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 управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв”

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

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

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

Макеєву Андрію Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 36-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Макеєв А.В. Дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Метою розробки є дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Об'єктом дослідження є процес управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Предметом дослідження є методи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Методи дослідження базуються на методах теорії графів і булевої алгебри , методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

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

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

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

Програму розроблено в середовищі RAD Studio Delphi 10.

Ключові слова: комп'ютерна інженерія, управління технологічною безпекою

ABSTRACT

Makeiev A.V. Research and software implementation of the technological safety management system based on probabilistic structural and logical models of industrial hazards. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the technological safety management system based on probabilistic structural and logical models of industrial hazards.

The purpose of the development is research and software implementation of the technological safety management system based on probabilistic structural and logical models of industrial hazards.

The object of the study is the process of technological safety management based on probabilistic structural and logical models of industrial hazards.

The subject of the research is methods of technological safety management based on probabilistic structural and logical models of industrial hazards.

Research methods are based on the methods of graph theory and Boolean algebra, methods of mathematical statistics, software development methods.

The result of the work is the software implementation of the technological safety management system based on probabilistic structural and logical models of production hazards.

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

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

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

The program was developed in the RAD Studio Delphi 10 environment.

Keywords: computer engineering, technological security management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ALARA – As Low As Reasonably Achievable – настільки низько, наскільки це можливо – принцип прийнятного ризику

FMEA – Failure Modes and Effects Analysis – аналіз видів і наслідків відмов

HCR – Human Cognitive Reliability – Надійність людини як функція його здібностей

HEP – Human Error Probability – Імовірність помилки людини

IRRAS – Integrated Reliability and Risk Analysis System – інтегральна надійність та аналіз ризику систем – пакет прикладних програм

NUREG – Nuclear Regulatory – Стандарт США (керівництво) в атомній промисловості

NRC – Nuclear Regulatory Commission – Комісія з ядерного регулювання США

OHSAS – Occupational Health and Safety Assessment Series – Система менеджменту охорони здоров'я та безпеки персоналу

RRR – Risk Reduction Ratio – коефіцієнт зменшення ризику

RRI – Risk Reduction Interval – інтервал зменшення ризику

THERP – Technique for Human Error Rate Prediction – Методика аналізу помилок людини в техніці (США)

АВНВ – аналіз видів і наслідків відмов

АЕС, АС – атомна станція

АП – аварійна послідовність

БД – база даних

БП – базисна подія

ВП – вихідна подія

ГНД – галузевий нормативний документ

ДВ – дерево відмов

ДП – дерево подій

ЗІЗ – засоби індивідуального захисту

ІАБ (російською – ВАБ) – імовірнісний аналіз безпеки

ІДУ ЦЗ – Інститут державного управління у сфері цивільного захисту

ІП – ініціююча подія

ІПМЕ – Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України

КС – кінцевий стан

ЛЧ – людський чинник

МНС – Міністерство надзвичайних ситуацій

МОЗ – Міністерство охорони здоров'я

МОН – Міністерство освіти і науки

МП – мінімальні перерізи

ННДІОП – Національний науково-дослідний інститут охорони праці

НД – нормативний документ

НС – надзвичайна ситуація

НП – небажана подія

НВ – небезпечний випадок

ОП – охорона праці

ОПН – об'єкт підвищеної небезпеки

ПНО – потенційно небезпечний об'єкт

ПЛАС – план ліквідації аварійних станів

ПП – прикладна програма

ППР – планово-попереджальний ремонт

ПТЕ – правила технічної експлуатації

РОП – ризик орієнтований підхід

СБ – система безпеки

СТП – стандарт підприємства

СУОП – система управління охороною праці підприємства

ТО – технічне обслуговування

ФБ – функція безпеки

Укр.НДІ ПБ – Український науково-дослідний інститут пожежної безпеки

ВСТУП

Актуальність теми. Однією з найбільш важливих областей життєдіяльності людства, що викликає незмінний інтерес, є безпека та її дослідження фахівцями різних галузей. Технологічна безпека як основа екологічної безпеки повинна регулюватися державою в першу чергу. Технологічні ризики створюють основне антропогенне навантаження на екологічні системи, тому ризики, розрахунки та запобігання ризику, управління ризиками стають важливими державними задачами. В основу чинного законодавства покладені (скопійовані) західні положення про безпеку, що ґрунтуються на визначенні ризику (Директива СОВЕЗО-II та інші), але нормативні документи нижчого рівня потребують суттєвого доопрацювання. В результаті маємо невизначеності з нормативною базою, що містить забагато протиріч та робить неможливим в такий спосіб регулювання безпеки [5].

Основна робота з регулювання безпеки – декларування, тобто розрахунок технологічного ризику недостатньо науково і методично описана, тому декларування, як основної процедури регулювання безпеки, на цей час фактично не існує, як наслідок – не впроваджено ринкові механізми регулювання безпеки, з чого Держава несе великі збитки [19].

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

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

– Огляд існуючих систем управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

– Дослідження системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

– Програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Об'єктом дослідження є процес управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Предметом дослідження є методи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Методи дослідження базуються на методах теорії графів і булевої алгебри , методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

– Розроблено вітчизняний продукт управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

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 управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв. Традиційно в нашій країні проводиться політика забезпечення 100% безпеки виробництв, яка закріплена в існуючій нормативно-правовій базі [1-23], що є наслідком застарілих соціальних систем. Загальний перелік нормативних документів (НД) з безпеки складає більш ніж 3000 документів, кожний з яких містить до тисячі різних вимог. Ці вимоги складались протягом десятків років за принципом заборони небезпечних умов виробництв, при яких виникали нещасні випадки та травми, або аварії з негативним впливом на довкілля. Іноді вимоги в різних документах неузгоджені, або навіть суперечливі між собою, на що можливо навести достатньо прикладів.

Усі вимоги безпеки, що чинні на підприємстві, встановлюються в стандартах підприємства (СТП) та інструкціях з охорони праці, які складають систему управління охороною праці (СУОП). СТП та інструкції враховують вимоги галузевих нормативних документів (ГНД), які, в свою чергу, розробляються на основі загальних нормативно-правових актів України та міжнародних стандартів. Усі названі документи повинні враховувати вимоги Законів України з безпеки, які, як правило, враховують вимоги із Міжнародних договорів та угод. Оцінка стану безпеки підприємства здійснюється на основі складних критеріїв, що встановлюються в галузі на основі перевірок виконання всіх вимог. Врахувати всі вимоги практично дуже складно, тому оцінка стану безпеки підприємства та його вплив на довкілля часто носить суб'єктивний характер.

Безпека виробництв тісно пов'язана з безпечністю машин та конструкцій і безпекою систем взагалі [26, 27]. Конструктори та технологи при розробці як безпечних конструкцій, так і безпечного виробництва до недавнього часу орієнтувалися виключно на метод спроб та помилок. Такий підхід виправдовував себе тоді, коли системи та конструкції були відносно простими у порівнянні з теперішніми. Наприклад, в авіаційній промисловості рішення подібних проблем часто полягало у використанні метода "літай, усувай недоліки – літай". Літаки розроблялися на основі вже існуючих або відомих технологій. Потім вони випробувались і експлуатувалися до того моменту, доки не виникала відмова чи аварія. Якщо встановлювалося, що причиною аварії були помилки у конструкції, а не помилки людини (пілота), то ці конструктивні недоліки усувалися і літаки літали знову. Цей метод – постфактум безпеки конструкцій непогано працював, коли літаки літали повільно та низько і будувалися з фанери дроту й тканини. Однак, оскільки системи ставали все більш складними, а можливості літаків, такі як швидкість, маневреність і кількість посадочних місць зростали, збільшувалась ймовірність значних наслідків аварії, що ставало неприпустимим.

Перший метод створення правил з безпеки після того, як нещасний випадок або аварія сталися, призвів до тієї великої множини правил з безпеки, що існують у вітчизняному законодавстві. Така політика існувала свого часу у більшості промислово розвинених країн. Так, у прийнятому у 1972 р. конгресом США законі "Про охорону праці" була встановлена вимога забезпечити промислові машини захистом їх операторів від усіх небезпек, пов'язаних із працею на цих машинах. Відгуком на цю вимогу з’явились роботи в яких доводилось, що коли всі небезпеки усунені та контрольовані, то неможливо організувати процес виробництва взагалі [13]. З часом ця вимога була змінена іншою, згідно якої для захисту оператора та працівників, що знаходяться в зоні дії машини, від небезпек повинна бути запроваджена одна або більше систем механічного захисту такі як захисні бар’єри, дворучне вимкнення, електронний захист тощо [14].

Але ці методи в нашій країні впроваджуються дуже повільно. Традиційно все виробництво поділено за списками умов праці, та в умовах з тяжкими і особливо тяжкими умовами праці видається додаткова зарплатня. Нещодавно проведена класифікація умов праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу [9], що надало змогу розподілу виробництва не тільки за списками умов праці але і за класами професійного ризику [3]. Відповідно до закону про обов'язкове страхування проводиться страхування життя на виробництві, що є одним з ринкових механізмів управління.

Розподіл виробництва по ступеню безпечності можливо проілюструвати за допомогою уявлень теорії множин, рисунок 1.1. Якщо множину можливих дій людини при виробленні будь-якого продукту **i** або **j** визначимо як $\mathbf{M_i}$, то в будь-якому випадку її можливо розподілити на дві частини: $\mathbf{M_{i1}}$ та $\mathbf{M_{i2}}$, де $\mathbf{M_{i1}}$ – множина можливих небезпечних дій (технологічних операцій виробництва), $\mathbf{M_{i2}}$ – множина можливих безпечних дій, а $\mathbf{M_3}$ – це множина технологічних операцій виробництва, що контролюється за нормативними документами.

Відповідно наведеному рисунку виробництво **i** можливо визначити як безпечне, та навпаки виробництво **j** – небезпечне виробництво, більшість дії повинна контролюватися за спеціальними процедурами.

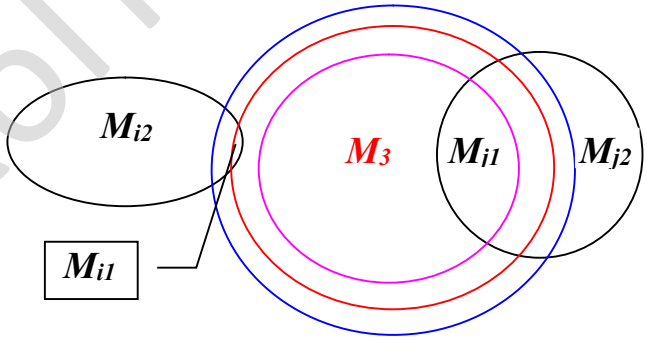


Рисунок 1.1 – Небезпечне виробництво

З досвіду відомо, що множина $\mathbf{M_3}$ – має нечітку, розмиту межу. Внутрішня її частка, зменшення множини $\mathbf{M_3}$ відбувається зі свідомих або несвідомих порушень існуючих заборон НД, а зовнішня частина межі – це ті можливі

правила, що заборонять небезпечні дії виробництва, які ще не виявлені на даний час. Звичайно, дії виробництва, що контролюються більш складні, потребують спеціального навчання, знань і умінь, і відповідних додаткових витрат часу для виконавця та коштів для підприємства. З цієї причини такі технологічні операції часто свідомо порушуються як з боку виконавця, так і з боку роботодавця. Психологічним виправданням для цього служить випадковість небезпечних дій, які були покладені в основу відповідних заборон [4]. Обставини, що здійснились колись, можливо дуже поодинокі, їх повторна реалізація є малоймовірною подією, що розуміють виконавці, тому і здійснюють свідоме порушення в надії на краще, що є частиною національного менталітету. Адже ж, ще на початку минулого століття російський основоположник системного аналізу як окремої науки, відзначав безтурботність та непередбаченість поведінки людини слов'янського походження [6]. Взагалі, оскільки будь-які виробничі дії, з урахуванням всіх обставин, є такими, що неможливий їх абсолютний повтор, множину можливих виробничих дій людини – $\mathbf{M}_i$ потрібно вважати випадковою та нескінченною множиною, з чого слідує:

– Неможливо зробити чіткі правила на всі порушення, завжди знайдеться якась незаборонена правилами модифікація відповідних дій людини, що призводить до небажаного результату;

– Неможливо заборонами усунути небезпеки, тобто модель забезпечення 100% безпеки нежиттєздатна за своєю суттю.

Оцінка стану безпеки виробництв у моделі забезпечення 100% безпеки здійснювалась та й зараз визначається за фактом аварій за рік.

Наприклад, для оцінки стану безпеки у галузі економіки України в останні роки використовується інтегральний показник професійного ризику виробництва – $I_{ге}$, який визначається як відношення витрат у минулому календарному році у галузі відшкодування шкоди потерпілим на виробництві до фактичних витрат на оплату праці у минулому календарному році в цій галузі економіки за формулою:

$$I_{ге} = (ВШ_{ге}/ ВОП_{ге})*100\% \quad (1.1)$$

де:

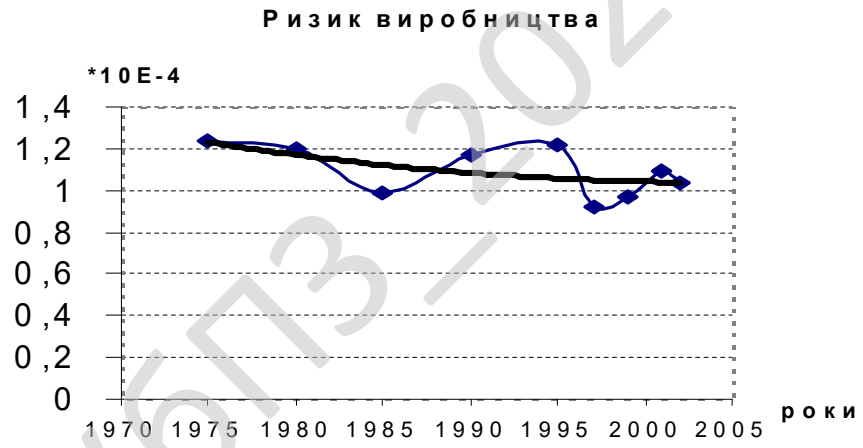
$ВШ_{ге}$ сума відшкодування шкоди потерпілим на виробництві, яка нарахована у минулому календарному році в галузі економіки;

$ВОП_{ге}$ – фактичні витрати на оплату праці у минулому календарному році в цій галузі економіки.

Формула 1.1 оцінює стан охорони праці не безпосередньо, а за допомогою підрахувань витрат на оплату праці, що призводить до великих невизначеностей результату з багатьох причин.

Прямі статистичні оцінки ризику смертності як відношення кількості загиблих до кількості працюючих, які можна знайти в звітах охорони праці та літературі [7] свідчать про його відносне стабільне значення протягом багатьох років, незважаючи на постійні заходи по зменшенню ризику та постійні витрати на безпеку, рисунок 1.2.

Рисунок 1.2 – Ризик виробництва в Україні



Як бачимо, ризик виробництва залишається дуже високим протягом 35 років та фактично не змінюється, що свідчить про низьку ефективність існуючої системи управління безпекою.

1.2 Область застосування

Областю застосування системи є управління технологічною безпекою. З появою безпеки систем як науки метод забезпечення безпеки і надійності систем

перетворився на метод гарантії безпеки систем, який названо "визначення, аналіз та виключення" [14]. Тобто, подальший розвиток у більшості промислово розвинених країн призвів до використання ризик орієнтованого підходу і методів, що раніше застосовувалися у сфері визначення надійності систем [26, 27]. На цей час у Західній Європі та США чинний Міжнародний стандарт OHSAS (Occupational Health and Safety Assessment Series) 18001-99 "Система управління охорони здоров'я та безпеки персоналу. Вимоги." [4]. Він визначає, що Система управління охороною праці (СУОП) – це загальна система менеджменту підприємства, яка забезпечує управління ризиками у сфері охорони праці та здоров'я працівників. Існує також серія стандартів ISO 14000 з екологічного керування [14-20], які широко поширені та призначені для управління довкіллям як складової загального менеджменту підприємства, без визначення числових значень ризику й безпеки. Міжнародні стандарти, що поширюються на управління довкіллям, призначені для забезпечення виробництв елементами ефективної системи управлення довкіллям, які можуть бути об’єднані з іншими елементами адміністративного управління, з метою сприяння організаціям у справі досягнення екологічних й економічних цілей. Тобто, аналіз закордонного досвіду управління екологічною безпекою свідчить, що регулювання її у промислово розвинених країнах здійснюється на підставі оцінки виникнення ризиків та управління довкіллям, як складової загального менеджменту. При цьому підході виходять з того, що безпека – це не абсолютна відсутність небезпеки, а "відсутність неприпустимого ризику, пов’язаного з можливістю завдання будь-якої шкоди", так як це за аналогією визначає ДСТУ 2156-93 [21].

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Програмний комплекс «БЕЗПЕКА»

Програмний комплекс (ПК) має блокову структуру, кожний блок може функціонувати як окремо, так і збиратися в необхідні конфігурації й може поставлятися в будь-якому составі. При цьому бази даних, уведені в програмні модулі зберігаються й не залежать від часу поставки наступних додаткових блоків тобто не залежать від поставки додаткових блоків. Бази даних побудовані на єдиній класифікаційній основі тобто користувач може замовити й працювати в якому те окремому програмному модулі й у випадку додавання надалі додаткових модулів, блоків йому не треба повторно вводити дані по підприємству. При цьому бази даних зберігаються по роках і й прив'язані до конкретних виробничих площадок, відособленим підрозділам, небезпечним об'єктам, об'єктам негативного впливу на навколишнє середовище (ОНВ), розташованих на конкретних територіях (ОКАТО, ОКТМО).

ПК забезпечує створення систем моніторингу безпеки об'єктів техносфери (СМБОТ), систем підтримки прийняття управлінських рішень (СППР) у ЧС, ведення баз даних виробничого контролю з використанням засобів вимірів, що забезпечують безперервні автоматичні виміри виділюваних у навколишнє середовище забруднюючих речовин, їхню систематизацію, обробку, автоматичний перенесення даних при формуванні інвентаризації джерел ЗВ, автоматичний перенесення даних з інвентаризації в розрахункові модулі при формуванні тому ПДВ, формування зведених даних при контролі на границі СЗЗ,

на території зони впливу об'єктів, перенос фактичних даних при розрахунку плати за НВОС.

