdx <> -1) and (GetSize(dIdx) = ElCount) then dAdr := GetAddress(dIdx)
  else begin
    Temp := GenName();
    dAdr := GetAddress(NewArray(Temp, Rows, Cols))
  end;

  try
    for I := 1 to ElCount do
    begin
      Real(Pointer(dAdr)^) := Func(Real(Pointer(sAdr)^));
      Inc(dAdr, 8);
      Inc(sAdr, 8);
    end;
  except
    if Temp <> '' then Clear(Temp);
    DoError(matCalcFuncError);
  end;

  if Temp <> '' then RenameArray(Temp, DestAr) else
    if not SizeEqual(sIdx, dIdx) then Resize(DestAr, Rows, Cols);
end;

function TWorkspace.CalcFunc(const SourAr: string; DestAr: string; Func:
TExtendFunc): string;
var
  ElCount, I, sIdx, dIdx: Integer;
  Rows, Cols: Integer;
  sAdr, dAdr: Integer;
  Temp: string;
begin
  Result := CheckResAr(DestAr);
  sIdx := GetSize(SourAr, Rows, Cols);
  ElCount := Rows * Cols;
  sAdr := GetAddress(sIdx);
  dIdx := Find(DestAr);
  if (dIdx <> -1) and (GetSize(dIdx) = ElCount) then dAdr := GetAddress(dIdx)
  else begin
    Temp := GenName();
    dAdr := GetAddress(NewArray(Temp, Rows, Cols))
  end;

  try
    for I := 1 to ElCount do
    begin
      Real(Pointer(dAdr)^) := Func(Real(Pointer(sAdr)^));
      Inc(dAdr, 8);
      Inc(sAdr, 8);
    end;
  except
    if Temp <> '' then Clear(Temp);
    DoError(matCalcFuncError);
  end;

  if Temp <> '' then RenameArray(Temp, DestAr) else
    if not SizeEqual(sIdx, dIdx) then Resize(DestAr, Rows, Cols);
end;


function TWorkspace.CalcFunc2(const SourArray1, SourArray2: Variant; DestAr:
string;
  Func: TRealFunc2): string;
var
  I, Adr1, Adr2, dAdr, Rows, Cols, ElCount, dIdx, sIdx1, sIdx2, Size2: Integer;
  IsValue1, IsValue2: Boolean;
```

КоПЗ_2023

```
  Value1, Value2: Real;
  Tmp, SourAr1, SourAr2: string;
begin
  IsValue1 := False; IsValue2 := False;
  Value1 := 0; Value2 := 0;
  Adr1 := 0; Adr2 := 0;
  Result := CheckResAr(DestAr);

  if VarType(SourArray1) and varTypeMask = varString then
  begin
    SourAr1 := SourArray1;
    sIdx1 := GetIndex(SourAr1);
    Adr1 := GetAddress(GetSize(sIdx1, Rows, Cols));
    if (Rows = 1) and (Cols = 1) then
    begin
      IsValue1 := True;
      Value1 := Real(Pointer(Adr1)^);
    end;
  end else
  begin
    IsValue1 := True;
    Value1 := SourArray1;
  end;

  if VarType(SourArray2) and varTypeMask = varString then
  begin
    SourAr2 := SourArray2;
    sIdx2 := GetIndex(SourAr2);
    Adr2 := GetAddress(sIdx2);
    Size2 := GetSize(sIdx2);
    if Size2 = 1 then
    begin
      IsValue2 := True;
      Value2 := Real(Pointer(Adr2)^);
    end else
    begin
      if not IsValue1 and (Size2 <> Rows * Cols) then
        DoError(matArraysNotAgree);
      GetSize(sIdx2, Rows, Cols);
    end;
  end else
  begin
    IsValue2 := True;
    Value2 := SourArray2;
  end;

  try
    if IsValue1 and IsValue2 then
    begin
      NewFillAr(Result, 1, 1, Func(Value1, Value2), 0);
      Exit;
    end;
  except
    DoError(matCalcFuncError);
  end;

  ElCount := Rows * Cols;

  dIdx := Find(DestAr);
  if (dIdx > -1) and (GetSize(dIdx) = ElCount) then
     dAdr := GetAddress(dIdx) else
  begin
    Tmp := GenName();
    dIdx := NewArray(Tmp, Rows, Cols);
    dAdr := GetAddress(dIdx);
  end;

  try
```

RGTB - 2023

```
    if IsValue1 then
      for I := 1 to ElCount do
      begin
        Real(Pointer(dAdr)^) := Func(Value1, Real(Pointer(Adr2)^));
        Inc(dAdr, 8);
        Inc(Adr2, 8);
      end
    else
    if IsValue2 then
      for I := 1 to ElCount do
      begin
        Real(Pointer(dAdr)^) := Func(Real(Pointer(Adr1)^), Value2);
        Inc(dAdr, 8);
        Inc(Adr1, 8);
      end
    else
      for I := 1 to ElCount do
      begin
        Real(Pointer(dAdr)^) := Func(Real(Pointer(Adr1)^),
Real(Pointer(Adr2)^));
        Inc(dAdr, 8);
        Inc(Adr1, 8);
        Inc(Adr2, 8);
      end;

  except
    DoError(matCalcFuncError);
  end;

  if Tmp <> '' then RenameArray(Tmp, DestAr) else
    if (GetRows(dIdx) <> Rows) or (GetCols(dIdx) <> Cols) then
      Resize(DestAr, Rows, Cols);
end;

function TWorkspace.ArrayExists(const Name: string): Boolean;
begin
  Result := not (Find(Name) = -1);
end;

function TWorkspace.NewFillAr(const Name: string; const FirstValue, LastValue:
Real;
  ElemCount: Integer): string;
var
  Step: Extended;
  Idx: Integer;
begin
  // Необхідно так підібрати крок, щоб вийшло очікуване число точок
  Step := (LastValue - FirstValue) / (ElemCount - 1);
  Idx := NewArray(Name, 1, ElemCount);
  FillAr(Name, FirstValue, Step);
  // У результаті погрішності речовинних чисел потрібно уточнювати останній
елемент
  Elem[Idx, 1, ElemCount] := LastValue;
  Result := Name;
end;

function TWorkspace.GetArrayCount: Integer;
var
  I: Integer;
begin
  Result := 0;
  if FArrayList.Count = 0 then Exit;
  for I := 0 to FArrayList.Count - 1 do
    if FArrayList[I][1] <> '$' then Inc(Result);
end;

procedure TWorkspace.GetNameList(List: TStrings);
var
  I: Integer;
```

КОПЗ_2023

```
begin
  List.Clear;
  if FArrayList.Count = 0 then Exit;
  for I := 0 to FArrayList.Count - 1 do
    if FArrayList[I][1] <> '$' then
      List.Add(FArrayList[I]);
end;

function TWorkspace.GetSize(Idx: Integer; var Rows,
  Cols: Integer): Integer;
begin
  if not FArrayList.ArrayExistsForIndex(Idx) then DoError(matBadName);
  Rows := FArrayList.ItemsRows[Idx];
  Cols := FArrayList.ItemsCols[Idx];
  Result := Idx;
end;

function TWorkspace.ArrayExists(Idx: Integer): Boolean;
begin
  Result := FArrayList.ArrayExistsForIndex(Idx);
end;

function TWorkspace.SLoad(const Name: string; const Value: Real): string;
begin
  Result := NewFillAr(Name, 1, 1, Value, 0);
end;

function TWorkspace.InsertRows(const RowsAr: string; DestAr: string;
  Row: Integer): string;
var
  Rows1, Cols1, Rows2, Cols2, Rows, Cols, Adr1, Adr2, Adr, iTmp: Integer;
  Temp: string;
begin
  Result := CheckResAr(DestAr);
  if (Row < 1) then DoError(matBadRow);
  // Визначаємо розміри й адресу масиву, що вставляється
  Adr1 := GetAddress(GetSize(RowsAr, Rows1, Cols1));
  if (Find(DestAr) = -1) and (Row = 1) then
  begin
    CopyArray(RowsAr, DestAr);
    Exit;
  end;
  // Визначаємо розміри й адресу масиву DestAr
  Adr2 := GetAddress(GetSize(DestAr, Rows2, Cols2));
  if Row > Rows2 then DoError(matBadRow);
  if Cols1 <> Cols2 then DoError(matBadCol);

  Rows := Rows1 + Rows2;
  Cols := Cols1;
  // Створюємо результуючий масив
  Temp := GenName();
  Adr := GetAddress(NewArray(Temp, Rows, Cols, False));

  Cols := Cols shl 3;
  iTmp := Cols * (Row - 1);

  Move(Pointer(Adr2)^, Pointer(Adr)^, iTmp);
  Move(Pointer(Adr1)^, Pointer(Adr + iTmp)^, Rows1 * Cols1 shl 3);
  Move(Pointer(Adr2 + iTmp)^, Pointer(Adr + iTmp + Rows1 * Cols)^, Rows2 * Cols
- iTmp);

  RenameArray(Temp, DestAr);
end;

function TWorkspace.InsertCols(const ColsAr: string; DestAr: string;
  Col: Integer): string;
var
  Rows1, Cols1, Rows2, Cols2, Rows, Cols, Adr1, Adr2, Adr, I: Integer;
  Temp: string;
```

КПІ - 2023

```
begin
  Result := CheckResAr(DestAr);
  if (Col < 1) then DoError(matBadCol);
  // Визначаємо розміри й адресу масиву, що вставляється
  Adr1 := GetAddress(GetSize(ColsAr, Rows1, Cols1));
  if (Find(DestAr) = -1) and (Col = 1) then
  begin
    CopyArray(ColsAr, DestAr);
    Exit;
  end;
  // Визначаємо розміри й адресу масиву DestAr
  Adr2 := GetAddress(GetSize(DestAr, Rows2, Cols2));
  if (Col > Cols2) then DoError(matBadRow);
  if Rows1 <> Rows2 then DoError(matBadCol);

  Cols := Cols1 + Cols2;
  Rows := Rows1;
  // Створюємо результуючий масив
  Temp := GenName();
  Adr := GetAddress(NewArray(Temp, Rows, Cols, False));

  Col := (Col - 1) * 8;
  Cols1 := Cols1 * 8;
  Cols2 := Cols2 * 8;
  Cols := Cols * 8;
  for I := 1 to Rows do
  begin
    Move(Pointer(Adr2)^, Pointer(Adr)^, Col);
    Move(Pointer(Adr1)^, Pointer(Adr + Col)^, Cols1);
    Move(Pointer(Adr2 + Col)^, Pointer(Adr + Cols1 + Col)^, Cols2 - Col);
    Inc(Adr2, Cols2);
    Inc(Adr1, Cols1);
    Inc(Adr, Cols);
  end;

  RenameArray(Temp, DestAr);
end;


procedure TWorkspace.SendMatlabCommand(Str: string);
begin
  Matlab := CreateOleObject('Matlab.Application');
  Matlab.Execute(Str);
end;

function TWorkspace.FIsRef(Index: Integer): Boolean;
begin
  Result := (FRefList.IndexOf(Pointer(Index)) <> -1);
end;

function TWorkspace.SizeEqual(const Array1, Array2: string): Boolean;
var
  R1, C1, R2, C2: Integer;
begin
  GetSize(Array1, R1, C1);
  GetSize(Array2, R2, C2);
  Result := (R1 = R2) and (C1 = C2);
end;

function TWorkspace.SizeEqual(Idx1, Idx2: Integer): Boolean;
var
  R1, C1, R2, C2: Integer;
begin
  GetSize(Idx1, R1, C1);
  GetSize(Idx2, R2, C2);
  Result := (R1 = R2) and (C1 = C2);
end;

function TWorkspace.IsSingle(const Name: String): Boolean;
```

КОПЗ_2023

```
var
  R, C: Integer;
begin
  GetSize(Name, R, C);
  Result := (R = 1) and (C = 1);
end;

function TWorkspace.IsSingle(Idx: Integer): Boolean;
var
  R, C: Integer;
begin
  GetSize(Idx, R, C);
  Result := (R = 1) and (C = 1);
end;

function TWorkspace.IsSquare(const Name: String): Boolean;
var
  R, C: Integer;
begin
  GetSize(Name, R, C);
  Result := (R = C);
end;

function TWorkspace.IsSquare(Idx: Integer): Boolean;
var
  R, C: Integer;
begin
  GetSize(Idx, R, C);
  Result := (R = C);
end;


function TWorkspace.GetElemCount: Integer;
begin
  Result := FElemCount;
end;

function TWorkspace.GetSize(const Name: string): Integer;
var Rows, Cols: Integer;
begin
  GetSize(Name, Rows, Cols);
  Result := Rows * Cols;
end;

function TWorkspace.GetSize(Idx: Integer): Integer;
var Rows, Cols: Integer;
begin
  GetSize(Idx, Rows, Cols);
  Result := Rows * Cols;
end;

procedure TWorkspace.GetMinMax(Name: string; var MinVal, MaxVal: Real);
var
  I, Adr, Rows, Cols: Integer;
  R: Real;
begin
  Adr := GetAddress(GetSize(Name, Rows, Cols));
  // Запам'ятовуємо останній елемент
  MinVal := ElemFast[Adr, Rows, Cols, Cols];
  MaxVal := MinVal;

  for I := 1 to Rows * Cols - 1 do
  begin
    R := Real(Pointer(Adr)^);
    if R > MaxVal then MaxVal := R else
      if R < MinVal then MinVal := R;
    Inc(Adr, 8);
  end;
end;
```

КПЗ_2023