ПК дозволяє створити вертикально-вертикально-інтегровану корпоративну інформаційну систему керування в області промислової, пожежної, екологічної, експлуатаційної, енергетичної безпеки, у тому числі вертикально-вертикально-інтегровані системи при обігу із твердими комунальними (побутовими) відходами, відходами виробництва, як на рівні компаній, так і на рівні суб'єктів РФ, округів, із забезпеченням прийому-передачі інформації в XML-форматі по всій інформаційній вертикалі керування.

ПК дозволяє в будь-який момент часу самостійно сформувати юридичною особою, суб'єктам малого й середнього підприємництва (СМСП), без залучення сторонніх організацій, з поточної єдиної бази дані підприємства, всю необхідну документацію в області охорони навколишнього середовища (ОС), промислової, пожежної, експлуатаційної безпеки, у тому числі, ПНООЛР, тім ПДВ, ПДВ, сформувати технічний звіт про незмінність технологічних процесів, робити розрахунок плати за НВОС, сформувати план ЛАРН, ПЛАС, ПЛА, декларацію пожежної безпеки, декларацію промислової безпеки, паспорт небезпечного об'єкта, технічні паспорти будинків, енергетичні паспорти, забезпечити побудова сценаріїв аварійних ситуацій, дерев відмов, зробити розрахунок умовних ймовірностей, розрахунок ризиків, побудувати поля потенційного ризику, розрахувати збиток, побудувати F/N, F/G діаграми, сформувати статзвіт 2 ТП (повітря), 2ТП (водгосп), 2ТП (відходи), 2ТП (рекультивація), 4-ОС, 5-ГР, 6-ГР, 70-ТП, 71-ТП, 1-ЛС, 2-ЛС і ін. з вивантаженням на веб-портали, Держстат, самостійно провести розрахунок класу небезпеки КОСНВОС, сформувати паспорти відходів, сформувати документи для одержання комплексних дозволів відповідно до вимог 219-ФЗ, сформувати інші документи відповідно до вимог НПА.

ПК дозволяє створити системи СЕМ, автоматизовані системи керування (АСУ) по ISO 14001.

Поставляється без ключів фізичного захисту.

Формат прийому-передачі інформації – XML.

Система поставляється безкоштовно.

Оплата виробляється тільки за річне ІТС (аналогічно ІТС "Гарант", "Консультант").

ІТС містить у собі постійну актуалізацію ПК відповідно до змін нормативно-правових документів, постійну актуалізацію бази нормативно-правових документів, консультації, допомога "гарячої" лінії, якщо буде потреба перегляд розрахунків з наданням допомоги по виправленню помилок.

У програму в рамках ІТС постійно додаються нові розрахункові методики, РД, СП, СТО, ПБ, постійно актуалізуються довідники, класифікатори.

Програми мають відкриті формати міжмашиного обміну (XML), відповідають вимогам ISO, дозволяють здійснювати прийом-передачу "вивантажень" у будь-які інші програми.

Програми мають універсальний редактор звітів, містять Програму-конструктор "Формування проекту нормативів ПДВ", Програму-конструктор "Проект нормативів утворення й розміщення відходів (ПНООЛР)", Програму-конструктор "Формування технічного звіту про незмінність технологічних процесів" і ін.

Склад ПК «БЕЗПЕКА»:

1. Блок програм "Документообіг, адміністрування документообігу".
2. Блок програм "Загальні відомості про територіальний орган".
3. Блок програм "Відомості про підприємство".
4. Блок програм "Атмосферне повітря".
5. Блок програм "Відходи виробництва й споживання".
6. Блок програм "Водоспоживання й водовідведення".
7. Блок програм "Надрокористування".
8. Блок програм "Землекористування".
9. Блок програм "Лісокористування".

10. Блок програм "Трубопроводи, нафтопроводи, водоводи".

11. Блок програм "Ризик".

12. Блок програм "Аварії".

13. Блок програм "Промислова безпека".

14. Блок програм "Пожежна безпека".

15. Блок програм "Енергетична безпека".

16. Блок програм "Експлуатаційна безпека об'єктів нерухомості".

17. Блок програм "Виробництво й споживання енергоносіїв".

18. Блок програм "Збиток".

19. Блок програм "Рекультивація".

20. Блок програм "Виробничий контроль".

21. Блок програм "Моніторинг".

22. Блок програм "Радіаційний, дозиметричний і радіохімічний контроль".

23. Блок програм "Розрахунок плати за негативний вплив на навколишнє середовище".

24. Блок програм "Фінансово-економічний моніторинг".

25. Блок програм "Зведені дані по регіоні по формах 2-тп".

26. Блок програм "Кліматичні, географічні характеристики району розташування підприємства".

27. Блок програм "Екологічний паспорт природокористувача".

28. Блок програм "Контроль виконання приписань, облік перевірок".

29. Блок програм "Аналіз даних".

30. Блок програм "Документи для постановки на облік".

31. Блок програм "Класифікатори (159)".

32. Блок програм "Довідники (100)".

33. Блок програм "Джерела інформації".

34. Блок програм "Адміністрування й налаштування".

35. Блок програм "Конструктор".

36. Інші програмні блоки.

Призначення програмного комплексу ПК "БЕЗПЕКА":

– забезпечення єдиного інформаційно-технологічного ланцюжка збору, зберігання, перевірки вірогідності подання й обробки інформації в області охорони навколишнього середовища (ОС), екологічної, пожежної, промислової, експлуатаційної безпеки по всій інформаційній вертикалі;

– самостійне формування юридичною особою, без залучення сторонніх організацій, всієї необхідної документації в області охорони навколишнього середовища (ОС), промислової, пожежної, експлуатаційної безпеки, у тому числі, ПНООЛР, томів ПДВ, ПДВ, планів ЛАРН, ПЛАС, ПЛА, декларацій пожежної безпеки, декларацій промислової безпеки, паспортів небезпечних об'єктів, побудова сценаріїв аварійних ситуацій, дерев відмов, розрахунок умовних імовірностей, розрахунок ризиків, побудова полів потенційного ризику, розрахунок збитку, побудова F/N, F/G діаграм і формуванні інших документів відповідно до вимог НПД;

– ведення баз даних виробничого контролю з використанням засобів вимірів, що забезпечують безперервні автоматичні виміри виділюваних у навколишнє середовище забруднюючих речовин, їхню систематизацію, обробку, автоматичний перенесення даних при формуванні інвентаризації джерел ЗВ, автоматичний перенесення даних з інвентаризації в розрахункові модулі при формуванні тому ПДВ, формування зведених даних при контролі на границі СЗЗ, на території зони впливу об'єктів, перенос фактичних даних при розрахунку плати за НВОС;

– створення корпоративних інформаційних систем керування при обігу із твердими комунальними (побутовими) відходами, відходами виробництва, із забезпеченням прийому-передачі інформації в XML-форматі по всій вертикалі керування;

– створення систем СЕМ, автоматизованих систем керування (АСУ) по ISO 14001;

– забезпечення прийому-передачі інформації в XML-форматі.

Функціональні можливості й переваги ПК "БЕЗПЕКА"

Система поставляється в рамках інформаційно-технологічного супроводу (ІТС) без обмеження кількості користувачів у юридичної особи, має зручний і зрозумілий інтерфейс із можливістю роботи з використанням для зручності декількох моніторів.

Состав: багаторівневий з обмеженням кількості підрозділів і об'єктів негативного впливу, відповідно до замовленої конфігурації на ІТС.

Програмний комплекс "Безпека" (ПК "Безпека") використовує клієнт-серверну технологію доступу до даних і дозволяє побудувати розподілену систему збору й обробки інформації, використовуючи локальні обчислювальні мережі, глобальну мережу Інтернет і бездротові мережі передачі даних, як окремо, так і в різних сполученнях.

Мережний варіант комплексу поставляється без обмеження кількості одночасно працюючих користувачів.

Програмний комплекс має блокову структуру, кожний блок може функціонувати як окремо, так і збиратися в необхідні конфігурації й може поставлятися в будь-якому составі. При цьому бази даних, уведені в програмні модулі зберігаються й не залежать від часу поставки наступних додаткових блоків тобто не залежать від поставки додаткових блоків. Бази даних побудовані на єдиній класифікаційній основі тобто користувач може замовити й працювати в якому те окремому програмному модулі й у випадку додавання надалі додаткових модулів, блоків йому не треба повторно вводити дані по підприємству. При цьому бази даних зберігаються по роках і й прив'язані до конкретних виробничих площадок, розташованих на конкретних територіях (ОКАТО).

Поставляється без ключів фізичного захисту.

Формат прийому-передачі інформації – XML.

Система поставляється безкоштовно.

Оплата виробляється тільки за річне ІТС (аналогічно ІТС "Гарант", "Консультант").

ІТС містить у собі постійну актуалізацію ПК відповідно до змін нормативно-правових документів, постійну актуалізацію бази нормативно-правових документів, консультації, допомога "гарячої" лінії, якщо буде потреба перегляд розрахунків з наданням допомоги по виправленню помилок.

У програму в рамках ІТС постійно додаються нові розрахункові методики, РД, СП, СТО, ПБ, постійно актуалізуються довідники, класифікатори.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

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

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

д) розробити рекомендації по організаційних та методичних заходах, які

забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Випадковою величиною називаємо величину, що характеризується упорядкованим набором

$$X \equiv (X1, X2, \ldots .) \tag{3.1}$$

дійсних чисел (значень) X1, X2, ..., що дає значення багатовимірної випадкової величини $\bar{x} \equiv (x_1, x_2,...)$. Кожна з дійсних величин x_1, x_2,... сама є випадковою величиною. Якщо множина елементарних подій, пов'язана з даним іспитом, позначена випадковою величиною **x** чи $\bar{\mathbf{x}}$**,** то імовірності випадкових подій однозначно описуються розподілом ймовірностей цієї випадкової величини.

Розподіл дійсної випадкової величини **x** задається її функцією розподілу

$$\Phi_x(X) \equiv \Phi(x) \equiv P\{ x < X \}. \tag{3.2}$$

Розподіл багатомірної випадкової величини $\bar{x} = (x1, x2, ...)$ задається функцією спільного розподілу:

$$\Phi_X(X_1, X_2,..,) \equiv \Phi(X_1, X_2,...,) \equiv P\{ x_1 < X_1, x_2 < X_2, ... \}. \tag{3.3}$$

Похідну функції розподілу називаємо щільністю розподілу ймовірностей величини **x:**

$$\varphi_x(x) \equiv \varphi(x) \equiv \frac{d\Phi}{dx} \tag{3.4}$$

Неперервна випадкова величина **X** розподілена нормально з центром ξ і дисперсією σ^2 (або нормально з параметрами ξ і σ^2), якщо:

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\xi}{\sigma}\right)^2} \equiv \frac{1}{\sigma}\varphi_u\left(\frac{x-\xi}{\sigma}\right) \tag{3.5}$$

$$\Phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{x-\xi}{\sigma}\right)^2} dx = \Phi_u\left(\frac{x-\xi}{\sigma}\right) \equiv \frac{1}{2}\left[1 + erf\left(\frac{x-\xi}{\sigma\sqrt{2}}\right)\right] \tag{3.6}$$

Розподіл стандартизованої нормальної величини (нормального відхилення) $u = \frac{x - \xi}{\sigma}$ дається формулами:

$$\varphi_u(u) \equiv \frac{1}{2\pi} \ell^{-\frac{1}{2}u^2} \text{ – щільність нормального розподілу} \quad (3.7)$$

і нормальна функція розподілу (інтеграл ймовірностей)

$$\Phi_u(u) \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-\frac{1}{2}u^2} du \equiv \left[1 + erf\left(\frac{u}{\sqrt{2}}\right)\right], \quad (3.8)$$

де **erf (z)–** функція помилок.

Ясно, що $\mathbf{M_u = 0; D_u = 1}$

Випадковий процес є випадковою функцією x(t) від незалежної змінної **t.** Кожен іспит дає визначену функцію X(t), що називаємо реалізацією процесу чи вибірковою функцією. Випадковий процес можна розглядати, або як сукупність реалізацій процесу X(t), або як сукупність випадкових величин, що залежать від параметру t. При цьому повинні бути задані розподіли ймовірностей систем випадкових величин $x_1 = x(t_1)$, $x_2 = x(t_2)$,... (вибіркових значень) для будь-якої скінченної множини значень t_1, t_2,... (вибіркових моментів). Випадковий процес може бути дискретним чи неперервним, якщо дискретний чи неперервний розподіл величин $x(t_1)$, $x(t_2)$,... для кожної скінченої множині t_1, t_2,... Процес називаємо випадковою послідовністю (процесом із дискретним часом), якщо незалежна змінна може приймати тільки злічену множину значень.

Таке визначення випадкового процесу припускає існування розподілу ймовірностей для функціонального простору його реалізацій. Кожна реалізація x(t) = X(t) утворить елементарну подію (вибіркову точку у функціональному просторі).

В ІАБ незалежною змінною **t** служить час, а величина x(t) означає стан фізичної системи. Велика частина випадкових процесів, розглянутих у ІАБ, є окремим класом випадкових процесів, що називають марковськими процесами.

Марковський процес – випадковий процес, при якому стан системи після моменту часу t залежить тільки від її стану в цей момент і не залежить від проходження процесу до моменту часу t.

Перехідна (перехідна імовірнісна функція) **P(t,x,s,E) –** імовірність того, що система, знаходячись у момент часу t у стані x, у момент часу s > t потрапить в один із станів множини **E.**

Якщо x – скінчений зліченний чи фазовий простір, а час неперервний, тоді перехідна ймовірність визначається функціями $\mathbf{P_{ij}(t,s)}$, рівними умовній імовірності того, що система знаходиться в j-ому стані в момент s, якщо в момент t вона знаходилася в i-ому стані.

Дискретний чи неперервний випадковий процес **x(t)** називаємо (простим) марковським процесом, якщо для кожної скінченої множини $t_1 < t_2 <_{..} < t_{n-1} < t_n$

$$p(X_{n,}\, t_n \mid X_{1,}\, t_1;...;\, X_{n-1,}\, t_{n-1}) = p(X_{n,}\, t_n \mid X_{n-1,}\, t_{n-1}) \qquad (3.9)$$

або

$$\varphi(X_{n,}\, t_n \mid X_{1,} t_{1;..;}\, X_{n-1,}\, t_{n-1}) = \varphi(X_{n,}\, t_n \mid X_{n-1,}\, t_{n-1})$$

відповідно. Якщо дано $x(t_{n-1}) = X_{n-1,}$ то знання $x(t_{n-2}), x(t_{n-3}),..$ не додає ніякої нової інформації про розподіл x (t_n).

Марковський процес цілком визначається своїм розподілом ймовірностей другого порядку і, отже, може бути заданий розподілами ймовірностей першого порядку і ймовірностей переходу

$$p(X_2,t_2 \mid x,\, t) \text{ або } \varphi(X_2,t_2 \mid x,\, t) \quad (t < t_2). \qquad (3.10)$$

Марковські випадкові послідовності часто називають ланцюгами Маркова. Кожен чисто випадковий процес є марковським. Багато фізичних процесів можуть бути описані як марковські. Важливий клас задач полягає у пошуку функцій (3.10) по їх ″початкових значеннях″ при $t = t_1$

З визначальної властивості (3.9) марковського процесу випливають рівняння Колмогорова-Смолуховського-Чепмена.

$$p(X_2,t_2 \mid X_1,t_1) = \sum_{x(t)} p(X_2,t_2 \mid x,\, t)\; p(x,\, t \mid X_1,t_1) \quad (t_1 \le t \le t_2)$$

або ∞

$$\varphi(X_2,t_2 \mid X_1,t_1) = \int \varphi(X_2,t_2 \mid x,t)\, \varphi(x,t \mid X_1,t_1)\, dx \quad (t_1 \le t \le t_2) \qquad (3.11)$$

$-\infty$

Рівняння (3.11) є різницеве рівняння першого порядку, що може бути вирішене щодо невідомої функції (3.10) незалежної змінної t, якщо задана $p(x, t \mid X_1,t_1)$ або $\varphi(x,t \mid X_1,t_1)$. Якщо $p_{(1)}(X_1,t_1)$ чи $\varphi_{(1)}(X_1,t_1)$ відомі, то марковський процес цілком визначений при всіх $t > t_1$.

Для марковського процесу з безперервним часом, імовірність настання небажаної події залежить від часу:

$$P(t) = 1 - e^{-\lambda t}, \qquad (3.12)$$

де λ – інтенсивність небажаної події (характеризує швидкість його настання).

При цьому середній проміжок часу між небажаними подіями $\Delta \bar{t} = \frac{1}{\lambda}$, дисперсія визначається аналогічно: $\sigma = \frac{1}{\lambda}$.

Таблиця 3.1 – Типи випадкових процесів

Варіант	Вид функції	Вид аргументу	Назва випадкового процесу
1	Дискретна	Дискретний	Дискретна випадкова послідовність – марковський ланцюг
2	Дискретна	Безперервний	Дискретний випадковий процес
3	Безперервна	Дискретний	Безперервна випадкова послідовність
4	Безперервна	Безперервний	Безперервний випадковий процес
5	Безперервна + Дискретна	Безперервний	Змішаний випадковий процес

Якщо тимчасові проміжки між переходами зі стану в стан не підкоряються показниковому законові (3.12), випадковий процес називається полумарковським, його дослідження потребує інших методів. У загальному випадку всі можливі

випадкові процеси можна представити, розглядаючи можливі сполучення аргументу і функції, що описує процес, таблиця 3.1.

Випадкові процеси описуються також за допомогою стохастичних матриць [9] або за допомогою графів [5]. Дерева подій являють собою одну з різновидів графа.

Марковська модель безпеки ПНО (АЕС)

Оцінка ймовірностей (кінцевих станів) потенційно небезпечного об’єкту при виникненні вихідних подій (ВП) (аварій) виконується по викладеній нижче методиці, заснованій на марковській моделі безпеки, схема якої стосовно найбільш складного об'єкта – АЕС, наведена на рисунку 3.1 [37,57].

На рисунку 3.1 введені наступні позначення:

– символами “0” – “l.0” позначені безпечні стани, тобто такі стани, у яких значення радіаційних показників не перевищують установлені норми. У ці стани входять при нормальній експлуатації – “0” і стані – “1.0” – “l.0”, що реалізуються при проектному протіканні аварій, коли виникнення вихідних подій супроводжується виконанням усіх необхідних функцій безпеки;

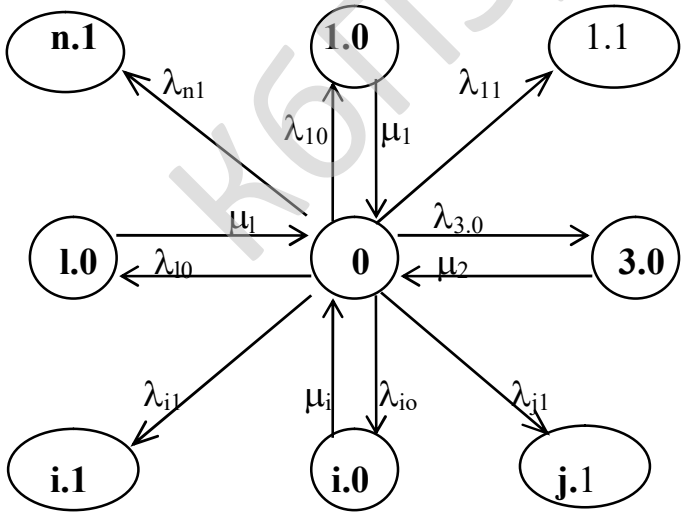


Рисунок 3.1 – Марковська модель безпеки

– символами “1.1” – “n.1” позначені стани з порушенням безпеки, що виникають або внаслідок запроектних ВП, або внаслідок невиконання однієї чи декількох функцій безпеки при проектних ВП, n – повне число аварійних послідовностей з порушенням безпеки;

– символами $\lambda_{10} - \lambda_{n1}$ позначені інтенсивності переходів у відповідні стани;

– символами $\mu_1 - \mu_l$ позначені інтенсивності переходів із безпечних станів при проектних ВП у стани нормальної експлуатації.

У дійсній моделі передбачається, що стани з порушенням безпеки є поглинаючими, тобто відновлення нормальної експлуатації АС при їх реалізації або неможливо, або недоцільно.

Інтенсивність переходу у стан “i0.” виражається формулою:

$$\lambda_{io}(t) = \lambda_i(t)\ \bar{Q}_i(t,t_p), \qquad (3.13)$$

де λ_i – інтенсивність (частота) і-ої ВП; $\bar{Q}`_i(t,t_p)$ – функція імовірності виконання всіх необхідних функцій безпеки при і-ої ВП.

Інтенсивність переходу в стан “i.1” з порушенням безпеки при і-ій ВП внаслідок невиконання однієї чи декількох функцій безпеки, що приводять до реалізації i_1 – ої аварійної послідовності, виражається формулою:

$$\lambda_{i\,1}(t) = \lambda_i(t)\ Q_i(t,t_p), \qquad (3.14)$$

де $Q_i(t,tp)$ – функція імовірності невиконання функцій безпеки для і-ої аварійної послідовності (АП) чи функція умовної ймовірності реалізації і-ої АП.

Імовірність реалізації і-го стану з порушенням безпеки визначаються по наступній формулі

$$P_{i1}(T) = \int_0^T \lambda_i(t)Q_i(t,t_p)P_0(t)dt, \qquad (3.15)$$

де:

$P_0(t)$– імовірність реалізації безпечного стану;

$Q_i(t,t_p)$ – визначається по деревам подій і деревам відмов відповідно до методик ІАБ;

T – розглянутий інтервал часу експлуатації ПНО (АЕС);

t_p – час приведення блоку в безпечний стан при виникненні **i-**ої ВП.

Оскільки важкі аварії є вкрай рідкими подіями, імовірність реалізації безпечного стану АС близька до одиниці. $P_0(t) \sim 1$ або $P_{i1}(T) << 1$, то формулу (3.15) можливо переписати у вигляді:

$$P_i(T) = \int_0^T \lambda_i(t) Q_i(t,t_p) dt. \qquad (3.16)$$

Формула (3.16) виражає консервативну (максимальну) оцінку $P_i(T)$ і рекомендується як основна залежність для оцінки ймовірностей кінцевих станів з порушенням безпеки.

Розглянемо приклад. Нехай дана деяка технічна система **T_1**, що складається з мінімального набору елементів: насосу **A,** і трьох засувок: **B, C, D,** розташованих у відповідності зі схемою рисунок 3.4. Критерієм успіху системи є подача води в точку **D_1** протягом 24 годин, небажану подію – відмову системи розглядаємо як подію, що є доповненням успіху.

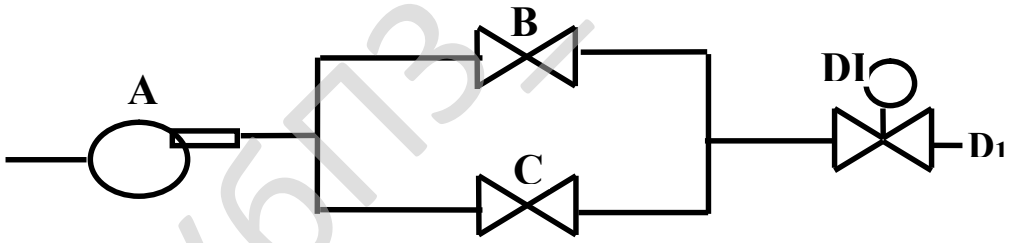


Рисунок 3.4 – Технічна система **T_1**

Тоді клас С моделі, що відповідає об'єкту моделювання, буде включати наступні елементи: відмову насоса на запуск – **x_1**, відмову насоса працювати 24 години – **x_2**, відмову засувки **B** на відкриття – **x_3,** відмову засувки **B** працювати 24 години – **x_4** , відмову засувки **C** на відкриття – **x_5,** відмову засувки **C** працювати 24 години – **x_6**, відмову засувки **D** на відкриття – **x_7**, відмову засувки **D** працювати 24 години – **x_8** і відсутність води в точці **D_1** – **x_9.** Крім того, необхідно розглянути подію **x_{10} ,** яка полягає в тому, що вода відсутня з деяких

причин на вході насосу, тобто в клас прообразу входить 9 елементів, що відображають технічні відмови. Очевидно, що у відношенні критерію успіху можуть бути записані рівності: $x_3 = x_5$ і $x_4 = x_6$, якщо засувки **B** і **C** однакові. Алгоритм побудови цієї моделі викладений у [7], відповідне ДВ представлене на рисунку 3.4, а відповідна булева функція буде:

$$x_9 = x_7 + x_8 + [(x_3 + x_4 + (x_1 + x_2 + x_{10}))^* (x_5 + x_6 + (x_1 + x_2 + x_{10}))],$$

або після спрощень відповідно до правил булевої алгебри (генерації мінімальних перерізів):

$$x_9 = x_1 + x_2 + x_7 + x_8 + x_{10} + x_4\ x_6 + x_4\ x_5 + x_3\ x_5 + x_3\ x_6 \qquad (3.17)$$

Отримана булева функція (3.17) дозволяє однозначно визначити події: x_1, x_2, x_7, x_8, x_{10} і сполучення подій: $x_4\ x_6$, $x_4\ x_5$, $x_3\ x_5$, $x_3\ x_6$, що приводять до відмови системи. Тобто у вигляді функції (3.17) отримана імовірнісна математична модель небажаної події – «відмова системи», що може досліджуватися відомими математичними методами.

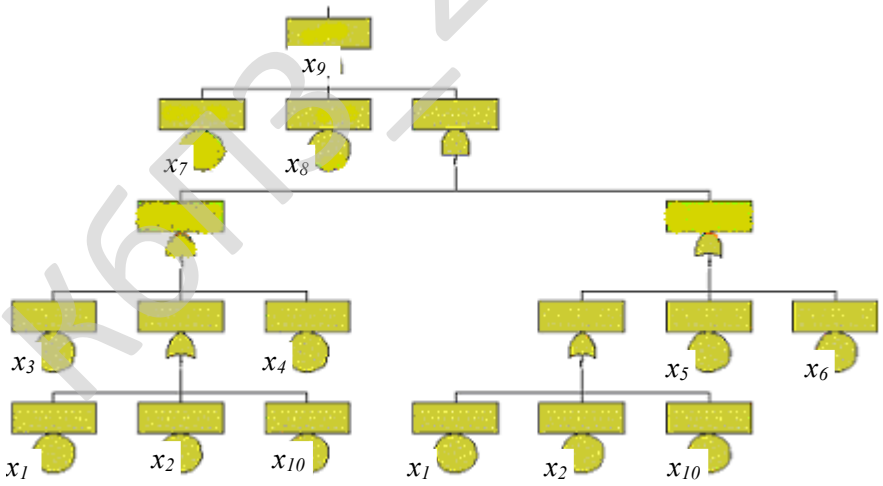


Рисунок 3.5 – Структурно-логічна модель відмови системи $\mathbf{T_1}$

3.2 Розробка структурної схеми

Виконання розрахунків в імовірнісних моделях (аналіз систем)

Виконання розрахунків в імовірнісних моделях, тобто аналіз систем, проводиться одночасно з визначенням мінімальних перерізів системи за допомогою розрахункового коду, зокрема, IRRAS [3, 4].

Імітаційні структурно-логічні моделі складних технічних систем відповідають марковській моделі випадкових процесів і складаються з дерев подій і дерев відмов [3, 5]. Дерева подій використовуються, якщо система розбивається на підсистеми, кожна з яких може виконати свою функцію і може мати різні ймовірності її виконання в залежності від різних факторів чи обставин. Структурно-логічні моделі у вигляді дерев відмов дозволяють відобразити будь-яку систему. Тобто, дерево відмов є графічною моделлю різних рівнобіжних і послідовних сполучень станів елементів системи (відмов), що приведуть до реалізації заздалегідь визначеної небажаної події. Для побудови дерева відмов необхідно провести аналіз можливого стану елементів і системи в цілому. Кожен елемент технічної системи може мати два стани: робоче і неробоче – відмову. Головна задача імовірнісного аналізу системи складається в розрахунку імовірності відмови системи на підставі даних про ймовірності відмов елементів. Розрізняють кілька типів відмов елементів в залежності від режиму роботи системи:

– Системи, що знаходяться у режимі очікування.

– Системи у режимі роботи

– Системи, що знаходяться у режимі очікування після їхнього запуску.

З досвіду відомо, що імовірності відмов елементів технічної системи залежать також від умов виникнення відмови, тому усі відмови технічних систем повинні бути класифіковані за наступними ознаками:

Функціональний прояв відмови (вид відмови):

– відмови на запуск;

– відмови в роботі;

– ушкодження, що може перейти у відмови.

Тип відмови:

– одиничний;

– множинний;

– відмови з загальної причини.

Тривалість усунення відмови:

– до 8 годин:

– понад 8 годин.

Режим роботи установки під час відмови:

– стаціонарний рівень потужності:

– зміна потужності:

– установка заглушена (для АЕС – реактор підкритичний).

Відмови в моделях ІАБ представляються як базисні події.

Базисна подія це таке ушкодження (дефект) як відмова устаткування, людська помилка, чи несприятлива умова для роботи елемента системи. Подія відмови не вимагає подальшої розробки, не може бути більше деталізована чи уточнена. Так, в попередньому прикладі на рисунку 3.4 базисні події – це події: $x_1, x_2, \ldots, x_8, x_{10}$.

Розглянемо об'єкт **О,** що складається з **n** систем $\mathbf{S_n}$**.** Для АЕС значення **n** може бути в межах кількох десятків, n = 1, 2, 3, … Кожна із систем $\mathbf{S_n}$ складається з **m** елементів $\mathbf{L_m}$**,** що мають **i** станів відмов з ймовірностями $\mathbf{P_i\ (L_m)}$**.** Звичайно i = 1, часто i = 2, іноді i = 3, але можливі й інші варіанти. Наприклад, для ємностей (баків) розглядають один тип відмови для всіх режимів роботи – течі, для засувки – два: відмови на відкриття (чи закриття) і відмови з загальної причини, для насосів три: відмови на запуск, відмови з загальної причини й відмови працювати заданий час $\mathbf{T_м}$**.** Імовірності відмов елементів у залежності від режиму роботи елементів системи обчислюються інтегруванням у межах часу $T_м$. У припущенні нормального закону їхнього розподілу, при

виконанні розрахунків за допомогою коду IRRAS, розрахунок відбувається за формулами таблиці 3.2[3].

Де інтенсивність відмов (λ) – умовна щільність імовірності виникнення відмови елемента – величина постійна, визначається дослідженням рядів статистичних даних відмов, як і закон розподілу щільності імовірності. Таким чином імовірний стан відмови кожного з елементів системи можна описати у вигляді логічної функції диз'юнкції:

$$P(L_m) = P_1(L_m) \,\&\, P_2(L_m) \& \ldots \& P_i(L_m) \ldots \quad (3.23)$$

Вплив відмов елементів системи на її відмову досліджується при аналізі роботи системи, залежить від конструкції системи, її схеми або від помилки оператора при експлуатації чи обслуговуванні системи. При цьому можливі наступні варіанти:

– відмова системи, при відмові одного елемента, тобто ймовірний стан системи можна записати як:

$$P(S_n) = P(L_m) \quad (3.24)$$

Для систем АЕС – це рідка подія, яку можна розглядати як недолік проектування (не виконується принцип одиничної відмови, згідно якого система повинна виконувати свої функції при відмові будь-якого елемента).

– відмови системи при відмові декількох елементів однієї групи:

$$P(S_n) = maj(P(L_1), P(L_2), \ldots, P(L_к)), \quad (3.25)$$

Що є розрахунковою подією для елементів, що знаходяться в резерві. Тут і далі «**maj**» – позначення для мажорантної логічної функції.

– відмови системи при послідовних відмовах декількох елементів у вигляді булєвої функції **B** від базисних подій :

$$P(S_n) = B(P(L_1), \ldots, P(L_m), P_o) \quad (3.26)$$

де P_o – базисна подія, пов'язана з помилкою оператора.

Така імовірнісна схема появи відмов, як показує досвід, є найбільш розповсюдженою схемою відмови технічної системи (відмови накопичуються до якоїсь межі).

Таблиця 3.2 – Типи розрахунків базисних компонентів у IRRAS

Тип розрахунку	Формула	Методика розрахунку
1	Імовірність (точкові оцінки)	Прямо задається імовірність базисної події чи частота вихідної події.
3	$1 - e^{-\lambda T_M}$	Розрахунок імовірності відмови компонент, що не ремонтуються, по рівнянню: $P = 1 - e^{-\lambda T_M}$
5	$\frac{\lambda \tau_R (1 - e^{-(\lambda + 1/\tau_R) T_M})}{1 + \lambda \tau_R}$	Розрахунок імовірності відмови компонент, що ремонтуються, де λ – інтенсивність відмов; τ_R – середній час ремонту (відновлення)
7	$1 + \frac{e^{-\lambda \tau_S} - 1}{\lambda \tau_S}$	Рівняння для розрахунку імовірності відмови резервних компонентів, що знаходяться у режимі очікування і періодично перевіряються. λ – інтенсивність відмов (резервна) і τ_s – час очікування, що дорівнює інтервалу між перевірками.

Іншими словами, відмови системи в будь-якому випадку можна представити у вигляді булєвої функції відмов її елементів і помилок оператора P_o. К прикладу, розглянемо складну технічну систему [67,68,106], яка складається з елементів: $\mathbf{w_{ij}}$, $\mathbf{e_j}$, $\mathbf{p_{ij}}$, $\mathbf{c_{ij}}$, $\mathbf{m_j}$, $\mathbf{r_i}$, $\mathbf{w_j}$, $\mathbf{b_j}$, $\mathbf{n_j}$. Підставляючи у вираз (3.26) значення логічних змінних, відповідно до їх ДВ через зазначені елементи системи (базисні події) і, спрощуючи отриманий вираз відповідно до операцій над логічними функціями, одержимо булєву функцію типу:

$$P(Sок) = B(wij, ej, pij, cij, mj, ri, wj, bj, nj), \quad (3.27)$$

у вигляді суми мінімальних перерізів, що представляє, із заданим ступенем точності, імовірність відмови системи:

$$P(S_{ок}) = w1j + w2j + r4 + r1 + \alpha\, wij + \alpha1\, p1j + \alpha2\, p2j + \alpha3\, nj + \alpha4\, p3j + e1,e2 + e1,e3 + e2,e3 + r2,r3 + w31,w33,p23 + w31,p22,p23 + w31,p21,p23 + p11,p12,p13 + p21,p22,p23 + \ldots \quad (3.28)$$

Мінімальний переріз визначається логічним добутком k базисних подій, що обумовлюють відмову системи (властивість перерізу). При цьому добуток **(k-1)** подій з цього набору **k** подій не повинний приводити до відмови системи (властивість мінімальності). Іншими словами, мінімальним перерізом називаємо сукупність первинних подій у системі, що мають дві властивості:

– Їх спільна реалізація призводить до відмови системи.

– Настання будь-якої комбінації меншого числа подій не призводить до відмови системи.

Тобто мінімальним перерізом є кожен з доданків виразу (3.28). Набір мінімальних перерізів системи однозначно визначений її деревом відмов і може бути отриманий вручну або за допомогою ЕОМ при використанні спеціальних алгоритмів вибору мінімальних перерізів [3].

Розглянемо довільне дерево відмов, що складається з комбінацій деяких базисних подій: **X_1, X_2, X_3, X_4,** зв'язаних логічними операторами: І (AND) – логічний добуток – (*) і АБО (OR) – логічна сума (+). Нехай ДВ таке, що вираз для верхньої події буде:

$$f(x_1, x_2, x_3, x_4) = (x_1+x_2)*(x_1+x_3) + (x_1+x_2)*x_4 + x_1*x_3;$$

Цей вираз можна спростити за допомогою наступних правил алгебри логіки:

1. $Z * (X+Y) = X*Z + Y*Z$;
2. $X+X=X$;
3. $X * X = X$;
4. $1+X =1$;
5. $1* X = X$.