```
function TWorkspace.GetMean(SourAr, DestAr: string;
  Dim: TDim = dimCols): string;
var
  C: Integer;
begin
  Result := CheckResAr(DestAr);
  if Dim = dimCols then C := GetRows(SourAr) else C := GetCols(SourAr);
  if Dim = dimCols then SumRows(SourAr, DestAr) else SumCols(SourAr, DestAr);
  CalcFunc2(DestAr, C, DestAr, fncDiv);
end;

function TWorkspace.GetMinMax(SourAr, DestAr: string;
  Dim: TDim = dimRows): string;
var
  I, J, Rows, Cols, sAdr, dAdr: Integer;
  R, MinVal, MaxVal: Real;
  Temp: string;
begin
  Result := CheckResAr(DestAr);
  sAdr := GetAddress(GetSize(SourAr, Rows, Cols));

  Temp := GenName();
  if Dim = dimRows then
    dAdr := GetAddress(NewArray(Temp, Rows, 2, False))
  else
    dAdr := GetAddress(NewArray(Temp, 2, Cols, False));

  case Dim of
    dimRows:
      for I := 1 to Rows do
      begin
        MinVal := ElemFast[sAdr, I, 1, Cols];
        MaxVal := MinVal;
        for J := 2 to Cols do
        begin
          R := ElemFast[sAdr, I, J, Cols];
          if R > MaxVal then MaxVal := R else
            if R < MinVal then MinVal := R;
        end;
        ElemFast[dAdr, I, 1, 2] := MinVal;
        ElemFast[dAdr, I, 2, 2] := MaxVal;
      end;

      dimCols:
        for I := 1 to Cols do
        begin
          MinVal := ElemFast[sAdr, 1, I, Cols];
          MaxVal := MinVal;
          for J := 2 to Rows do
          begin
            R := ElemFast[sAdr, J, I, Cols];
            if R > MaxVal then MaxVal := R else
              if R < MinVal then MinVal := R;
          end;
          ElemFast[dAdr, 1, I, Cols] := MinVal;
          ElemFast[dAdr, 2, I, Cols] := MaxVal;
        end;
  end; // of Case
  RenameArray(Temp, DestAr);
end;

procedure TWorkspace.GetMin(SourAr, MinElems, Indexes: string;
  Dim: TDim = dimCols);
var
  Rows, Cols, sAdr, eAdr, iAdr, I, J, Idx: Integer;
  R, Min: Real;
  DoElem, DoIndexes: Boolean;
  TempEl, TempIdx: string;
```

КОПІЯ 2023

```
begin
  eAdr := 0; iAdr := 0;
  sAdr := GetAddress(GetSize(SourAr, Rows, Cols));
  DoElem := MinElems <> '';
  DoIndexes := Indexes <> '';
  if (not DoElem) and (not DoIndexes) then DoError(matBadParams);

  if DoElem then
  begin
    CheckResAr(MinElems, False);
    TempEl := GenName();
  end;

  if DoIndexes then
  begin
    CheckResAr(Indexes,  False);
    TempIdx := GenName;
  end;

  case Dim of

  dimCols:
    begin
      if DoElem then eAdr := GetAddress(NewArray(TempEl, 1, Cols, False));
      if DoIndexes then iAdr := GetAddress(NewArray(TempIdx, 1, Cols, False));
      for J := 1 to Cols do
      begin
        Min := ElemFast[sAdr, 1, J, Cols];
        Idx := 1;
        for I := Rows downto 2 do
        begin
          R := ElemFast[sAdr, I, J, Cols];
          if R < Min then
          begin
            Min := R;
            Idx := I;
          end;
        end;
        if DoElem then ElemFast[eAdr, 1, J, Cols] := Min;
        if DoIndexes then ElemFast[iAdr, 1, J, Cols] := Idx;
      end;
    end;

  dimRows:
    begin
      if DoElem then eAdr := GetAddress(NewArray(TempEl, Rows, 1, False));
      if DoIndexes then iAdr := GetAddress(NewArray(TempIdx, Rows, 1, False));
      for I := 1 to Rows do
      begin
        Min := ElemFast[sAdr, I, 1, Cols];
        Idx := 1;
        for J := Cols downto 2 do
        begin
          R := ElemFast[sAdr, I, J, Cols];
          if R < Min then
          begin
            Min := R;
            Idx := J;
          end;
        end;
        if DoElem then ElemFast[eAdr, I, 1, 1] := Min;
        if DoIndexes then ElemFast[iAdr, I, 1, 1] := Idx;
      end;
    end;

  end; // of Case
  if DoElem then RenameArray(TempEl, MinElems);
  if DoIndexes then RenameArray(TempIdx, Indexes);
end;
```

КбПЗ_2023

```
function TWorkspace.GetElemFast(Adr, Row, Col, ColCount: Integer): Real;
begin
  Result := Real(Pointer((Adr + ((Row - 1) * ColCount + Col - 1) shl 3))^);
end;

procedure TWorkspace.SetElemFast(Adr, Row, Col, ColCount: Integer;
  const Value: Real);
begin
  Real(Pointer((Adr + ((Row - 1) * ColCount + Col - 1) shl 3))^) := Value;
end;

procedure TWorkspace.GetMax(SourAr, MaxElems, Indexes: string;
  Dim: TDim = dimCols);
var
  Rows, Cols, sAdr, eAdr, iAdr, I, J, Idx: Integer;
  R, Max: Real;
  DoElem, DoIndexes: Boolean;
  TempEl, TempIdx: string;
begin
  eAdr := 0; iAdr := 0;
  sAdr := GetAddress(GetSize(SourAr, Rows, Cols));
  DoElem := MaxElems <> '';
  DoIndexes := Indexes <> '';
  if (not DoElem) and (not DoIndexes) then DoError(matBadParams);

  if DoElem then
  begin
    CheckResAr(MaxElems, False);
    TempEl := GenName();
  end;

  if DoIndexes then
  begin
    CheckResAr(Indexes,  False);
    TempIdx := GenName;
  end;

  case Dim of
    dimCols:
      begin
        if DoElem then eAdr := GetAddress(NewArray(TempEl, 1, Cols));
        if DoIndexes then iAdr := GetAddress(NewArray(TempIdx, 1, Cols));
        for J := 1 to Cols do
        begin
          Max := ElemFast[sAdr, 1, J, Cols];
          Idx := 1;
          for I := Rows downto 2 do
          begin
            R := ElemFast[sAdr, I, J, Cols];
            if R > Max then
            begin
              Max := R;
              Idx := I;
            end;
          end;
          if DoElem then ElemFast[eAdr, 1, J, Cols] := Max;
          if DoIndexes then ElemFast[iAdr, 1, J, Cols] := Idx;
        end;
      end;
    dimRows:
      begin
        if DoElem then eAdr := GetAddress(NewArray(TempEl, Rows, 1));
        if DoIndexes then iAdr := GetAddress(NewArray(TempIdx, Rows, 1));
        for I := 1 to Rows do
        begin
          Max := ElemFast[sAdr, I, 1, Cols];
          Idx := 1;
```

КбПЗ_2023