Отже,

$$f(x_1, x_2, x_3, x_4) = (x_1+x_2)*(x_1+x_3) + (x_1+x_2)*x_4 + x_1*x_3 = x_1*x_1 + x_1*x_2 + x_1*x_3 + x_2*x_3 + x_1*x_4 + x_2*x_4 + x_1*x_3 = x_1 + x_1*x_2 + x_1*x_3 + x_1*x_4 + x_2*x_3 + x_2*x_4 = x_1 * (1 + x_2 + x_3 + x_4) + x_2*x_3 + x_2*x_4 = x_1 + x_2*x_3 + x_2*x_4.$$

Тобто, $\mathbf{f(x_1, x_2, x_3, x_4) = x_1 + x_2*x_3 + x_2*x_4.}$

Іншими словами, верхня подія відбудеться, якщо відбудеться: подія $\mathbf{x_1}$, або події $\mathbf{x_2}$ і $\mathbf{x_3}$,або $\mathbf{x_2}$ і $\mathbf{x_4}$, тобто верхня подія залежить від 3-х мінімальних перерізів.

Взагалі, за імовірність відмови системи приймаємо мінімальну апроксимацію верхньої границі мінімальних перерізів.

Мінімальна апроксимація верхньої границі мінімальних перерізів – це обчислення апроксимує ймовірність об'єднання мінімальних перерізів для дерев відмов. Рівняння для мінімальної апроксимації верхньої границі мінімальних перерізів:

$$S = \prod_{i=1}^{m} (1 - C_i), \qquad (3.29)$$

де

S – мінімальна верхня границя мінімальних перерізів для неготовності системи;

C_i – імовірність і-го мінімального перерізу;

m – число мінімальних перерізів.

Оскільки C_i – імовірності мінімальних перерізів, є малі величини, то з точністю до другого порядку малості, можливо на основі (3.29) обчислювати ймовірність відмови системи як суму мінімальних перерізів;

Мінімальні перерізи є ключовими інструментами для кількісного аналізу моделей ІАБ. Однак мінімальні перерізі також надають якісну, упорядковану інформацію, що доцільно використовувати для виявлення важливих відмов елементів, а також ситуацій, що можуть приводити до небажаних наслідків. Наприклад, група мінімальних перерізів, що складаються з одного елементу

системи, описує відмову окремих елементів, результатом яких є відмова всієї системи, випадок, що відповідає рівнянню (3.24).

Для моделювання імовірних небезпек виробництва потрібно також побудова декількох ДВ які можливо зв’язати за допомогою ДП в єдину модель виробництва. Тобто, для цілей побудови моделей небезпечних виробництв можливе використання не тільки ДВ але й ДП у випадках коли можливий опис виникнення НВ як послідовних відмов низки систем при їх незалежній роботі.

Дослідження значимості базисних подій, що входять в модель системи

Аналіз значимості й чутливості – Importance and Sensitivity Analysis – у ІАБ проводиться для формування практичних висновків по показниках ризику небажаних подій.

Аналіз значимості складається у визначенні значення внеску складових у частоту небажаної події, у частоти аварійних послідовностей і в показники ризику систем.

Кількісні показники значимості в імовірнісних моделях

Ціль оцінки значимості полягає у виявленні аварійних послідовностей, відмов систем та елементів, помилок персоналу важливих з погляду небажаної події. Для довільних систем, моделі яких складаються з одного дерева відмов, небажана подія – верхня подія ДВ. На основі імовірнісної структурно-логічної моделі можна визначити якісну значимість і кількісну значимість.

Якісна значимість пов'язана з логічною структурою моделей систем. Логічна структура моделі системи включає дерева відмов і дерева подій, а також сполучення відмов, що приводять до небажаних подій (мінімальні перерізи). Мінімальні перерізи представляють упорядковану інформацію, що використовується для виявлення важливих відмов елементів, а також ситуацій, що можуть приводити до небажаних наслідків. За допомогою мінімальних перерізів можуть бути виявлені відмови з загальної причини (для технічних систем).

Якісна інформація імовірнісних моделей забезпечує цінні критерії, що дозволяють оцінити значимість складових ризику і його змін. Кількісні показники значимості характеризують складові ймовірності небажаної події і її змін, що одержують на основі кількісних результатів ІАБ. Тут використовують отримані значення ймовірностей відмов систем, частот аварійних послідовностей або частот ушкодження. Хоча кількісні показники значимості можуть дати більш детальну інформацію, ніж якісні, вони, у той же час знаходяться під більшим впливом від невизначеностей, пов'язаних з розрахунками.

Загальна ідея визначення значимості базисних подій для системи полягає у визначенні оцінки зміни імовірності верхньої (небажаної) події за умови, що базисна подія яка нас цікавить ніколи не відбувається **P = 0,** чи відбувається з імовірністю **P = 1 –** достовірна подія.

Існують два основних показники значимості, широко використовувані у ІАБ:

– значимість по Бірнбауму (Birnbaum)

– значимість по Фусселлу-Веселі (Fussell-Vesely).

Значимість по Бірнбауму

Показник значимості Бірнбаума події X визначається як відношення зміни частоти небажаної події (ушкодження активної зони для АЕС) до зміни імовірності реалізації події X, тобто це є похідна:

$$B(x) = \frac{d}{dx} P(x), \text{ або } \quad B(x) = F(1) - F(0), \qquad (3.30)$$

де F(1) – це значення небажаної події (CDF – core damage frequency – частота ушкодження активної зони) у припущенні, що подія реалізувалася, а F(0) – у припущенні, що подія не відбулася. Таким чином, міра значимості Бірнбаума є різницею між частотою небажаної події, обчисленої в припущенні реалізації події X, і частотою небажаної події, обчисленої в припущенні, що подія X не реалізувалася.

В IRRAS показник значимості Бірнбаума події X визначається також через коефіцієнти зміни ризику й інтервали зміни ризику:

– RRR – коефіцієнт зменшення ризику (Risk Reduction Ratio)

– RRI – інтервал зменшення ризику (Risk Reduction Interval).

Коефіцієнт зменшення ризику показує наскільки зменшиться верхня границя мінімальних перерізів, якщо імовірність основної події зменшиться до нуля.

$$RRR = F(X)/ F(0) \quad (3.31)$$

$$RRI = F(X) – F(0)$$

Коефіцієнт збільшення ризику й інтервал збільшення ризику:

– RIR – коефіцієнт збільшення ризику (Risk Increase Ratio).

– RII – інтервал збільшення ризику (Risk Increase Interval).

Коефіцієнт збільшення ризику показує як збільшиться мінімальна верхня границя мінімальних перерізів, якщо ймовірність основної події збільшиться до одиниці.

$$RIR = F(1)/ F(X) \quad (3.32)$$

$$RII = F(1) – F(X).$$

Бірнбаум дорівнює сумі інтервалів зміни ризику

$$B = RII + RRI;$$

Значення інтервалу зменшення ризику дорівнює добутку Бірнбаума і номінальної імовірності базисної події

$$RRI = B*X$$

Значення інтервалу збільшення ризику дорівнює добутку Бірнбаума і доповнення імовірності базисної події

$$RRI = B*(1-X).$$

Значимість по Фусселлу-Веселі

Показник значимості Фусселлу-Веселі події X визначається як відносний внесок події в частоту небажаної події – ушкодження активної зони (чи в частоту аварійної послідовності). Значимість по Фусселлу-Веселі може бути виражена як сума внесків мінімальних перерізів, що містять подію X,

$$FV = [F(X) - F(0)] / F(X), \quad (3.33)$$

де F(X) – це значення небажаної події (CDF – core damage frequency – частота ушкодження активної зони) при номінальній імовірності базисної події, а F(0) у припущенні, що подія не відбулася. Показник значимості Фусселлу-Веселі має зв'язок із попередньо визначеними показниками значимості:

$$FV = (B*X) / R(X)$$

Аналіз чутливості

Аналіз чутливості полягає у визначенні чутливості результатів ІАБ до вихідних допущень, моделей та даних. При проведенні аналізу як значимості, так і чутливості, особливо для первинних (базисних) подій, варто враховувати їхній взаємозв'язок. Неминуче події, що мають велику розрахункову значимість, будуть також демонструвати високу чутливість.

Розрахунки аналізу чутливості проводяться після завершення задачі аналізу значимості.

Аналіз чутливості проводиться стосовно параметрів факторів, що, по – перше, були визначені в процесі аналізу значимості як домінантні і, по-друге, мали великий ступінь невизначеностей, тобто мали великий розкид.

Як уже відзначалося, ціль аналізу чутливості двояка і полягає в наступному:

– Визначити наскільки частота небажаної події чуттєва до можливих залежностей між відмовами елементів (базисними подіями) і між помилками персоналу;

– Виявити ті допущення моделювання, що суттєво можуть впливати на результати. Такі допущення мають місце, головним чином, в областях з недостачею інформації в які необхідно покладатися на експертні висновки. У цьому випадку аналіз чутливості можна виконати заміняючи допущення на альтернативні й оцінюючи їхній індивідуальний вплив на результати.

Аналіз чутливості проводиться шляхом варіації параметрів у межах діапазону від мінімальних до максимальних значень, що відповідають 10% і 90% квантилям їхніх розподілів. Як такі параметри можуть розглядатися наступні:

– інтенсивності (імовірності) незалежних відмов або подій;

– показники помилкових дій персоналу;

– тимчасові характеристики технічного обслуговування й ремонтів.

За результатами аналізу чутливості коректується аналіз значимості, якщо це буде визнано необхідним.

Аналіз чутливості може проводитися на основі систем або на основі аварійних послідовностей. Для практичного виконання перерахованих досліджень чутливості необхідно знати правила внесення змін у імовірнісні моделі в розрахунковому коді (IRRAS), з цієї причини подальший виклад матеріалу носить чисто практичний характер, описано в [7]. Приводимо загальний огляд дій виконання аналізу чутливості дерева відмов, які полягають в наступному:

– Якщо повинні бути зроблені зміни логіки дерева відмов (наприклад, при додаванні базисної події, при пересуванні базисної події або при заміні OR-gate на AND-gate) зміни проводяться використовуючи графічний і логічний редактор дерева відмов.

– Якщо зміни здійснюються в даних, зробити це можна одним із двох способів: зміни даних “постійно” в опції вводу даних головного меню або зміни даних “тимчасово” – використовуються опції змін набору.

Методи врахування людського чинника в моделі

Людський чинник розуміємо як ризик, що пов'язаний з помилками людини-оператора об’єкту підвищеної небезпеки. Людина – оператор має спеціальну підготовку, на підприємстві діє система управління охороною праці (СУОП) яка передбачає комплекс заходів щодо підтримки кваліфікації робітників.

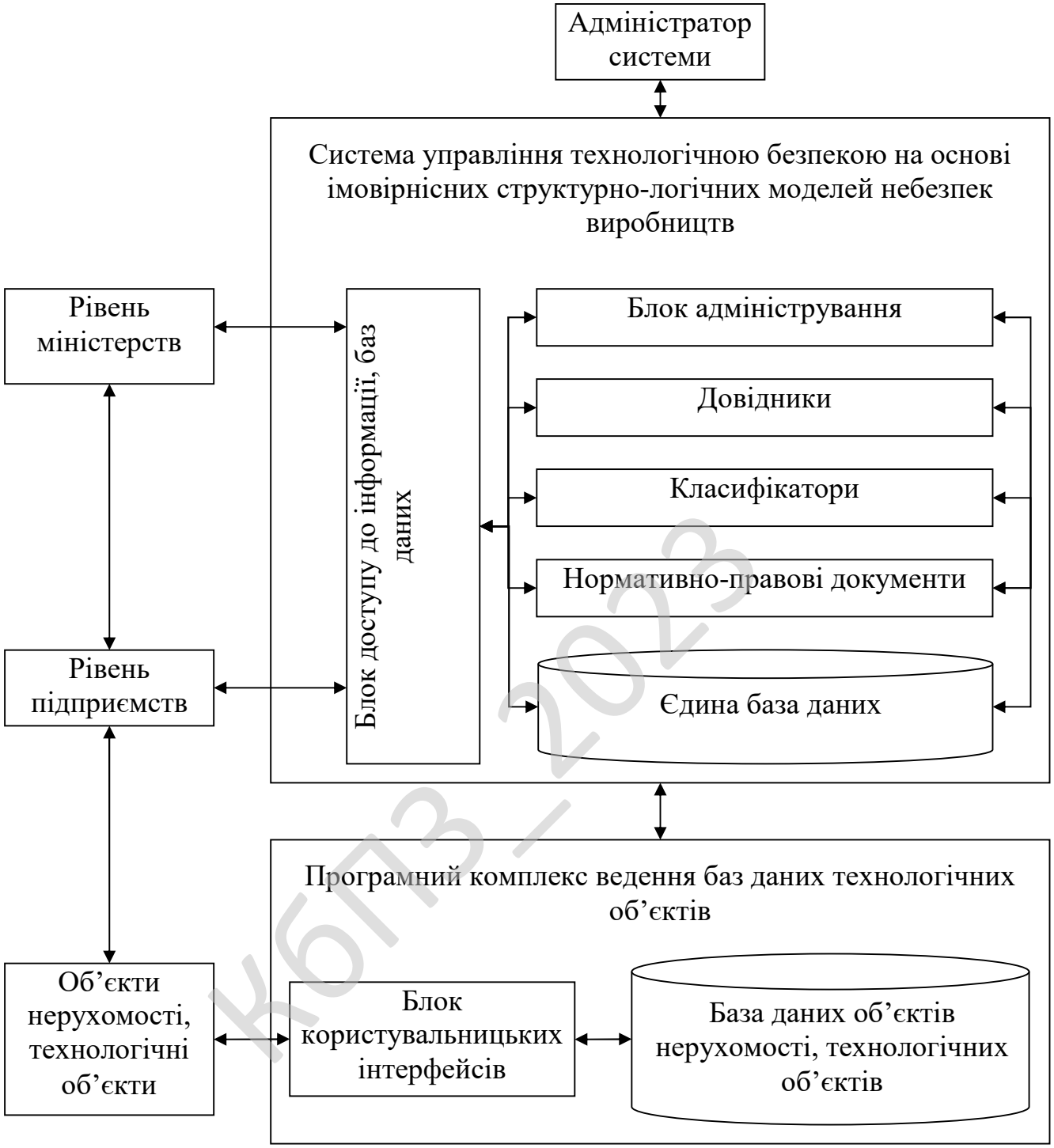


Рисунок 3.9 – Структурна схема системи

На рисунку 3.9 зображена структурна схема системи.

3.3 Розробка функціональної схеми

Для управління системою оператору необхідна інформація, яка б дозволяла:

– швидко оцінити загальний стан об’єкту, тобто в стані нормальної експлуатації, в умовах очікуваної експлуатаційної події чи в аварійному стані і переконатися, що виконуються запроектовані автоматичні дії по забезпеченню безпеки;

– визначити відповідні дії, які необхідно розпочати оператору.

Для виконання ролі оператора устаткування людині потрібна інформація з параметрів окремих систем об’єкту й устаткування.

Необхідно, щоб проект сприяв успішному виконанню оператором своїх дій у межах наявного часу, в умовах передбачуваного навколишнього фізичного середовища і психологічного навантаження. Бажано звести до мінімуму необхідність у негайному втручанні оператора. У проекті варто врахувати той факт, що таке втручання прийнятне тільки в тому випадку, коли проектувальник може довести, що оператор має досить часу для прийняття рішення і відповідних заходів; що необхідна інформація, на основі якої оператор повинен приймати своє рішення, представлена в дохідливій і чіткій формі і що фізичні параметри навколишнього середовища в приміщенні пульту управління та на додаткових пультах управління об’єктом після даної події є прийнятними.

Процедура системного аналізу помилок

Процедура системного аналізу помилок людини має загальні кроки для різних методик [4].

Ця процедура відома в англійській абревіатурі як SHARP – Systematic Human Action Reliability Procedure. Процедура включає сім кроків і два етапи, на яких приймаються рішення. Два перших кроки виконуються системними аналітиками, два подальших – фахівцями з аналізу людського чинника, останні три кроки процедури виконуються спільними зусиллями.

Розглянемо процедуру виконання кожного кроку.

– Крок 1 – Визначення **дій** людини, включаючи дії по ремонту, роботу по програмах, дії по локалізації аварій, тобто всіх дій з помилками персоналу, які погіршують або поліпшують ситуацію.

– Крок 2 – **Скрінінг – відбір** важливих подій, помилок, що мають ключове значення для імовірності аварійної ситуації (наприклад, плавлення активної зони).

– Крок 3 – Розділення – **виділяються** всі дії оператора, що вимагають більш детального аналізу, тобто задача розбивається на більш дрібні, характерними складовими яких будуть:

1. Здатність зрозуміти, що треба робити (для аналітиків), розділення дій по операціях;

– Нездатність визначити систему.

– Нездатність виконати дії.

2. Включення кроків в модель ІАБ;

3. Необхідність (можливість) побудови додаткових аварійних послідовностей, якщо визначилися інші помилки операторів.

– Крок 4 – Уявлення – повне **представлення** всіх помилок і їх аналіз з докладними діями.

– Крок 5 – Визначення **взаємного впливу** елементарних дій (операцій), впливи на наступні етапи;

– Крок 6 – Розрахунки – визначаються **кількісні** значення ймовірностей помилок;

– Крок 7 – Документування.

Обсяг робіт на кожному кроці залежить від типу методики, що використовується.

Визначення базових значень ймовірностей помилок людини

Помилки під час експлуатації частіше складають помилкові дії: або не передбачені, або передбачені в експлуатаційних процедурах, чи процедурах

технічного обслуговування; рідше – невиконання окремих дій, що вимагаються. Прикладами є неправильний вибір засобів управління, передача неправильних команд чи інформації, зміна послідовності виконання задач і занадто раннє чи занадто пізнє виконання задач. Такі помилки можуть виникнути в результаті помилок при прийнятті рішень операторами: невірно витлумачених чи нечітких процедур; помилкові показання контрольно-вимірювальних приладів; неправильне розуміння чи просто помилка оператора. Для визначення надійності системи ці можливі помилки потрібно враховувати в проекті.