```
          for J := Cols downto 2 do
          begin
            R := ElemFast[sAdr, I, J, Cols];
            if R > Max then
            begin
              Max := R;
              Idx := J;
            end;
          end;
          if DoElem then ElemFast[eAdr, I, 1, 1] := Max;
          if DoIndexes then ElemFast[iAdr, I, 1, 1] := Idx;
        end;
      end;

  end;  // of Case
  if DoElem then RenameArray(TempEl, MaxElems);
  if DoIndexes then RenameArray(TempIdx, Indexes);
end;

function TWorkspace.GetElemType(const Name: string): TStandartType;
const
 MinOfType: array [1..6] of Int64 =
   (Low (Byte), Low (Shortint), Low (Word), Low (Short), Low (DWord), Low
(Integer));
 MaxOfType: array [1..6] of Int64 =
   (High(Byte), High(Shortint), High(Word), High(Short), High(DWord),
High(Integer));
var
  I, Rows, Cols, Adr: Integer;
  R, MaxEl, MinEl: Real;
begin
{
  Розпізнаються наступні типи:
  Real - stReal  (8 байт)
  Byte - stByte  (1 байт, від 0 до 255)
  ShortInt - stShortInt  (1 байт, від -128 до 127)
  Word - stWord (2 байти, від 0 до 65535)
  Short - stShort (2 байти, від -32768 до +32767)
  DWord - stDWord (4 байти, від 0 до 4 млрд)
  Integer - stInteger  (4 байти, від -2 млрд. до +2 млрд)

  Інші типи не розпізнаються, оскільки їх упевнено розпізнати неможливо
}
  Result := TStandartType(0);
  Adr := GetAddress(GetSize(Name, Rows, Cols));
  MinEl := Real(Pointer(Adr)^);
  MaxEl := MinEl;
  for I := 1 to Rows * Cols do
  begin
    R := Real(Pointer(Adr)^);
    if Frac(R) <> 0 then Exit;
    if R > MaxEl then MaxEl := R else
      if R < MinEl then MinEl := R;
    Inc(Adr, 8);
  end;

  for I := 1 to 6 do
    if (MinEl >= MinOfType[I]) and (MaxEl <= MaxOfType[I]) then
    begin
      Result := TStandartType(I);
      Break;
    end;
end;

procedure TWorkspace.FSaveArrayToBinFile(const FileName: string;
  ArIdx: Integer; AddToEnd: Boolean);
var
  cHead, cHead1: array[1..30] of Char;
  FS: TFileStream;
```

КОПІЯ 2023

```
  NewFile: Boolean;
  Rang: TPoint;
  WriteDataType, ReadDataType: Word;
  ArName: TNameArray;
  Name, NameAr: string;
  Adr, Rows, Cols, ByteCnt, ByteCnt1, J, ArSize: Integer;
  TByte: Byte;
  TWord: Word;
  TDWord: DWord;
label
  BeginWrite;
begin
  if FMStream = nil then FMStream := TMemoryStream.Create;
  Adr := GetAddress(GetSize(ArIdx, Rows, Cols));
  FArrayList.SetIndex(ArIdx); // Індексуємо масив, щоб його потім не шукати
  NameAr := GetName(ArIdx);

  cHead := matBinaryHeader;

  NewFile := not FileExists(FileName);

  WriteDataType := Byte(GetElemType(NameAr));

  case TStandartType(WriteDataType) of
    stByte, stShortint : ByteCnt := 1;
    stWord, stShort    : ByteCnt := 2;
    stDWord, stInteger : ByteCnt := 4;
    else
      ByteCnt := 8;
  end;
  FS := nil; // Ініціалізуємо об'єкт :)

  if NewFile then
  begin // СТВОРЕННЯ НОВОГО ФАЙЛУ
    FS := TFileStream.Create(FileName, fmCreate);
    FS.Write(cHead, 30);  // Пишемо заголовок
    ArName := StrToName(NameAr);
    FS.Write(ArName, 32); // Пишемо імені масиву
    Rang := Point(Rows, Cols);
    FS.Write(Rang, 8);    // Пишемо розміри масиву
    FS.Write(WriteDataType, 2);// Пишемо тип елементів
  end else if not AddToEnd then
  begin // ПОШУК В ІСНУЮЧОМУ ФАЙЛІ
    FS := TFileStream.Create(FileName, fmOpenReadWrite);
    FS.Read(cHead1, 30);  // Читаємо заголовок
    if cHead <> cHead1 then
    begin
      FS.Free;
      DoError(matBadFileFormat);
    end;
    while FS.Position < FS.Size do
    begin
      FS.Read(ArName, 32); // Зчитуємо імені масиву
      Name := ArName;
      while Name[Length(Name)] = #0 do Delete(Name, Length(Name), 1);
      FS.Read(Rang, 8);      // Зчитуємо розміри
      FS.Read(ReadDataType, 2); // Зчитуємо тип елементів
      case TStandartType(ReadDataType) of
        stByte, stShortint : ByteCnt1 := 1;
        stWord, stShort    : ByteCnt1 := 2;
        stDWord, stInteger : ByteCnt1 := 4;
        else ByteCnt1 := 8;
      end;
      if (Name = NameAr) and (Rows * Cols * ByteCnt =
          Rang.X * Rang.Y * ByteCnt1) then goto BeginWrite else
      begin
        // Якщо імена збіглися, то старе імені необхідно стерти
        if Name = NameAr then
        begin
```

КПІ 2023

```
        // Вертаємося назад на 42 байта
        FS.Seek(-42, soFromCurrent);
        // Записуємо спочатку імені символ '$'
        ArName[1] := '$';
        FS.Write(ArName, 32);
        FS.Seek(10, soFromCurrent); // Переміщаємося на попереднє місце
      end;
      // Пропускаємо елементи масиву
      FS.Seek(Rang.X * Rang.Y * ByteCnt1, soFromCurrent);
    end;
  end; // of While
  // Якщо масив не був знайдений, то пишемо необхідну інформацію
  ArName := StrToName(NameAr);
  FS.Write(ArName, 32); // Пишемо імені масиву
  Rang := Point(Rows, Cols);
  FS.Write(Rang, 8);    // Пишемо розміри масиву
  FS.Write(WriteDataType, 2);// Пишемо тип елементів
end; // if NewFile

// Якщо записуємо в кінець існуючого файлу, то:
if not NewFile and AddToEnd then
begin
  FS := TFileStream.Create(FileName, fmOpenWrite);
  FS.Seek(0, soFromEnd);
  ArName := StrToName(NameAr);
  FS.Write(ArName, 32); // Пишемо імені масиву
  Rang := Point(Rows, Cols);
  FS.Write(Rang, 8);    // Пишемо розміри масиву
  FS.Write(WriteDataType, 2);// Пишемо тип елементів
end;

BeginWrite:

//Якщо тип Real, то відразу записуємо масив у файл
if TStandartType(WriteDataType) = stReal then
  FS.Write(Pointer(Adr)^, Rows * Cols * 8) else
begin
  FMStream.SetSize(Rows * Cols * ByteCnt);
  FMStream.Seek(0, soFromBeginning);
  ArSize := Rows * Cols - 1;
  case TStandartType(WriteDataType) of
    stByte, stShortint :
            begin
            for J := 0 to ArSize do
            begin
              TByte := Trunc(Real(Pointer(Adr + (J shl 3))^));
              FMStream.Write(TByte, ByteCnt);
            end;
          end;
    stWord, stShort:
            begin
            for J := 0 to ArSize do
            begin
              TWord := Trunc(Real(Pointer(Adr + (J shl 3))^));
              FMStream.Write(TWord, ByteCnt);
            end;
          end;

    stDWord, stInteger:
            begin
            for J := 0 to ArSize do
            begin
              TDWord := Trunc(Real(Pointer(Adr + (J shl 3))^));
              FMStream.Write(TDWord, ByteCnt);
            end;
          end;

  end; // of case
  // Записуємо потік FMStream в FS
```

КОПЗ_2023

```
    FS.Write(Pointer(FMStream.Memory)^, Rows * Cols * ByteCnt);
  end;
  FMStream.Clear;
  FS.Free;
end;

function TWorkspace.CalcProd(SourAr, DestAr: string; Dim: TDim): string;
var
  Rows, Cols, I, J, sAdr, dAdr: Integer;
  R: Real;
  Tmp: string;
begin
  Result := CheckResAr(DestAr);
  sAdr := GetAddress(GetSize(SourAr, Rows, Cols)); // Визначаємо адресу й
розміри
  Tmp := GenName();
  try
    case Dim of
      dimRows:
              begin
                dAdr := GetAddress(NewArray(Tmp, Rows, 1));
                for I := 1 to Rows do
                begin
                  R := 1;
                  for J := 1 to Cols do
                    R := R * ElemFast[sAdr, I, J, Cols];
                  ElemFast[dAdr, I, 1, 1] := R
                end;
              end;

      dimCols:
              begin
                dAdr := GetAddress(NewArray(Tmp, 1, Cols));
                for J := 1 to Cols do
                begin
                  R := 1;
                  for I := 1 to Rows do
                    R := R * ElemFast[sAdr, I, J, Cols];
                  ElemFast[dAdr, 1, J, Cols] := R
                end;
              end;
    end;  // of Case
  except
    Clear(Tmp);
    DoError('CalcProd ERROR!');
  end;
  RenameArray(Tmp, DestAr);
end;

function TWorkspace.GetMinOrMax(Name: string; Num: Integer; Dim: TDim;
  ReturnMax: Boolean; var Index: Integer): Real;
var
  I, Rows, Cols, Adr: Integer;
begin
  Index := 0;
  if ReturnMax then Result := -MaxDouble else Result := MaxDouble;
  Adr := GetAddress(GetSize(Name, Rows, Cols));

  case Dim of
    dimRows:
      begin
        if (Num < 1) or (Num > Rows) then DoError(matBadRow);
        if ReturnMax then
        begin
          for I := Cols downto 1 do
            if ElemFast[Adr, Num, I, Cols] > Result then
            begin
              Index := I;
              Result := ElemFast[Adr, Num, I, Cols];
```

КОПІЯ_2023

```
            end
          end else
          begin
            for I := Cols downto 1 do
              if ElemFast[Adr, Num, I, Cols] < Result then
              begin
                Index := I;
                Result := ElemFast[Adr, Num, I, Cols];
              end
          end;
        end;

    dimCols:
      begin
        if (Num < 1) or (Num > Cols) then DoError(matBadCol);
        if ReturnMax then
        begin
          for I := Rows downto 1 do
            if ElemFast[Adr, I, Num, Cols] > Result then
            begin
              Index := I;
              Result := ElemFast[Adr, I, Num, Cols];
            end
        end else
        begin
          for I := Rows downto 1 do
            if ElemFast[Adr, I, Num, Cols] < Result then
            begin
              Index := I;
              Result := ElemFast[Adr, I, Num, Cols];
            end
        end;
      end;
  end; // of Case
end;

function TWorkspace.NewZeros(const Name: string; Rows, Cols: Integer): string;
begin
  NewArray(Name, Rows, Cols, True);
  Result := Name;
end;

function TWorkspace.NewOnes(const Name: string; Rows, Cols: Integer): string;
begin
  Result := NewFillAr(Name, Rows, Cols, 1, 0);
end;

procedure TWorkspace.LoadFromTextFile(const FileName: string;
  ArName: string);
begin
  if ArName = '' then FLoadFromTextFile(FileName) else
    FLoadArrayFromTextFile(FileName, ArName);
end;

procedure TWorkspace.SaveToBinFile(const FileName: string; ArName: string = '');
var
  I: Integer;
begin
  if ArName = '' then
  begin
    if FileExists(FileName) then DeleteFile(FileName);
    if FArrayList.Count > 0 then
      for I := 0 to FArrayList.Count - 1 do
        if IsTrueName(FArrayList[I]) and (GetSize(I) <> 0) then
          FSaveArrayToBinFile(FileName, I);
  end else
    FSaveArrayToBinFile(FileName, GetIndex(ArName), False);
end;
```

КОПІЯ 2023

```
initialization
  Base := TWorkspace.Create(matBaseWorkspace);
finalization
  Base.Free;
end.
```

КбПЗ_2023

EKG.pas - створення ЕКГ

```
unit EKG;


interface



uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Signals, Math, Printers, Matrix;


const
  SNoSignals = 'Сигнали ЕКГ не виявлені!';


type
  TEKGFrame = class(TFrame)
    ScrollBox1: TScrollBox;
    Image: TImage;
    Panel1: TPanel;
    PrintDialog1: TPrintDialog;
    panLine1: TPanel;
    panLine2: TPanel;
    procedure panLine1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
  private
    FGrid: TBitmap; // Кеш міліметрівки
    FGridColor: Byte;  // Колір сітки (за замовчуванням 50 - рудий)
    FPrintGridColor: TColor; // Колір роздрукованою сітки
    FDinEKGColor: TColor; // Цим кольором відображається ЕКГ у процесі знімання
    FEKGColor: TColor; // Колір графіків
    FPeaksColor: TColor; // Колір ліній, що виділяють піки ЕКГ

    // Параметри вихідних сигналів
    FAmplitude, FFreq: Integer; // Амплітуда й частота

    // Параметри відображення
    Fmm_on_m, Fmm_on_sec: Integer; // Число міліметрів на мілівольт  і
міліметрів у секунду



    FInterval: Integer; // Відстань між сусідніми кривими
    FGridVisible: Boolean; // Визначає видимість сітки