Значення імовірності помилки у випадку наявності адекватних процедур та їх застосування при виконанні складних робіт зменшується у 50 разів.

Представляє інтерес також таблиця обліку залежності ймовірностей помилки від змін обставин: змін в складі бригади, що виконує роботу, розділення дій за часом і місцем, наявність чи відсутність вказівок на виконання роботи (підказки) і комбінацій названих обставин. Врахування всіх обставин проводиться по нижченаведеному алгоритму. Значення імовірності помилки без врахування змін обставин (НЕР) дорівнює N, в залежності від їх комбінацій – номери для типів помилкових дій людини, приймає значення:

– 1 (одиниця) у разі повних змін;

– (1+N)/2, у разі високих (великих) змін;

– (1+6N)/7, у разі помірних змін;

– (1+19N)/20, у разі низьких (малих) змін;

– N, у разі нульових змін.

Ця таблиця відноситься до методики ASP, що широко використовується для аналізу помилок людини-оператора. При цьому, обставини сильно змінюють імовірність помилки. Наприклад, якщо базова ймовірність N = 0.1, то в залежності від змінених обставин імовірність помилки буде: 0,1; 0,145; 0,23; 0,55; і 1,0 відповідно до різного ступеню залежності.

Побудова дерева помилок персоналу

Дерева аналізу надійності людини вперше були введені в методиці THERP[46]. Приклад дерева аналізу надійності людини приведено на рисунку 3.10. Події в такому дереві позначаються у вигляді відрізків прямих ліній, розташованих під кутом один до одного.

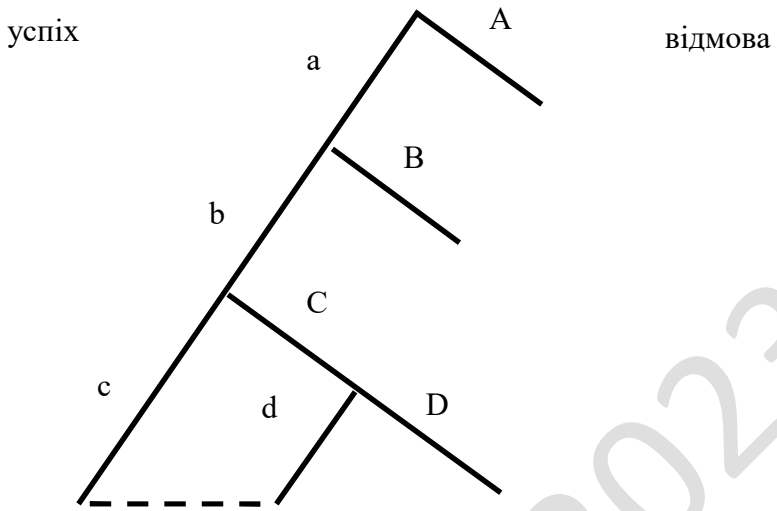


Рисунок 3.10 – Приклад дерева аналізу надійності людини

Відрізки, розташовані **праворуч,** зображають **неуспішні** дії (відмови), які позначаються великими літерами латинського алфавіту, відрізки зліва **– успішні дії**, позначаються малими буквами латинського алфавіту. Поруч з позначенням, звичайно можуть бути пояснюючі надписи й значення ймовірності відповідної події. Дії по відновленню функцій систем позначаються горизонтальними пунктирними лініями.

Відрізки показують послідовність дій. Початок послідовності дій знаходиться вгорі. Кожне дерево, в залежності від числа гілок, має різне число послідовностей. Послідовністю тут будемо називати дії, або їх сукупність, що приводять до якогось результату (на кінець гілки). Так дерево, представлене на рисунку 3.10 містить такі послідовності: abc**, A,** a**B,** ab**C****D,** ab**C**d**.**

Чисельні значення ймовірностей неуспіху проставляються поруч з подіями. Зазначимо, що успіх і відмова є взаємно доповнюючими подіями. Нехай імовірності будь-яких дій **A, B, C, D** рівні P(f) = 0.006. Імовірності послідовностей, таким чином, можуть бути легко розраховані (рисунок 3.11).

У приведеному прикладі ймовірність помилки досить мала. Діапазон зміни ймовірності помилки взагалі – від 0 до 1, звичайно, вона належить інтервалу (0,01;0,4). Імовірності більше 0,2 прийнято вважати високими.

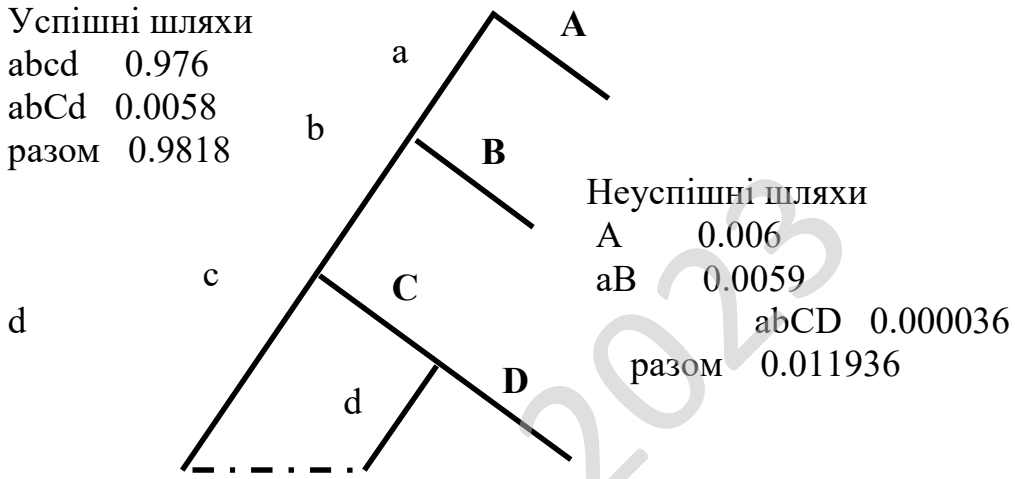


Рисунок 3.11 – Приклад розрахунку послідовностей дерева аналізу надійності людини

При побудові дерева надійності людини також як і при побудові дерева відмов систем, важливе значення мають всі допущення, припущення і межі. Всі вони повинні бути чітко описані.

Аналіз загального ризику з врахуванням помилок персоналу

При оцінці ризику від промислових об’єктів (АЕС) необхідно враховувати дії людини, тобто повинен використовуватися комплексний підхід: системний аналіз надійності обладнання, аналіз надійності людини і, в результаті, аналіз загального ризику:

Відмова системи = відмова обладнання + помилка людини.

Наведена формула дещо спрощена, оскільки є взаємодії, які можуть поліпшити або погіршити ситуацію. Розуміти її треба так: оператор остання лінія захисту (в глибоко ешелонованому захисті).

Цілі моделювання подій, пов'язаних з помилками людини:

– представити помилки і їх механізми;

– якісно представляти роль чинників, що впливають на поставлену задачу;

– кількісно описати помилку, що дозволить зробити аналіз чутливості;

– виробити стратегію запобігання помилкам або відновлення функцій оператором.

Основні базисні події, пов'язані з помилками операторів, можуть вводитися в моделі ІАБ на рівні систем – в дерева відмов, або на рівні послідовностей – в дерева подій. Введення в дерева відмов основних базисних подій, пов'язаних з помилками операторів, має деякі переваги:

– отримані при детальному моделюванні відомості враховуються спочатку і використовуються для подальшого аналізу;

– не використовуються консервативні оцінки;

– консервативний відбір для дерев відмов і оцінки базисних подій, пов'язаних з помилками операторів, призводить до того, що завищений рівень людських помилок суперечить повному аналізу. Це сприяє виконанню детального моделювання і кількісних оцінок.

Загальне дерево відмов може мати вигляд, як на рисунку 3.13.

Імовірність відмови системи в цьому випадку визначається з урахуванням можливої помилки оператора, що має всі кількісні характеристики і свій закон розподілу ймовірностей.

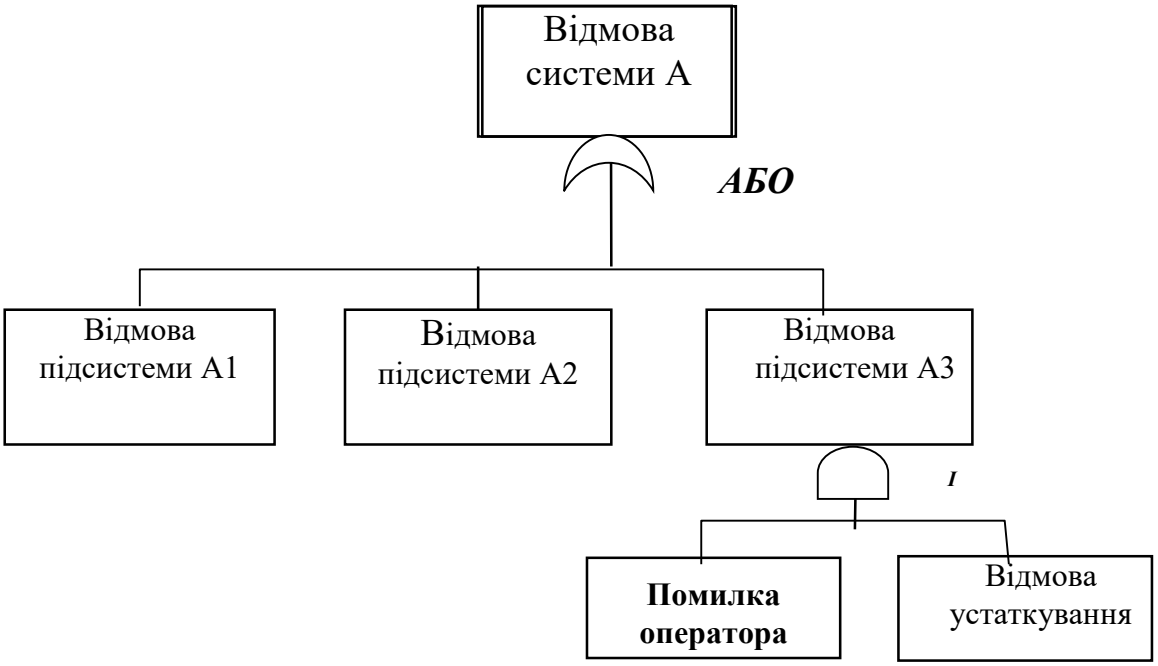


Рисунок 3.13 – Приклад дерева відмов, що враховує помилку оператора

У деревах подій врахування людського чинника (HF) може відбуватися як проміжні події із заданими ймовірностями відновлення систем, що відмовили. У випадку складної залежності проміжні події можуть вводитися окремими деревами аналізу надійності людини. Ці дерева можуть мати звичайну форму, або спеціальну, яка розглянута вище.

Отже, в моделях ІАБ дії (помилки) оператора можуть враховуватися в таких варіантах:

– дії зроблені неправильно;

– дії, що призводять до відмов;

– як успішні й аварійні послідовності;

– як дії по відновленню оператором функцій систем, що відмовили;

– при аналізі взаємозалежності і взаємного впливу окремих подій (THERP);

– як типи різних помилок.

В цілому використання методики THERP цілком виправдано для моделювання помилок оператора. Методика проста, зрозуміла й однозначна. Для

широкого її застосування необхідно навести зв'язок імовірності помилки оператора від його навченості, досвіду та стану діючої системи управління охороною праці на підприємствах різних галузей України, тобто адаптації методики до конкретних умов.

Застосування імовірнісних моделей для визначення ризику виробництва

Ризик виробництва розуміємо як змінну що характеризує стан безпеки виробництва і дорівнює добутку імовірності небажаної події на величину її наслідків.

Побудова структурно-логічних моделей процесів охорони праці

Якщо маємо випадковий процес дій людини в процесі виробництва, шляхом його дослідження, вивчення й аналізу можливо встановити залежності кінцевих подій від причин та базисних подій, що його складають, а звідсіля і розробка моделей будь-яких НВ на базі наведених вище алгоритмів, визначення (побудова) структурно-логічної моделі НВ як системи технологічного, природного або соціального характеру. Поняття "Система" розуміємо в загальному визначенні, як множину об'єктів разом із відносинами між об'єктами і їх атрибутами (визначення А. Хола). При цьому, на відміну від систем безпеки АЕС, не завжди існують матеріальні зв'язки між елементами в вигляді трубопроводів або електричних з'єднань. Частіше в системах, що пропонується розглядати, зв'язки між елементами існують тільки в логічному виді. В цьому полягає різниця між системами АЕС і системами що пропонується розглядати. Відношення системи до того чи іншого типу буде, очевидно, залежати від факторів та обставин, які пов'язані з джерелом небезпек, їх впливу на розвиток та наслідки небажаних подій. Ці фактори та обставини повинні бути проаналізовані фахівцями галузі, відібрані для складання й опису моделі ці з них, що складають замкнуту систему. Важливим етапом цієї роботи буде визначення базисних подій, при цьому обов'язкове дотримання всіх вище приведених вимог щодо базисної події. Фактори та обставини, які будуть прийматися за базисні події, повинні

розглядатися як деякі захисні бар'єри, що перешкоджають появі та розвитку небажаної події. Базисна подія у цьому випадку розглядається як відмова захисного бар'єру, людська помилка, чи несприятлива умова для функціонування визначених захисних бар'єрів системи. Подія відмови, як і для технічних систем, не вимагає подальшої розробки, не може бути більше деталізована чи уточнена.

Побудова імовірнісних моделей (ДВ) в цьому випадку, аналіз системи, отримання МП, розрахунок імовірності небажаної події – все це задачі ІАБ, їх вирішення можливо за допомогою коду IRRAS.

Використання ДВ в задачах розрахунку ризиків

На основі ДВ досить просто будувати моделі складних технічних систем. Якщо система, ризик якої визначається, сама складається з декількох систем, здатних впливати на наслідки, доцільно вводити моделі дерев подій, що зв'язують роботу систем єдиною логікою. Можливе застосування методу аналізу ДВ і не для технічних систем. Як базисні події в такому випадку мають бути узяті такі події, що впливають на ризик (наслідки) і не можуть бути більш спрощені, при цьому імовірності чи інші їхні імовірнісні характеристики можуть бути знайдені якими-небудь методами. При цьому алгоритм і правила побудови й розрахунку ДВ змінюються відповідно до наведеного нижче алгоритму.

Основні кроки аналізу ризику за допомогою структурно-логічних моделей для будь яких систем:

– Визначити і коротко характеризувати небажану подію, що можлива як підсумкові дії небезпечного або шкідливого чинника.

– Визначити і коротко охарактеризувати початкову подію, що обумовлює дію шкідливого чинника.

– Визначити базисні (незалежні) події, що можуть впливати на процес дії небезпечного чинника за час від початкової події до небажаної події.

– Зробити оцінку інтервалу часу від початкової події до небажаної події.

– Визначити чинники й обставини, що можуть впливати на хід подій за час від початкової події до небажаної події.

– Оцінити зв'язок базисних подій (чинників) і обставин за час від початкової події до небажаної події.

– Визначити можливі міри й засоби запобігання дії шкідливого або небезпечного чинника, і ввести їх у модель базисними подіями.

– Визначити можливе втручання людини в процес і можливі помилки при цьому, і врахувати їх у імовірнісної моделі.

– Зробити оцінку ймовірностей приведених базисних подій.

– Побудувати дерево відмов, визначити ступінь ризику під час дії небезпечного чинника, і можливі шляхи запобігання небажаної події.

При побудові дерева надійності людини також як і при побудові дерева відмов систем, важливе значення мають всі допущення, припущення і межі визначеної системи.

Управління ризиками. Загальна схема

Мету управління ризиком при здійсненні будь-якої діяльності можна визначити як забезпечення безпеки персоналу і навколишнього природного середовища, шляхом встановлення і підтримки прийнятного рівня ризику, при використанні оптимальним чином з максимальною ефективністю наявних матеріальних ресурсів.

Управління ризиками – це діяльність, пов'язана з ідентифікацією, аналізом ризиків і прийняттям рішень, спрямованих на мінімізацію негативних наслідків настання вихідних подій (явищ) і/чи зменшення імовірності їхньої реалізації до прийнятних значень. У загальному випадку процес управління ризиками при здійсненні діяльності на об'єкті включає виконання шести процедур (рисунок 3.13).

Планування управління ризиками – це процес прийняття рішень по застосуванню і плануванню управління ризиками для конкретної діяльності. Цей процес може містити в собі:

– Організацію на підприємстві спеціального підрозділу (групи управління ризиками), відповідального за оцінку і управління.

– Вибір методики оцінки ризиків.

– Визначення джерел даних для ідентифікації ризику.

– Визначення тимчасового інтервалу для аналізу ситуації.

Дуже важливим буде визначення припустимих (прийнятних) рівнів ризику, які повинні вибиратися на основі чинного законодавства.