    FWS: TWorkspace; // постійна робоча область
    FNames: array of string; // Імена графіків
    FHead, FFoot, FPageInfo: string; // Текстова інформація, що
                                     // виводиться на друк
    FProcessCnt: Integer; // Лічильник  використовуємий при зніманні сигналів
ЕКГ
    FPixel: Integer; // Зберігає номер останнього засвіченого пікселя
    FLimit: Integer; // Ліміт використовуємий при зніманні ЕКГ
    FCoef: Real; // Коефіцієнт, показуючий, скільки точок проецирується на 1
піксель
    FSignCount: Integer; // Число сигналів, показуємі у процесі знімання ЕКГ
    Fidx: Integer; // Індексування масиву Matri
    FIsProcess: Boolean; // Дорівнює True, якщо виконується процес знімання ЕКГ
    FPosLine1: Integer;  // Позиція 1 лінії
    FPosLine2: Integer;  // Позиція 2 лінії
    FLinesVisible: Boolean;  // Визначає, чи видні лінії
    FStepVisible: Boolean; // Визначає видимість каліброваного імпульсу
    FList: TStringList; // Зберігає текст повідомлення MessageText
    procedure SetFont(); // Установлює шрифт тексту
    procedure FindOptimalInterval(); // розраховує інтервал між кривими так
```

КОТЭ-2023

```
                                        // щоб їх усі можна було роздрукувати


    // Промальовування кривих
    procedure _DrawImage(DrawHead: Boolean = True; DrawFoot: Boolean = True);
    procedure DrawGrid(AllCanvas: Boolean = True);
    procedure SetPosLine1(const Value: Integer);
    procedure SetPosLine2(const Value: Integer);
    procedure SetSignalsText(const Value: string);
    function GetSignalsText: string; // Промальовування сітки

  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;



    //ФУНКЦІЇ ДЛЯ ВІЗУАЛІЗАЦІЇ КРИВИХ ЕКГ
    // Завантаження сигналів ЕКГ із робочою області WS
    procedure LoadSignals(WS: TWorkspace; DataArray: string; PeaksAr: string =
'');
    // Видаляє сигнали ЕКГ із робочою області. Викликає метод DrawImage().
    // Звичайно служить для виводу повідомлення про помилку  MessageText
    procedure ClearSignals();
    // Текст повідомлення, виведеного на міліметрівку. Текст виводиться тільки
    // при відсутності сигналів ЕКГ. Звичайно це повідомлення про помилку,
    // при якій має сенс прекратити відображення кривих
    property MessageText: string read GetSignalsText write SetSignalsText;
    // Установлює імена масивів
    procedure SetNames(Names: array of string);
    // Промалювання графіків ЕКГ (і міліметрівки при необхідності)
    procedure DrawImage();
    // Печатка кривих на принтері
    procedure Print();



    // ФУНКЦІЇ ДЛЯ ЗНІМАННЯ СИГНАЛІВ ЕКГ
    // Підготовляє сітку до відображення що знімається ЕКГ. SignCount -
кількість відведень
    procedure StartProcess(SignCount: Integer);
    // Процедура візуалізує процес знімання ЕКГ. Щораз у неї подається
    // масив знятих точок (по одній точці для кожного відведення)
    procedure AddPoints(const NewPoints: array of Real);
    // Процедура зупиняє процес знімання ЕКГ. Інші процедури
    // будуть недоступні, поки ви не зупините процес знімання ЕКГ
    procedure StopProcess();
    // Повертає True, якщо відбувається знімання ЕКГ
    property IsProcess: Boolean read FIsProcess;



    // НАСТРОЮВАННЯ ПАРАМЕТРІВ

// Установка параметрів
    procedure SetParams(Amplitude, Freq, mm_on_m, mm_on_sec: Integer);

// Колір сітки на екрані монітора
    property GridColor: Byte read FGridColor write FGridColor;

 // Колір сітки при печатці
    property PrintGridColor: TColor read FPrintGridColor write FPrintGridColor;

  // Колір кривих
    property EKGColor: TColor read FEKGColor write FEKGColor;

  // Колір кривих у процесі знімання ЕКГ
    property DinEKGColor: TColor read FDinEKGColor write FDinEKGColor;
```

КПЗ - 2023

```
  // Колір ліній, що виділяють піки ЕКГ
    property PeaksColor: TColor read FPeaksColor write FPeaksColor;

 // Інтервал між кривими
    property Interval: Integer read FInterval write FInterval;

 // Включає або виключає міліметрівку
    property GridVisible: Boolean read FGridVisible write FGridVisible;

// Текст заголовка для друку
    property HeadText: string read FHead write FHead;

 // Додаткова інформація для друку (унизу екрана)
    property FootText: string read FFoot write FFoot;

 // Інформація про сторінку для друку
    property PageInfo: string read FPageInfo write FPageInfo;

 // Позиція першою лінії
    property PosLine1: Integer read FPosLine1 write SetPosLine1;

  // Позиція другою лінії
    property PosLine2: Integer read FPosLine2 write SetPosLine2;

   // Визначає видимість лінії (споконвічно лінії не видні)
    property LinesVisible: Boolean read FLinesVisible write FLinesVisible;

  // Визначає видимість каліброваного імпульсу
    property StepVisible: Boolean read FStepVisible write FStepVisible;
  end;




implementation

{$R *.dfm}

{ TEKGFrame }


constructor TEKGFrame.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FGridColor := 50;// $50A0FF;
  FEKGColor  := clBlack;
  FPrintGridColor := clBlack;
  FDinEKGColor := clBlack;
  FPeaksColor := clRed;
  FAmplitude := 1000;
  FFreq      := 500 ;
  Fmm_on_m  := 10  ;
  Fmm_on_sec := 50  ;
  FInterval  := 15  ;
  GridVisible:= True;
  PosLine1 := 10;
  PosLine2 := 50;
  FLinesVisible := False;
  FStepVisible := True;
  FList := TStringList.Create;
  MessageText := SNoSignals;
  FGrid := TBitmap.Create;
  FGrid.PixelFormat := pf8bit;


  // Створюємо статичну робочу область
  FWS := TWorkspace.Create('Криві ЕКГ');
```

КбПЗ_2023

```
end;


destructor TEKGFrame.Destroy;
begin
  FWS.Free;
  FList.Free;
  FGrid.Free;
  inherited;
end;


procedure TEKGFrame.DrawGrid(AllCanvas: Boolean = True);
var
  I, J: Integer;
  P: PByteArray;
begin
  if (FGrid.Width = Image.Width) and (FGrid.Height = Image.Height) and
     (Image.Width * Image.Height > 0) then
  begin
    // Значить сітка минулого разу вже рисувалася
    // Просто копіюємо готову сітку
    Image.Canvas.Draw(0, 0, FGrid);
  end else
  begin
    // Сітка ще не рисувалася. Малюємо її
    FGrid.Width := Image.Width;
    FGrid.Height := Image.Height;
    for I := 0 to FGrid.Height - 1 do
    begin
      P := FGrid.ScanLine[I];
      for J := 0 to FGrid.Width - 1 do
      begin
        if ((I mod 4 = 0) and (J mod 4 = 0)) or ((J mod 20 = 0) and (I mod 2 =
0)) or
           ((I mod 20 = 0) and (J mod 2 = 0)) then
          P[J] := FGridColor;
      end;
    end;
    Image.Canvas.Draw(0, 0, FGrid);
  end;