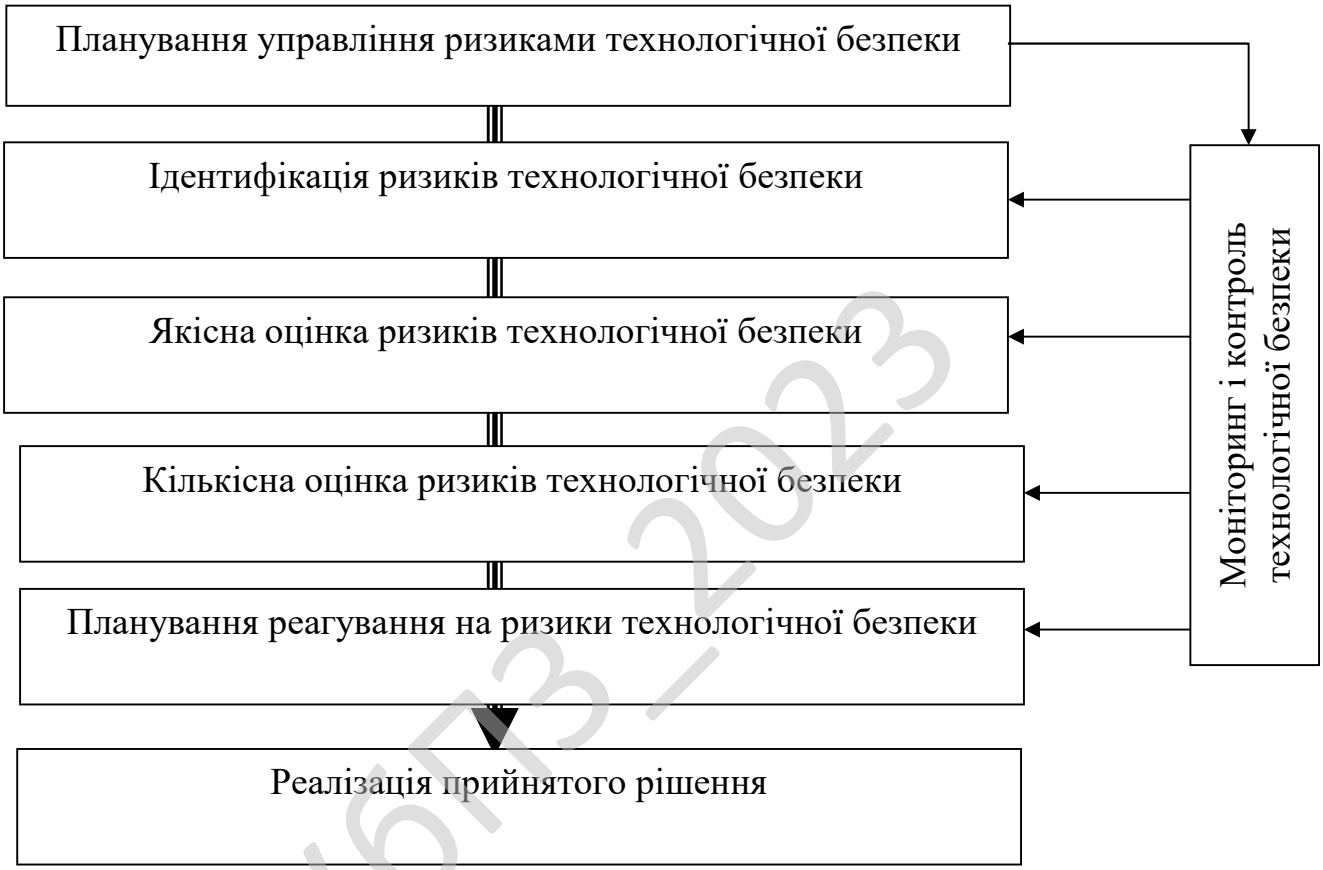


Рисунок 3.13 – Функціональна схема системи

Ідентифікація ризиків визначає, які ризики можуть вплинути на діяльність, що розглядається. Характеристики цих ризиків повинні бути оформлені документально. Ідентифікація ризиків повинна проводитися регулярно протягом усієї діяльності об'єкта. Спеціалізований підрозділ повинний залучати до робіт по ідентифікація ризиків всіх учасників процесу: проектантів,

експлуатаційників, фахівців інших підрозділів і незалежних експертів. Ідентифікація ризиків повинна організовуватися як ітераційний процес. Перші розрахунки потенційного ризику виконують проектанти. У процесі діяльності об'єкту, з урахуванням досвіду експлуатації, уточнюються дані по надійності систем і устаткування, процедурам управління, помилкам персоналу і робиться перерахунок ризиків для об'єкту. Для формування об'єктивної оцінки в завершальній стадії процесу оцінки можуть брати участь незалежні експерти. Приклад ідентифікації ризиків, для радіаційних ризиків викладений у галузевому нормативному документі НРБУ-97/Д-2000.

Якісна оцінка ризиків – це процес якісного аналізу результатів ідентифікації, а також визначення ризиків, що вносять найбільший внесок у загальний ризик і які потребують вживання заходів по їхньому зниженню.

Як уже було показано раніше, останній етап якісного аналізу систем полягає в представленні умов невиконання функцій системи у вигляді так званої множини мінімальних перерізів [37]. Набір мінімальних перерізів системи однозначно визначений її деревом відмов і може бути отриманий при використанні спеціальних алгоритмів вибору мінімальних перерізів. Кількісні дані по базисних подіях впливають на важливість самого мінімального перерізу – його відсотковий вклад в імовірність відмов системи.

Якісна оцінка визначає ступінь важливості ризику і складових його подій. Доцільно створити банк даних ризиків усієї діяльності на об'єкті, заснований на систематизованих даних, у тому числі даних по впливу ризиків на персонал. На цьому етапі можливо визначення чинників найбільшого впливу, що створить передумови управління.

Кількісна оцінка ризиків визначає значення імовірності виникнення ризиків і впливу їхніх наслідків на діяльність, що допомагає приймати оптимальні рішення й уникати невизначеності (у змісті управління) при цьому. Кількісна оцінка ризиків передбачає виконання попередніх процесів, це завершальний етап задачі визначення ризиків.

Планування реагування на ризики – це розробка методів і технологій зниження негативних наслідків ризиків. Якісне, науково обґрунтоване планування можливе за умови виконання всіх попередніх етапів процесу відповідно до рисунка 3.13. Стратегія планування повинна відповідати типам ризиків, їх величині і значимості, наявності ресурсів і тимчасових параметрів. У найбільш небезпечних випадках, можливо потрібно кілька варіантів реагування на ризики. Планування повинне здійснюватися у відповідності зі спеціальною методикою, що враховує специфіку об'єкту, чинні на ньому правила й інструкції.

Реалізація прийнятого рішення проводиться з врахуванням даних моніторингу і контролю параметрів, що проводяться з метою перевірки дотримання вимог встановлених норм. Моніторинг і контроль повинні здійснюватися спеціалізованим підрозділом об'єкту. При цьому повинні постійно контролюватися процес ідентифікації ризиків, виконання плану реагування на ризики, оцінка ефективності заходів для зниження ризиків, величина залишкового ризику і його прийнятність.

Якісний контроль виконання діяльності подає інформацію, що сприяє прийняттю ефективних рішень по запобіганню нових ризиків чи зм'якшення наслідків. Контроль може ініціювати вибір альтернативних стратегій, прийняття коректив, перепланування проекту для досягнення базового плану.

Для цілей моніторингу і перевірки дотримання норм забезпечується належне устаткування і впроваджуються відповідні процедури перевірки. Зазначене устаткування належним чином обслуговується і випробовується, а також калібрується з належною періодичністю на основі еталонів, що відповідають національним чи міжнародним еталонам.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.14. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

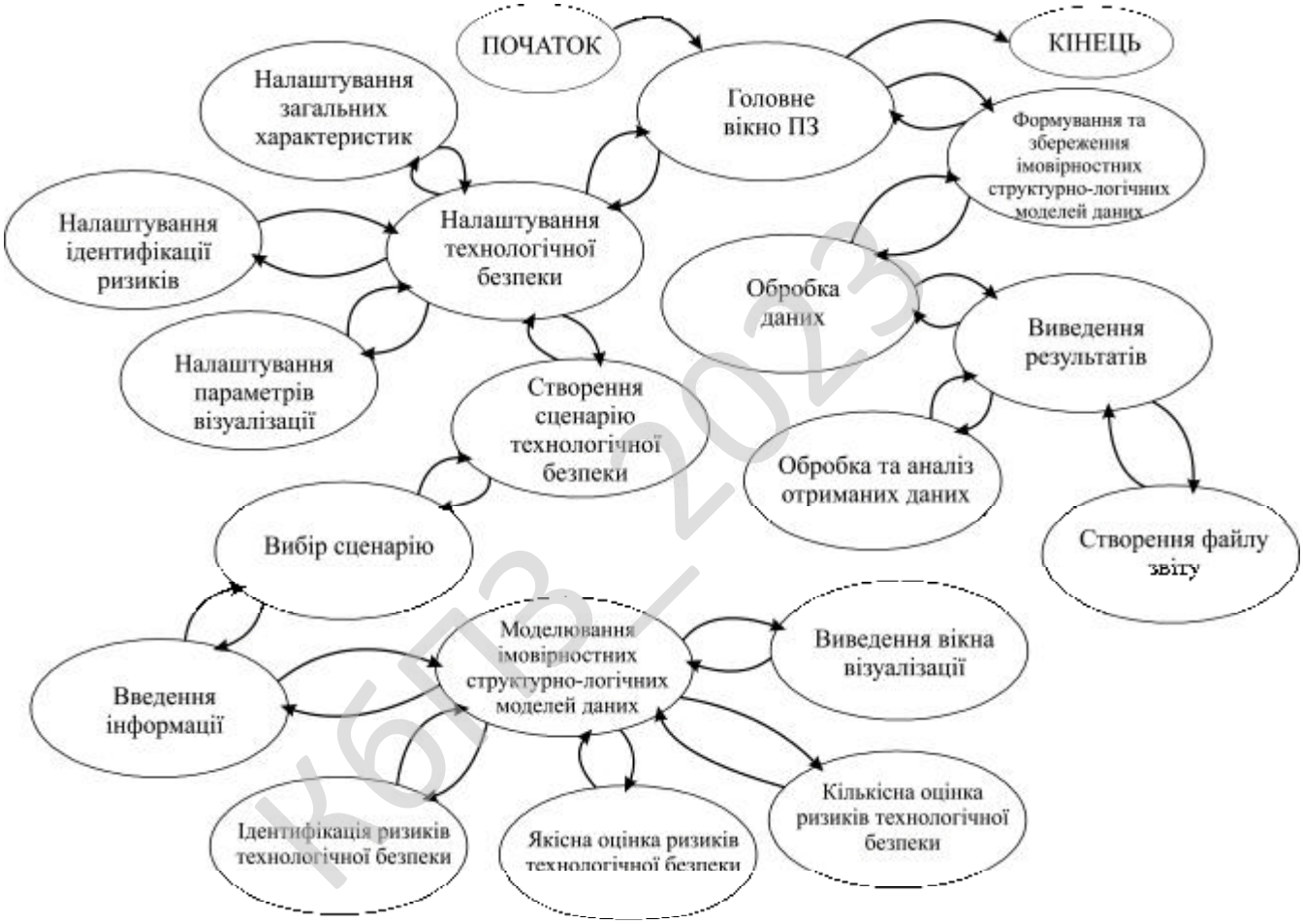


Рисунок 3.14 – Діаграма взаємодії процесів

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

уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

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

Опис алгоритмів функціонування системи.

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою.

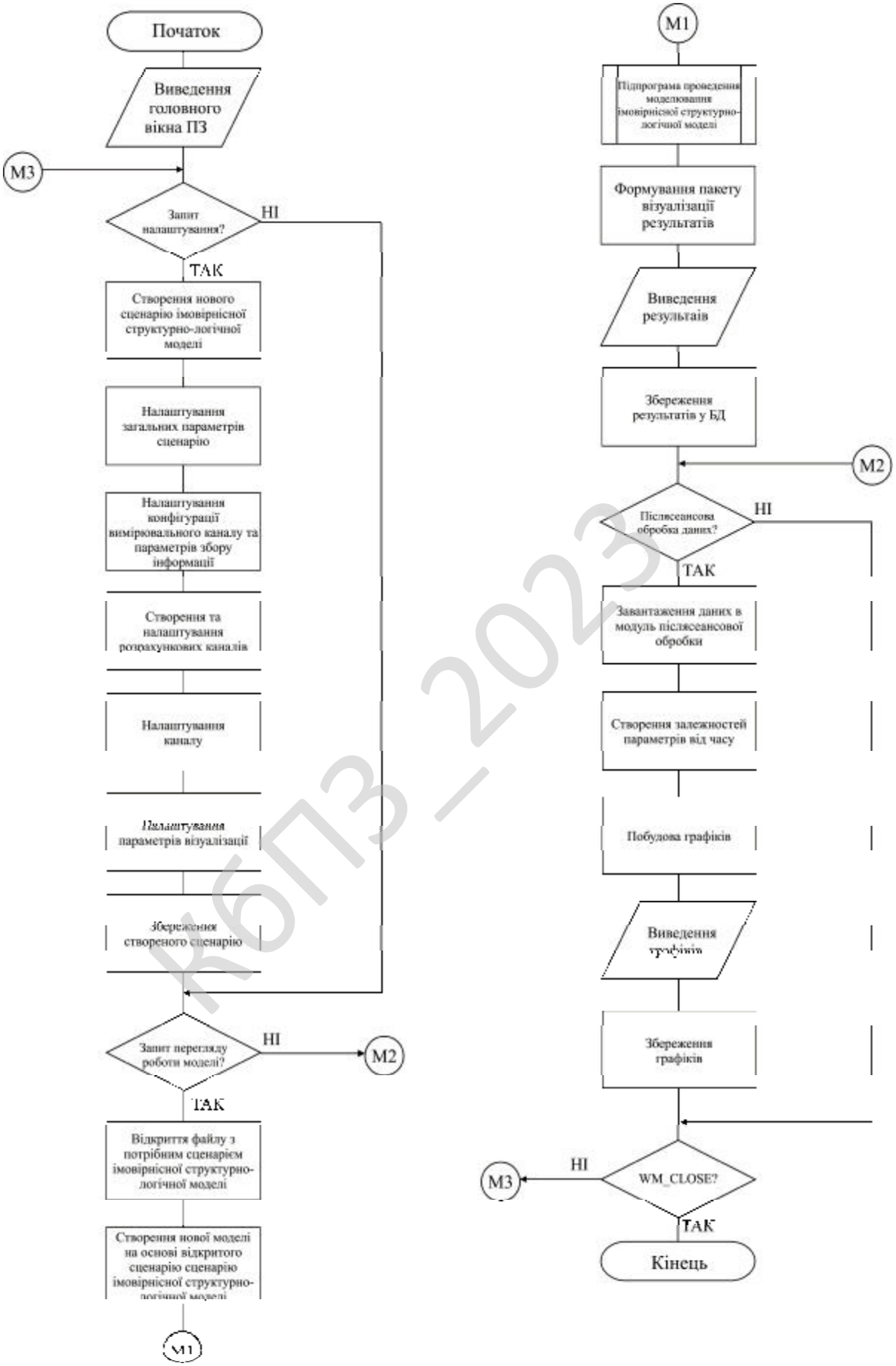


Рисунок 4.1 – Блок-схема основної програми

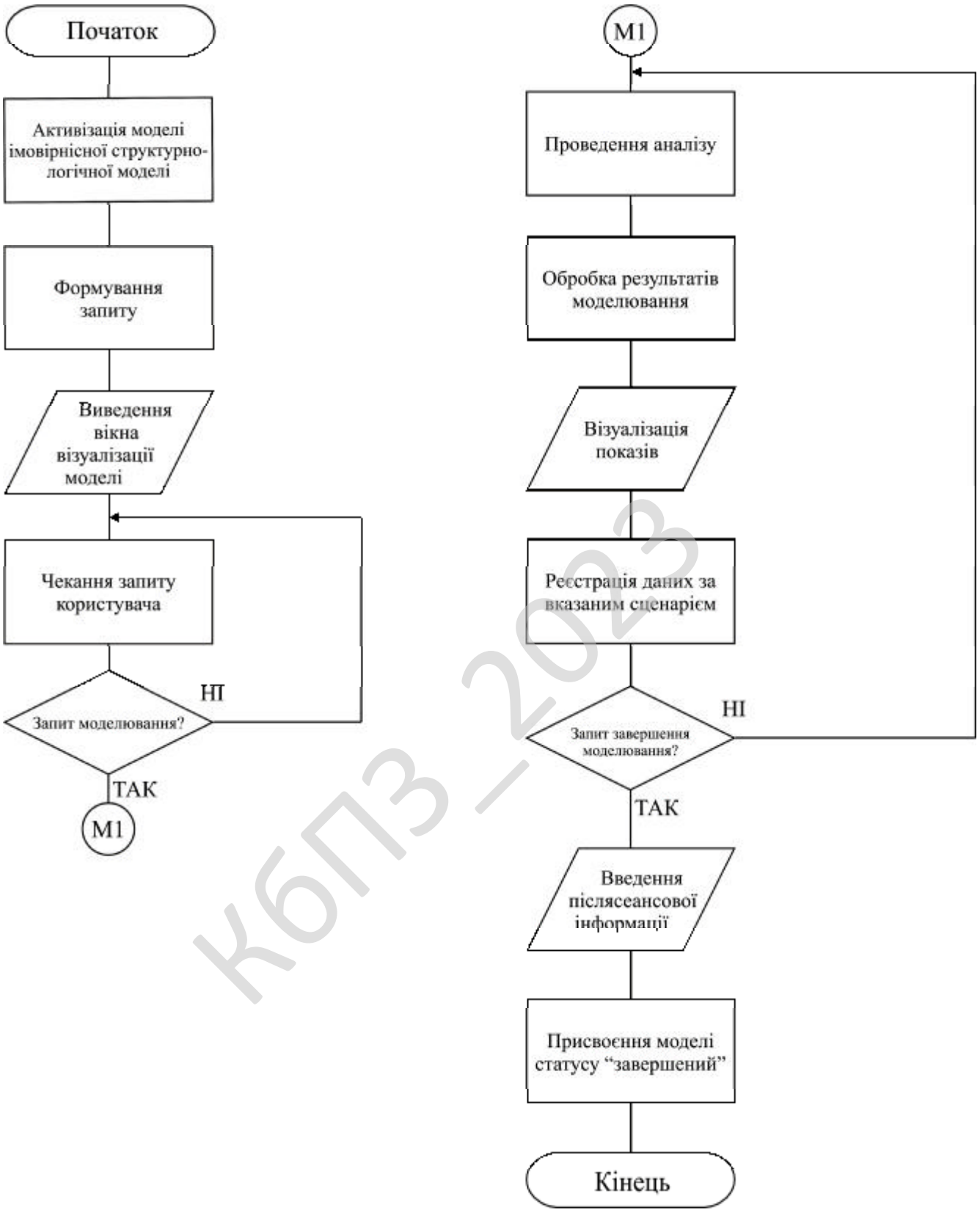


Рисунок 4.2 – Блок-схема роботи підпрограми

Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

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

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

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

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

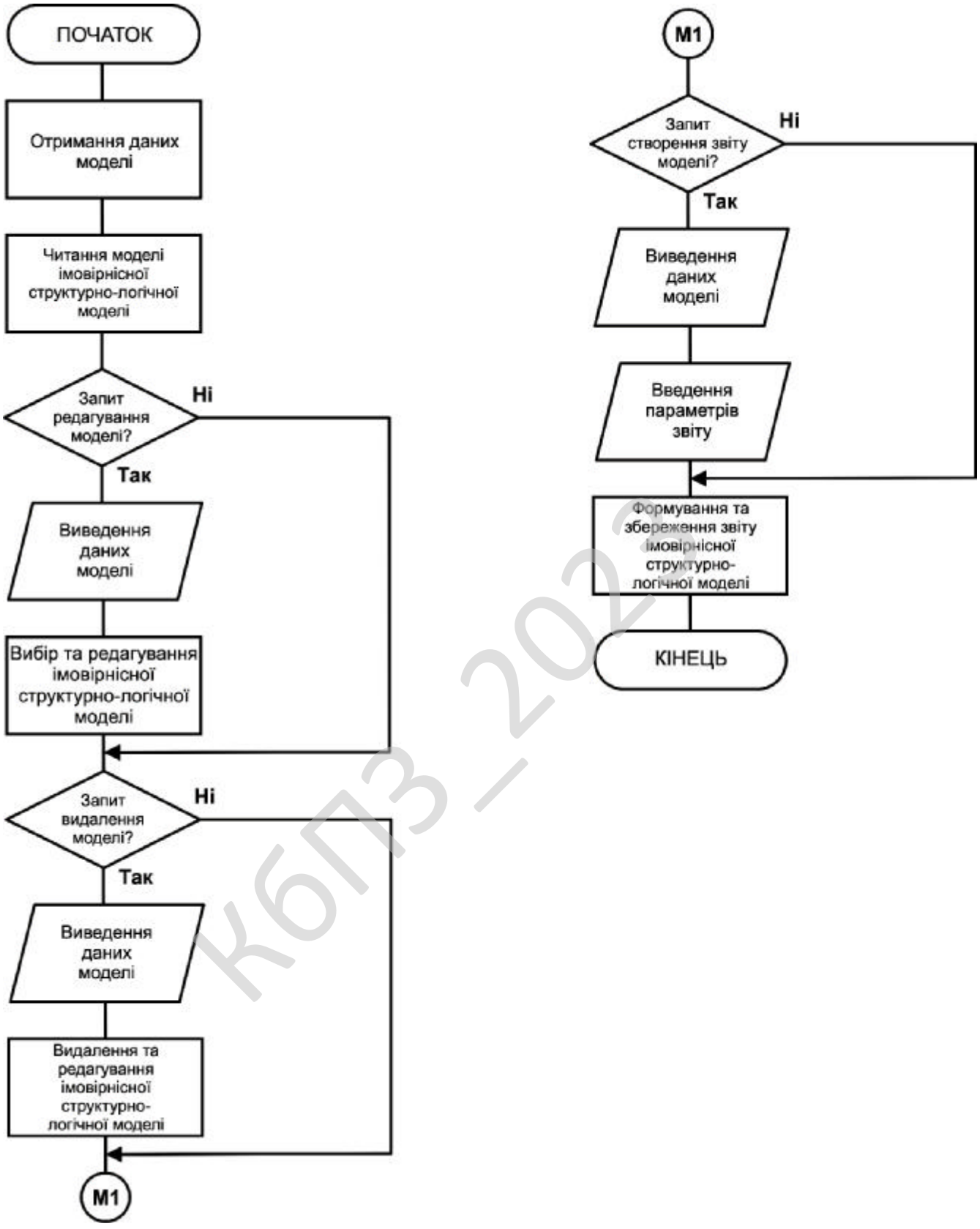


Рисунок 4.3 – Блок-схема роботи підпрограми

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

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

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

Наведемо частину коду розробленої програми. Розглянемо код процедури отримання даних з різного обладнання для управління технологічною безпекою на основі модулю імовірнісних структурно-логічних моделей небезпек виробництв.

```
var
  FormRef: longint;
//змінна ініціалізації бібліотеки
  SeriesNom: integer;
//змінна для вибору графіка
  VAX_Tran: TVAX_Tran;
implementation
{$R *.dfm}
//Функції бібліотеки для зв'язку
```

```
function InitLib(): longint; external 'PIC18_Modbus.dll';
procedure comm_setParam; external 'PIC18_Modbus.dll';
procedure CloseLib; external 'PIC18_Modbus.dll';
function read_RAM(CID:byte;var bufData:array of
byte;adrRAMH:word;adrRAML:word;len:byte): word; external 'PIC18_Modbus.dll';
function write_RAM(CID:byte; bufData:pointer; adrRAMH:word; adrRAML:word;
len:byte): word;  external 'PIC18_Modbus.dll';
//Вихід із програми
procedure TM.btnExitClick(Sender: TObject);
begin
      Close;
end;
//Скидання значень
procedure TM.btnResetClick(Sender: TObject);
begin
      tokEditMin.Text:='0';
      tokEditMax.Text:='20';
      baseTokEdit.Text:='0';
      ValueMemo.Lines.Clear;
      VaxChart.Series[0].Clear;
      VaxChart.Series[1].Clear;
      VaxChart.Series[2].Clear;
      VaxChart.Series[3].Clear;
      VaxChart.Series[0].Title:='-----';
      VaxChart.Series[1].Title:='-----';
      VaxChart.Series[2].Title:='-----';
      VaxChart.Series[3].Title:='-----';
      SeriesNom:=0;
end;
procedure TM.FormCreate(Sender: TObject);
begin
      ValueMemo.Lines.Clear;
      VaxChart.Title.Text.Clear;
      SeriesNom:=0;
end;
//Встановлення зв'язку
procedure TM.btnConnectClick(Sender: TObject);
begin
      FormRef:=InitLib();
//Підключення бібліотеки
      comm_setParam;
//Установка параметрів
```

```
      if FormRef=0 then showMessage('Помилка установки зв'язку')
      else showMessage('Зв'язок установлено');
      btnConnect.Enabled:=False;
end;
procedure TM.FormClose(Sender: TObject; var Action: TCloseAction);
begin
      if FormRef>0 then CloseLib;
//Закриття при виході із програми
end;
procedure TM.btnStartClick(Sender: TObject);
type pBwr=^byte;
     p=^byte;
var tokMin,tokMax,lenWr,Len,n,i,tokBase: integer;
    BufWr,BufRd,temp:array [0..511] of byte;
//Буфери запису/читання й тимчасовий буфер
    Bwr: pBwr;
    B: p;
    ResBuf: array [0..255] of real;
//буфер результатів
begin
      ValueMemo.Lines.Clear;
      tokBase:=strtoint(baseTokEdit.Text);
// значення базису
      tokMin:=strtoint(tokEditMin.Text);
// початкове значення введене користувачем
      tokMax:=strtoint(tokEditMax.Text);
//кінцеве
//перевірка введених значень
      if((tokMin<0) or (tokMin=tokMax)) or ((tokMax<tokMin) or (tokMax>20)) then
      begin
            showMessage('Невірний діапазон введених значень');
            exit;
      end;
      if (tokBase<0) or (tokBase>20000) then
      begin
            showMessage('Невірний діапазон введених значень');
            exit;
      end;
//посилка значення базису
      n:=65535*tokBase div 20000;
      BufWr[0]:=lo(n);
      BufWr[1]:=hi(n);
```

```
      B:=@(BufWr[0]);
      write_RAM($02,B,$00,$02,$02);
      sleep(50);
      ConvertToHex(tokMin,tokMax,BufWr,LenWr);
//одержання буфера запису
      n:=0;
      Bwr:=@(BufWr[0]);
      while n<LenWr+1 do
      begin
write_RAM($02,Bwr,$00,$00,$02);
//запис даних
            sleep(50);
//затримка перед читанням даних
            read_RAM($02,BufRd,$00,$04,$02);
//читання даних
            for i:=0 to 1 do temp[i+n]:=bufrd[i];
//збереження отриманих значень у масив
            inc(Bwr,2);
            n:=n+2;
      end;
      ConvertToDec(Temp,ResBuf,LenWr,Len);
//одержання результатів
      for n:=0 to Len do ValueMemo.Lines.Add(FloatToStr(ResBuf[n],ffFixed,3,3));
//виведення значеня
      showMessage('Виміри кінчені');
end;
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.

У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Сервіс; Структурно-логічна модель; Налаштування; Довідка.

– Вікно вибору необхідного розділу.

– Функціональних кнопок ПЗ.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

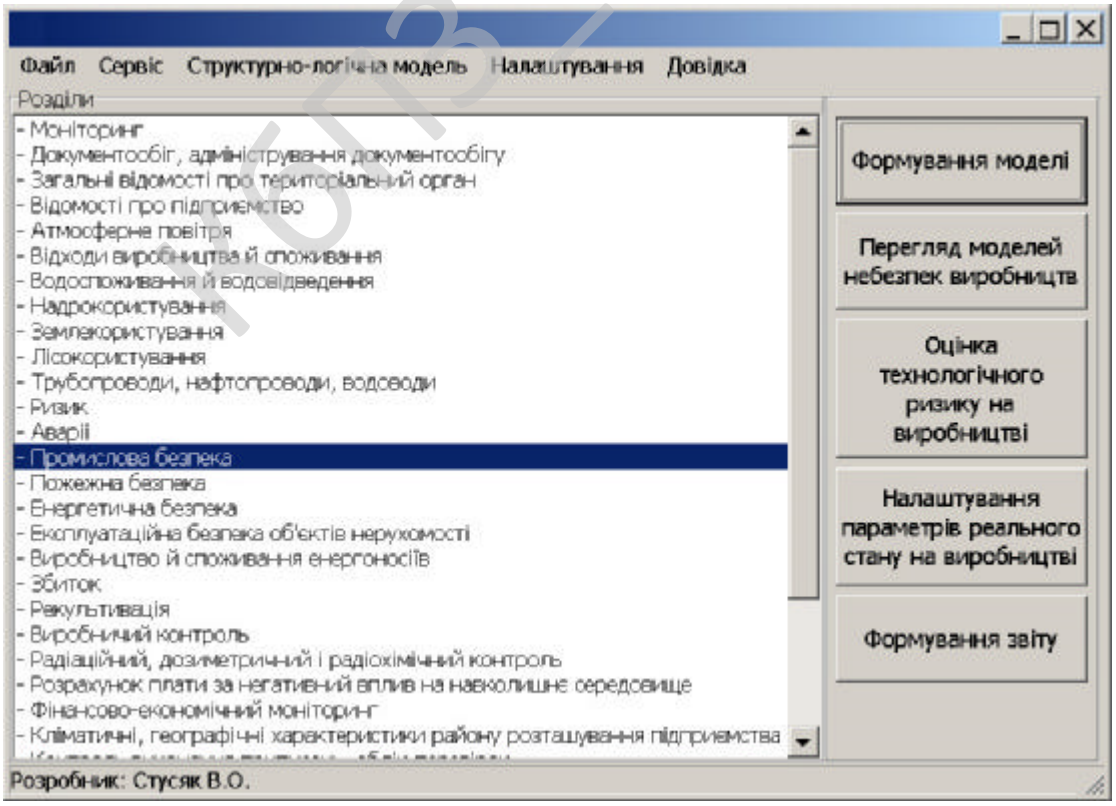


Рисунок 5.1 – Головне вікно розробленого ПЗ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1.

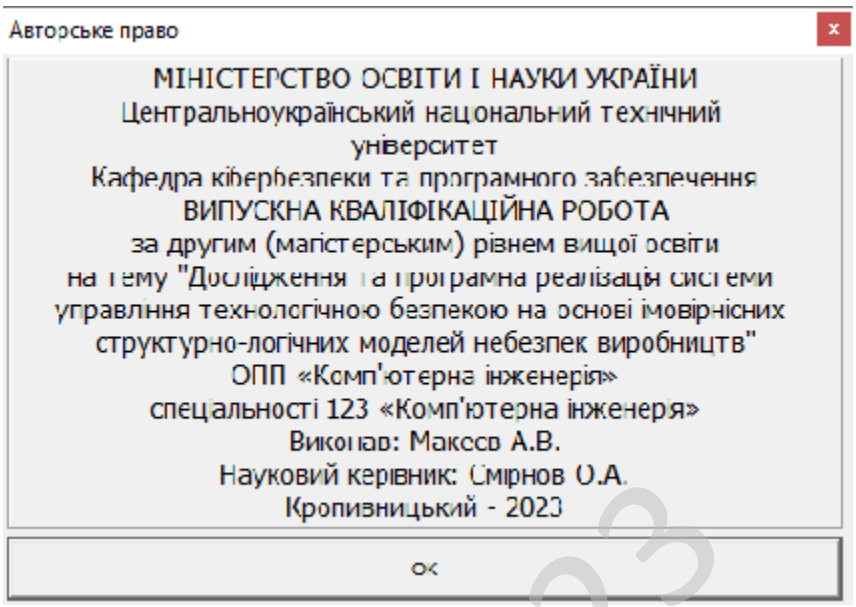


Рисунок 5.2 – Авторське право

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ - «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

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

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

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Метою розробки є дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Об'єктом дослідження є процес управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Предметом дослідження є методи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Методи дослідження базуються на методах теорії графів і булевої алгебри , методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

– Розроблено вітчизняний продукт управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

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	280
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	28000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

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

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

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

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

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

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

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 93 = 157 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	157	Ф 7.1-7.4
Впровадження	13	Д13
Всього	198	–

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.

$$Ч = \frac{198 \cdot 1}{60 - 5} = 3,6 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4,17
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	35,33

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{35 \cdot 3}{1,2} = 87,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 87,5/(60 \cdot 8) = 0,18 \text{ ставки.}$$

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

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.

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,6	8084,5	87312,6
Інженер-електронщик	0,18	7000	3780
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,5	7000	10500
Системний програміст	0,25	7000	5250
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,5	9000	13500
Всього за період розробки	$R_{сп}$ = 7,03	-	$Ф_{роб}$ = 178842,6

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.

$$З_{сд} = \frac{178843}{7,03 \cdot 60} = 424 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

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

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Компбест за 06.10.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		7347
Процесор	Intel Core i7-4770K 3.5GHz/5GT/s/8MI s1150	-
Системна плата	MSI H81M-S03 (s1150, DDR3, USB 3.(INTEL H81 , PCI-Ex16) з підтримкоі відеоядра процесора	-
Відеокарта	nVidia GeForce GTX 660, 2 GB GDDR5, 192 bit	-
Жорсткий диск	240 GB SSD	-
Оперативна пам’ять	Samsung DDR3-1333 16Gb PC3-10600R EC(Registered (2x8Gb)	-
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 450W(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-les chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

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.

$$З_о = 424 \cdot 198/280 = 300 \text{ грн.}$$

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, %.

$$З_д = 300 \cdot 10 \cdot 0{,}01 = 30 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(300+30) = 73 \text{ грн.}$$

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, %.

$$Г_{осп} = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

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, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 210 \cdot 1 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 22,4 грн./шт., DVD-R box – 33,6 грн./шт.

$$З_{М2} = 100 \cdot 33{,}6 = 3360 \text{ грн.}$$

Згідно 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.

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

$$З_М = (105 + 3360 + 1702)/280 = 18 \text{ грн.}$$

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, %.

$$О_п = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 280 прим.):

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

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

$$A_м = 190286 \cdot 3/(280 \cdot 12) = 170 \text{ грн.}$$

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

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

$$C_п = 300+30+73+45+18+45+170 = 681 \text{ грн.}$$

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. Основна зарплата виконавців	$З_о$	300
2. Додаткова зарплата виконавців	$З_д$	30
3. Відрахування на соціальні потреби	$C_{оц}$	73
4. Загальногосподарські витрати	$Г_{осп}$	45
5. Витрати на матеріали	$З_М$	18
6. Освоєння нових операційних систем, мов програмування	$О_п$	45
7. Амортизація основних фондів	$A_м$	170
8. Повна собівартість програмного забезпечення	$C_п$	681
9. Плановий прибуток	$П_р$	341
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	1022
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	204,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1226,4

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 681 = 341 \text{ грн.}$$

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	–	1226
Всього капітальних витрат	–	1226

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	30000	25000
2. Витрати на електроенергію	$З_{ел}$	1488	1030
3. Витрати на амортизацію	$З_{ам}$	0	307
Всього витрат за рік	$І$	31488	26337

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 витрати на обслуговування системи зменшились з 30000 грн до 25000 грн на рік.

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	–	1226	–	306,5
Всього відрахувань	-	–	1226	–	306,5

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

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

$$З_{ел\ баз} = 0{,}545 \cdot 1300 \cdot 2{,}1 = 1487{,}85 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 900 \cdot 2{,}1 = 1030{,}05 \text{ грн.}$$

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.

$$Ев = (1022 - 681) \cdot 280 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 28000) \cdot 3/12 = 50208 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{1760367}{(1022 - 681) \cdot 280 \cdot 12 / 3} = 4{,}5 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (31488\text{-}26337)\text{-}0{,}25 \cdot 1226 = 4845 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	280
2. Повна собівартість розробленої програми	Грн.	681
3. Ціна розробленої програми	Грн.	1022
4. Плановий прибуток від реалізації розробленої програми	Грн.	341
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	95480
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	50208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4845
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,24

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{1226}{31488 - 26337} = 0,24 \ року .$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Комп’ютер – невід’ємна складова сучасного життя. За допомогою обчислювальної техніки вирішують складні робочі задачі, ведуться наукові дослідження, створюються архітектурні креслення і твори мистецтва. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп’ютерної техніки. Незважаючи на видиму безпеку та розвитку сучасних технологій, при роботі за комп'ютером є ряд чинників, які можу вплинути на здоров'я людини. Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп’ютером.

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, затверджені наказом Мінсоцполітики від 14.02.2018р. № 207, зареєстровані в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 [2].

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ.

У розділі даної магістерської роботи висвітлюються основні питання охорони праці працівників, робота яких пов'язана з роботою за комп'ютером, планування робочого приміщення, де працюють користувачі ПК; параметри мікроклімату, освітленість робочих місць та виробничих приміщень; шумові завади.

Правильна організація і раціональне устаткування робочого місця можливість ефективно і з якнайменшими витратами праці виконувати свої функції, плідно спілкуватися співробітниками і підлеглими, підтримувати високу працездатність і робочий настрій.

Велике значення має раціональна конструкція і розташовує елементів робочого місця, що важливе для підтримки оптимальної робочої пози людини-оператора, а також необхідно дотримувати правильний режим праці і відпочинку.

Що стосується питання охорони праці людини необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності.

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

Робота працівників пов'язана з роботою за комп'ютером, тому актуальною є розгляд саме умов праці та стану охорони праці працівників які постійно працюють з комп'ютерною технікою.

Завдання даного розділу полягає у тому, щоб розробити якісний програмний продукт необхідно організувати безпеку на робочому місці програміста. Під час проектування безпеки робочому місці з ПК необхідно домагатися високої якості та надійності технічного забезпечення, але й створювати комфортні параметри довкілля для розробників.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5,4
Довжина	6
Висота	2,75

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого·

Геометрична характеристика	Одиниця виміру	Нормативне значення *	Фактичне значення
Площа, S	$м^2$	не менше 6.0	8,1
Обсяг, V	$м^3$	не менше 20.0	24,3

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працюють 4 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних

машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], 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 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 поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Розрахункова частина

Для захисного штучного заземлення будемо застосовувати вертикальні електроди з сталевого прокату круглого перерізу діаметром 35 мм., довжиною *L=1,5 м.,* та горизонтальний електрод – метелева полоса з перетином 35·4 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,5 м*. Відстань між вертикальними заземлювачами (електродами) *A*=2 *м*. Глибина закладення горизонтального контура заземлення *t=0,7 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+1,5/2=1,45 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [11];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Діаметр ветикального електрода (задан):