  // Обрізаємо при необхідності все зайве:
  if not AllCanvas then
    with Image do
    begin
      Canvas.Pen.Color := clWhite;
      Canvas.Brush.Color := clWhite;
      Canvas.Rectangle(0, 0, Width, 20);
      Canvas.Rectangle(0, Height - 20, Width, Height);
    end;
end;


procedure TEKGFrame._DrawImage(DrawHead: Boolean = True; DrawFoot: Boolean =
True);
var
  I, J, eIdx, h, Rows, Cols, X: Integer;
  S: string;
begin
  if FIsProcess then Exit;
  with FWS do begin
    if not ArrayExists('EKG') then begin
      panLine1.Visible := False;
      panLine2.Visible := False;
      Image.Width := ScrollBox1.Width - 5;
      Image.Height := ScrollBox1.Height - 5;
      Image.Picture := nil;
```

КбПЗ – 2023

```
      if GridVisible then DrawGrid;
      SetFont;
      for I := 0 to FList.Count - 1 do
        Image.Canvas.TextOut(10,
          Round(10 + I * Image.Canvas.TextHeight('A')), FList[I]);
      Exit;
    end;


    // Виконуємо стиск кривих:
    mScaleSignals(FWS, 'EKG', 'EKG1',
Round(GetCols('EKG')/(FFreq/Fmm_on_sec/4)));
    eIdx := GetSize('EKG1', Rows, Cols);



    // Виконуємо перерахування амплітуди

    for I := 1 to Rows do
      for J := 1 to Cols do
        Elem[eIdx, I, J] := -Elem[eIdx, I, J]*4*Fmm_on_m/FAmplitude;


    // Установлюємо висоту
    Image.Height := (Rows - 1) * FInterval * 4 + 2 * (Fmm_on_m * 4 + 20);

 // Установлюємо ширину
    if DrawHead and DrawFoot then
      Image.Width := 1024 else
      Image.Width := Cols + 20;

 // Малюємо сітку
    Image.Picture := nil;
    if GridVisible then DrawGrid(not(DrawHead or DrawFoot));

    Image.Canvas.Pen.Color := FEKGColor;
    h := Fmm_on_m * 4 + 20;
    // Друкуємо заголовок
    SetFont;
    if DrawHead then
      Image.Canvas.TextOut(1, 1, FHead);

 // Друкуємо текст унизу
    if DrawFoot then begin
      S := IntToStr(Fmm_on_sec) + ' мм/з ' + IntToStr(Fmm_on_m) + ' мм/мВ ' +
FFoot;
      Image.Canvas.TextOut(1, Image.Height - 15, S);

      Image.Canvas.TextOut(Min(Image.Width, 1024) -
        Image.Canvas.TextWidth(FPageInfo), Image.Height - 15, FPageInfo);
    end;
    for I := 1 to Rows do begin
      if FStepVisible then
      begin

   // Малюємо сходинку
        Image.Canvas.MoveTo(5, h);
        Image.Canvas.LineTo(8, h);
        Image.Canvas.LineTo(8, h - Fmm_on_m * 4);
        Image.Canvas.LineTo(12, h - Fmm_on_m * 4);
        Image.Canvas.LineTo(12, h);
        Image.Canvas.LineTo(16, h);
      end;
      Image.Canvas.MoveTo(1 + 20, Round(Elem[eIdx, I, 1] + h));
      for J := 2 to Cols do
        Image.Canvas.LineTo(J + 20, Round(Elem[eIdx, I, J] + h));

  // Виводимо назву кривої
      if I <= Length(FNames) then begin
```

КПЗ – 2023

```
        SetFont;
        if FStepVisible then
          Image.Canvas.TextOut(29, h, FNames[I - 1]) else
          Image.Canvas.TextOut(1, h - 15, FNames[I - 1]);
      end;
      h := h + FInterval * 4;
    end;
    if ArrayExists('Peaks') then
    begin
      Image.Canvas.Pen.Color := FPeaksColor;
      H := Image.Height;
      for I := 1 to GetCols('Peaks') do
      begin
        X := Round((Fmm_on_sec * Elem['Peaks', 1, I] / FFreq) * 4) + 20;
        Image.Canvas.MoveTo(X, 20);
        Image.Canvas.LineTo(X, H - 20);
      end;
    end;
  end;
  with panLine1 do
  begin
    Width := 2;
    Top := Image.Top;
    Height := Image.Height;
    Visible := FLinesVisible;
    PosLine1 := PosLine1;
  end;
  with panLine2 do
  begin
    Width := 2;
    Top := Image.Top;
    Height := Image.Height;
    Visible := FLinesVisible;
    PosLine2 := PosLine2;
  end;

end;


procedure TEKGFrame.FindOptimalInterval;
begin
  FInterval := Round((720 - 2 * (Fmm_on_m * 4 + 20)) /
    (4 * (FWS.GetRows('EKG') - 1)));
end;


procedure TEKGFrame.LoadSignals(WS: TWorkspace; DataArray: string; PeaksAr:
string = '');
begin
  if WS.ArrayExists(DataArray) then
  begin
    FWS.CopyArray(WS, DataArray, 'EKG');
    if PeaksAr = '' then
      FWS.Clear('Peaks')
    else
      FWS.CopyArray(WS, PeaksAr, 'Peaks');
  end else
  begin
    MessageText := SNoSignals;
    DrawImage();
  end;
end;


procedure TEKGFrame.Print;
var
  X1, Y1, X2, Y2: Integer;
  Points, Points: Real;
  Col: TColor;
```

КоПЗ_2023

```
  Interv: Integer;
begin
  if FIsProcess then Exit;
  if FWS.ArrayExists('Peaks') then FWS.RenameArray('Peaks', 'Peaks1');

 // Запам'ятовуємо поточний інтервал
  Interv := FInterval;
  FindOptimalInterval;

 // Запам'ятовуємо колір
  Col := FGridColor;
  FGridColor := FPrintGridColor;
  _DrawImage;
  FGridColor := Col;

  if PrintDialog1.Execute then begin
    Printer.Orientation := poLandscape;
    Printer.BeginDoc;
    Printer.Canvas.Refresh;
    Points:=GetDeviceCaps(Printer.Canvas.Handle, LOGPIXELSX) / 101.538;
    Points:=GetDeviceCaps(Printer.Canvas.Handle, LOGPIXELSY) / 101.538;

    X1:=round((Printer.PageWidth - Min(Image.Picture.Bitmap.Width, 1024) *
Points) / 2) + 80;
    Y1:=round((Printer.PageHeight - Min(Image.Picture.Bitmap.Height, 768) *
Points) / 2);
    X2:=round(X1 + Min(Image.Picture.Bitmap.Width, 1024)*Points);
    Y2:=round(Y1 + Min(Image.Picture.Bitmap.Height, 768)*Points);

    Printer.Canvas.CopyRect(Rect(X1, Y1, X2, Y2),Image.Picture.Bitmap.Canvas,
    Rect(0,0,Min(Image.Picture.Bitmap.Width, 1024),
Min(Image.Picture.Bitmap.Height, 768)));
    Printer.EndDoc;
  end;
  FInterval := Interv;
  if FWS.ArrayExists('Peaks1') then FWS.RenameArray('Peaks1', 'Peaks');
  _DrawImage(False, False);
end;

procedure TEKGFrame.SetFont;
begin
  Image.Canvas.Font.Color := clBlack;
  Image.Canvas.Font.Name := 'MS Sans Serif';
  Image.Canvas.Font.Size := 8;
  Image.Canvas.Font.Style := [fsBold];
end;


procedure TEKGFrame.SetParams(Amplitude, Freq, mm_on_m, mm_on_sec: Integer);
begin
  if Amplitude > 0 then FAmplitude := Amplitude;
  if Freq      > 0 then FFreq      := Freq;
  if mm_on_m  > 0 then Fmm_on_m  := mm_on_m;
  if mm_on_sec > 0 then Fmm_on_sec := mm_on_sec;
end;
procedure TEKGFrame.DrawImage;
begin
 _DrawImage(False, False);
end;


procedure TEKGFrame.SetNames(Names: array of string);
var
  I, Len: Integer;
begin
  Len := Length(Names);
  SetLength(FNames, Len);
  for I := 0 to Len - 1 do
    FNames[I] := Names[I];
```

КПЗ – 2023

```
end;


procedure TEKGFrame.StartProcess(SignCount: Integer);
begin
  if FIsProcess then Exit;
  panLine1.Visible := False;
  panLine2.Visible := False;
  if ScrollBox1.VertScrollBar.IsScrollBarVisible then
    Image.Width := ScrollBox1.Width - 25 else
    Image.Width := ScrollBox1.Width - 5;
  FSignCount := SignCount;
  Image.Height := (SignCount - 1) * FInterval * 4 + 2 * (Fmm_on_m * 4);
  Image.Picture := nil;
  if GridVisible then DrawGrid();
  FProcessCnt := 0;
  FPixel := 0;
  FLimit := Image.Width + 1;
  Fidx := FWS.GetIndex(FWS.NewFillAr('ProcessEKG', SignCount,
    Image.Width + 2, -1000000, 0));
  // Визначаємо коефіцієнт
  FCoef := FFreq/Fmm_on_sec/4;
  FIsProcess := True;
end;


procedure TEKGFrame.AddPoints(const NewPoints: array of Real);
var
  I, K, h: Integer;
begin
  if not FIsProcess then Exit;
  Inc(FProcessCnt); // Знаходимо номер точки
  K := Trunc(FProcessCnt/FCoef); // Визначаємо номер пікселя
  if K = FPixel then Exit; // Якщо той же піксель, то виходимо

  h := Fmm_on_m * 4;
  Image.Canvas.Pen.Mode := pmXor;
  Image.Canvas.Pen.Width := 1;
  Image.Canvas.Pen.Color := FDinEKGColor xor clWhite;
  for I := 1 to FSignCount do begin
    // Стираємо старий піксель
    Image.Canvas.MoveTo(K, Round(FWS.Elem[Fidx, I, K]));
    Image.Canvas.LineTo(K + 1, Round(FWS.Elem[Fidx, I, K + 1]));
    // Установлюємо нове значення
    FWS.Elem[Fidx, I, K] := -NewPoints[I - 1] * 4 * Fmm_on_m/FAmplitude + h;
    // Зафарбовуємо піксель
    if K >= 2 then begin
      Image.Canvas.MoveTo(K - 1, Round(FWS.Elem[Fidx, I, K - 1]));
      Image.Canvas.LineTo(K, Round(FWS.Elem[Fidx, I, K]));
    end;
    h := h + FInterval * 4; // Розраховуємо крок
  end;
  FPixel := K; // Запам'ятовуємо оброблений піксель
  if FPixel >= FLimit then begin
    FPixel := 0;
    FProcessCnt := 0;
  end;
end;


procedure TEKGFrame.StopProcess;
begin
  FIsProcess := False;
end;


procedure TEKGFrame.panLine1MouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
```

КБТЗ_2023

```
  if not (ssLeft in Shift) then Exit;
  TPanel(Sender).Left := TPanel(Sender).Left + X;
  // Щодо лівої панелі:
  if panLine1.Left < Image.Left + 20 then panLine1.Left := Image.Left + 20;
  if (panLine1.Left > panLine2.Left - 5) then
    panLine2.Left := panLine1.Left - 10;
  if panLine1.Left >= (Image.Width + Image.Left) then
    panLine1.Left := (Image.Width + Image.Left) - 10;
// Щодо правої панелі:
  if panLine2.Left >= (Image.Width + Image.Left) then
    panLine2.Left := (Image.Width + Image.Left) - 1;
  if (panLine2.Left < panLine1.Left + 5) then
    panLine2.Left := panLine1.Left + 10;
  FPosLine1 := Round((FFreq * (panLine1.Left - Image.Left - 20)) / (4 *
Fmm_on_sec));
  FPosLine2 := Round((FFreq * (panLine2.Left - Image.Left - 20)) / (4 *
Fmm_on_sec));
  if FPosLine1 < 1 then FPosLine1 := 1;
  if FPosLine2 < 1 then FPosLine2 := 1;
end;


procedure TEKGFrame.SetPosLine1(const Value: Integer);
begin
  FPosLine1 := Value;
  if FPosLine1 < 1 then FPosLine1 := 1;
  panLine1.Left := Round((Fmm_on_sec * Value / FFreq) * 4) + 20 + Image.Left;
  if FPosLine1 > FPosLine2 - 10 then SetPosLine2(FPosLine1 + 20);
end;


procedure TEKGFrame.SetPosLine2(const Value: Integer);
begin
  FPosLine2 := Value;
  if FPosLine2 < 1 then FPosLine2 := 1;
  panLine2.Left := Round((Fmm_on_sec * Value / FFreq) * 4) + 20 + Image.Left;
  if FPosLine2 < FPosLine1 + 10 then SetPosLine1(FPosLine2 + 20);
end;


procedure TEKGFrame.ClearSignals;
begin
  if not FIsProcess then
  begin
    FWS.Clear;
    DrawImage;
  end;
end;


procedure TEKGFrame.SetSignalsText(const Value: string);
begin
  FList.Text := Value;
end;


function TEKGFrame.GetSignalsText: string;
begin
  Result := FList.Text;
end;


end.
```

КОПЭ 2023

about.pas - довідка

```
unit frmAbout;

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;
    Memo1: TMemo;
    OKButton: TButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var  AboutBox: TAboutBox;

implementation
{$R *.dfm}
procedure TAboutForm.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи моніторингу
біомедичної інформації');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Лисенко І.А.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Гаєвський Віктор Сергійович');
Memo1.Lines.Add('                гр. КН-22МЗ');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;
procedure TAboutForm.Button1Click(Sender: TObject);
begin
AboutForm.Close;
end;
end.
```

КДТЗ_2023