$$Dв = 35 \text{ мм.} = 0,035 \text{ м.}$$

Відношення *A/L=2/1,5=1,33.*

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

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

$$=0,366(54,5/1,5)[(lg(2\cdot1,5/0,035)+(1/2\ )lg((4\cdot1,45+1,5)/(4\cdot1,45-1,5))]=$$

$$=27,2\ Ом.$$

Визначаємо коефіцієнт екранування ветикальних електродів $Кев=0,53$ при орієтовній кількості ветикальних електродів, яке дорівнює 5 [11].

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

$$N=R_O /(Кев\ R_{ЗН})= 27,2 / (0,53\cdot4)=12,8 \approx 13\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2\cdot13=26,9 \approx 30\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

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

$$=0,366(40\cdot5/30)\cdot lg((2\cdot30^2)/(0,035\cdot0,7))=20,6\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичної зони для з'єднуючої полоси [11]:

$B = 35\ мм. = 0,035\ м.$- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [11]:

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

$$=(27,2\cdot\ 20,6)/(27,2\cdot\ 0,55+\ 7\cdot\ 20,6\cdot\ 0,53)=3,38\ Ом.$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \le R_{ЗН}$ виконується ($3,38 \le 4$).

Так як R суттево быльше $R_{ЗН}$, зменшимо кількість вертикальних електродів до 11 і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює *11* при $R =3,9\ Ом.$

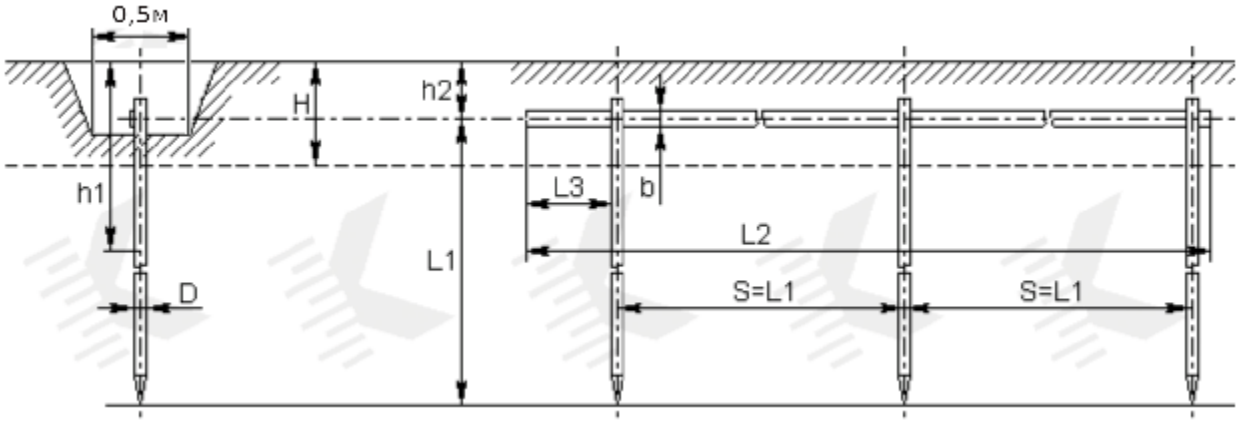


Рисунок 8.1 – Схема штучного заземлення.

При необхідності можна зменшити кількість електродів заземлювача зменшивши загальний опір розтіканню електричного струму заземлювача методом зменшення питомого опору ґрунта, домішуючи у грунт безпосередньо навколи електродів заземлювача розчини солей NaCl, CaCl, сажу, соду, шлак, коксову дрібницю, або спеціальні суміші.

8.5 Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

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 управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

– Досліджена система управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності

предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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

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

Програма реалізована на мові високого рівня RAD Studio Delphi 10. 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 FEAL.

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 4845 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,24 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Макеєв А.В. Дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Закон України про використання ядерної енергії та радіаційну безпеку. м. Київ, 1995 рік. N 39/95-ВР. – с.43.

3. Закон України «Про об'єкти підвищеної небезпеки» 18.01.2001 р, N 2245-III //ВВРУ. – К., 2001. -№15 – с. 73.

4. Закон України “Про захист населення і територій від надзвичайних ситуацій технологічного та природного характеру” від 8. 06. 2000 р. N 1809-III //ВВРУ. – К., 2000. -№40 – с. 37.

5. Закону України “Про пожежну безпеку“ від 17.13.93 р. N 3745-XII //ВВРУ. – К., 1994. -№5 – с. 23.

6. Закон України "Про страхування" від 7.03.1996 №85/96 (Відомості Верховної Ради України, 1996 р., N 18. – с.79.

7. Постанова КМ України "Про єдину державну систему запобігання і реагування на надзвичайні ситуації технологічного та природного характеру". від 3.08.98 № 1198.//ОВУ. –К.,1998. – №31. – с.41.

8. Закон України "Про ліцензування певних видів господарської діяльності"

9. Закон України "Про стандартизацію" від 17.05. 2001 N 2408-III.

10. Постанова Кабінету Міністрів «Про ідентифікацію та декларування безпеки об'єктів підвищеної небезпеки» №956 від 11.07.02//ОВУ. – К., 2003. – №29. – с.23.

11. Постанова Кабінету Міністрів України “Про схвалення Концепції Державної програми забезпечення технологічної безпеки в основних галузях економіки” від 11 червня 2003 р. N 351-р//ОВУ. – К.,2003. – №19. – с.26.

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

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

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

15. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

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

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

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

52. 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.

53. 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.

54. 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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 36-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

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

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 114 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 21.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Смірнов О.А.

Дослідження та програмна реалізація системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2023 року

Файл Main.pas основної програми

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, DbChart, Series,
Math_exp, About;

procedure ConvertToHex(var tmin:integer;var tmax:integer;var Buffer:array of
byte;var Length: integer);stdcall;
procedure ConvertToDec(var Buffer1:array of byte;var Buffer2:array of real;var
Length1:integer;var Length2: integer);stdcall;

type
  TVAX_Tran = class(TForm)
    btnConnect: TButton;
    VaxChart: TDBChart;
    tokEditMin: TEdit;
    tokEditMax: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    btnStart: TButton;
    btnReset: TButton;
    btnSave: TButton;
    btnGraf: TButton;
    ValueMemo: TMemo;
    btnExit: TButton;
    Label3: TLabel;
    Label4: TLabel;
    baseTokEdit: TEdit;
    Series1: TPointSeries;
    Series2: TPointSeries;
    Series3: TPointSeries;
    Series4: TPointSeries;
    procedure btnExitClick(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnConnectClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnStartClick(Sender: TObject);
    procedure btnGrafClick(Sender: TObject);
    procedure btnSaveClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  VAX_Tran: TVAX_Tran;
  FormRef: longint;     //змінна ініціалізації бібліотеки
  SeriesNom: integer;     //змінна для вибору графіка

implementation

{$R *.dfm}
//Функції бібліотеки для зв'язку із приладом по протоколі Modbus
function InitLib(): longint; external 'PIC18_Modbus.dll';
procedure comm_setParam; external 'PIC18_Modbus.dll';
procedure CloseLib; external 'PIC18_Modbus.dll';
function read_RAM(CID:byte;var bufData:array of
byte;adrRAMH:word;adrRAML:word;len:byte): word; external 'PIC18_Modbus.dll';
```

```
function write_RAM(CID:byte;bufData:pointer;adrRAMH:word;adrRAML:word;len:byte):
word;  external 'PIC18_Modbus.dll';

//Вихід із програми
procedure TVAX_Tran.btnExitClick(Sender: TObject);
begin
      Close;
end;

//Скидання значень
procedure TVAX_Tran.btnResetClick(Sender: TObject);
begin
      tokEditMin.Text:='0';
      tokEditMax.Text:='20';
      baseTokEdit.Text:='0';
      ValueMemo.Lines.Clear;
      VaxChart.Series[0].Clear;
      VaxChart.Series[1].Clear;
      VaxChart.Series[2].Clear;
      VaxChart.Series[3].Clear;
      VaxChart.Series[0].Title:='-----';
      VaxChart.Series[1].Title:='-----';
      VaxChart.Series[2].Title:='-----';
      VaxChart.Series[3].Title:='-----';
      SeriesNom:=0;
end;

procedure TVAX_Tran.FormCreate(Sender: TObject);
begin
      ValueMemo.Lines.Clear;
      VaxChart.Title.Text.Clear;
      VaxChart.Title.Text.Add('Графік системи управління технологічною безпекою
на основі імовірнісних структурно-логічних моделей небезпек виробництв
транзистора');
      SeriesNom:=0;
end;

//Установлення зв'язку із приладом
procedure TVAX_Tran.btnConnectClick(Sender: TObject);
begin
      FormRef:=InitLib();    //Підключення бібліотеки
      comm_setParam;     //Установка параметрів COM порту
      if FormRef=0 then showMessage('Помилка установки зв'язку із приладом')
      else showMessage('Зв'язок із приладом установлений');
      btnConnect.Enabled:=False;
end;

procedure TVAX_Tran.FormClose(Sender: TObject; var Action: TCloseAction);
begin
      if FormRef>0 then CloseLib;    //Закриття порту передачі даних при виході
із програми
end;

procedure TVAX_Tran.btnStartClick(Sender: TObject);
type pBwr=^byte;
     p=^byte;
var tokMin,tokMax,lenWr,Len,n,i,tokBase: integer;
    BufWr,BufRd,temp:array [0..511] of byte;    //Буфери запису/читання й
тимчасовий буфер
    Bwr: pBwr;
    B: p;
    ResBuf: array [0..255] of real;    //буфер результатів
begin
      ValueMemo.Lines.Clear;
      tokBase:=strtoint(baseTokEdit.Text);   //значення базового значення
параметру технологічної безпеки
      tokMin:=strtoint(tokEditMin.Text);   //початкове значення значення
параметру технологічної безпеки уведене користувачем
      tokMax:=strtoint(tokEditMax.Text);   //кінцеве
```

```
      //перевірка уведених значень
      if((tokMin<0) or (tokMin=tokMax)) or ((tokMax<tokMin) or (tokMax>20)) then
      begin
            showMessage('Невірний діапазон уведених значень');
            exit;
      end;
      if (tokBase<0) or (tokBase>20000) then
      begin
            showMessage('Невірний діапазон уведених значень');
            exit;
      end;
      //посилка у систему управління технологічною безпекою на основі
імовірнісних структурно-логічних моделей небезпек виробництв значення базового
значення параметру технологічної безпеки
      n:=65535*tokBase div 20000;
      BufWr[0]:=lo(n);
      BufWr[1]:=hi(n);
      B:=@(BufWr[0]);
      write_RAM($02,B,$00,$02,$02);
      sleep(50);
      ConvertToHex(tokMin,tokMax,BufWr,LenWr);    //одержання буфера запису
      n:=0;
      Bwr:=@(BufWr[0]);
      while n<LenWr+1 do
      begin
            write_RAM($02,Bwr,$00,$00,$02);   //запис даних у систему управління
технологічною безпекою на основі імовірнісних структурно-логічних моделей
небезпек виробництв
            sleep(50);  //затримка перед читанням даних із системи управління
технологічною безпекою на основі імовірнісних структурно-логічних моделей
небезпек виробництв
            read_RAM($02,BufRd,$00,$04,$02);  //читання даних із системи
управління технологічною безпекою на основі імовірнісних структурно-логічних
моделей небезпек виробництв
            for i:=0 to 1 do temp[i+n]:=bufrd[i];    //збереження отриманих
значень у масив
            //for i:=0 to 3 do ValueMemo.Lines.Add(inttohex(BufRd[i],4));
            //ValueMemo.Lines.Add('==========');
            inc(Bwr,2);
            n:=n+2;
      end;
      ConvertToDec(Temp,ResBuf,LenWr,Len);    //одержання остаточних результатів
      for n:=0 to Len do ValueMemo.Lines.Add(FloatToStr(ResBuf[n],ffFixed,3,3));
//вивід значень
      showMessage('Виміри кінчені');
end;

{************************************************************
Процедура перетворення значень уведених користувачем (tokMin і tokMax)
у масив шіснадцяткових чисел(Buffer) для запису в RAM,
Length - довжина масиву. Buffer і Length - вихідні значення.
************************************************************}
procedure ConvertToHex(var tmin:integer; var tmax:integer;var Buffer:array of
byte;var Length: integer);
var i:integer;
    min,max,n:word;
begin
      min:=65535*tmin div 20;
      max:=65535*tmax div 20;
      n:=min;
      i:=0;
      while n<=max+1 do
      begin
            Buffer[i]:=lo(n);
            Buffer[i+1]:=hi(n);
            n:=n+256;
            i:=i+2;
            if i>=511 then break;
      end;
```

```
      Length:= i-1;
end;

{*********************************************************
Процедура перетворення масиву даних зчитаних із системи управління технологічною
безпекою на основі імовірнісних структурно-логічних моделей небезпек
виробництв(Buffer1)
у масив десяткових чисел(Buffer2) для подальшої побудови графіка.
Length1 - довжина масиву Buffer1, Length2 - довжина масиву Buffer2.
Buffer2 i  Length2 - вихідні значення.
*********************************************************}
procedure ConvertToDec(var Buffer1:array of byte;var Buffer2:array of real;var
Length1:integer;var Length2: integer);
var i,n:integer;
begin
      n:=0;
      i:=0;
      while n<=Length1 do
      Begin
            Buffer2[i]:=((5*Buffer1[n])/$FF30)+((5*256*Buffer1[n+1])/$FF30);
            n:=n+2;
            i:=i+1;
      end;
      Length2:= i-1;
end;

//Вивід графіка системи управління технологічною безпекою на основі імовірнісних
структурно-логічних моделей небезпек виробництв
procedure TVAX_Tran.btnGrafClick(Sender: TObject);
var n:integer;
    i:real;
    //B_data:TStringList;
begin
      //B_data:=TStringList.Create;
      //B_Data.LoadFromFile('Results.txt');
      if SeriesNom>3 then SeriesNom:=0;
      i:=0;
      for n:=0 to ValueMemo.Lines.Count-1 do
      Begin
            VaxChart.Series[SeriesNom].Title:=baseTokEdit.Text+' мкА';

VaxChart.Series[SeriesNom].AddXY(strtofloat(ValueMemo.Lines[n]),i,''{,Color});
      //Вивід графіка по значанням з таблиці
            i:=i+0.078;
      end;
      SeriesNom:=SeriesNom+1;
      {if (B_data.Count>0) then    //перевірка наявності даних у файлі
      begin
            for n:=0 to B_Data.Count-1 do
                  VaxChart.Series[0].AddXY(strtofloat(B_data.Strings[n]),n);
  //Вивід графіка за значеннями з файлу
      end;}
end;

//Збереження отриманих результатів у файл
procedure TVAX_Tran.btnSaveClick(Sender: TObject);
var B_data:TStringList;
    n:integer;
begin
      B_data:=TStringList.Create;
      if SeriesNom>3 then SeriesNom:=0;
      B_data.Add('Вихідні дані для базового значення параметру технологічної
безпеки в '+baseTokEdit.Text+' мкА');
      for n:=0 to ValueMemo.Lines.Count-1 do
            B_data.Add(ValueMemo.Lines[n]);
      B_data.SaveToFile('Results'+inttostr(SeriesNom)+'.txt');
      //ValueMemo.Lines.SaveToFile('Results'+inttostr(SeriesNom)+'.txt');
end;
```

```
end.
```

Файл Math_exp.pas - обробка отриманих у ході роботи системи управління технологічною безпекою на основі імовірнісних структурно-логічних моделей небезпек виробництв даних та математичні розрахунки

```
unit Math_exp;



interface

uses
  Windows,  SysUtils, Classes,  Dialogs, Math, Forms, Controls, StdCtrls;

  const _PI = 3.14159265358979323846264 3;
  const  PI_2: extended = _PI/2;
  const  _LDSF = 3;
  const  _LDSC = 4;
  const  _FUNC = 1;
  const  _COP  = 2;



type
TAddress = Cardinal;

type TFloatType = double;

type PFloatType = ^TFloatType;



type
TArray4 = array [1..4] of byte;
type TCode = array of Byte;
type TArray = array of TFloatType;
type PArray = ^TArray;
type TAddFunc = function(N: Integer; SF: TArray): TFloatType;



type TArray = array of Integer;
type TArray = array of Cardinal;


type TIntType = array of Cardinal;



type TIntFunc  = array of
                  record
                   Type:   Cardinal;          //повертається тип ф-ії
                   Addr: Cardinal;             //адреса ф-ії
                   N:    Integer;              //N=-1 Infinite
                   TV:   array of Cardinal;    //масив типів змінних (Size=N)
                  end;

type TAlgFunction = array of
                     record
                      Name: String;        //ім'я ф-ії
                      Func: TIntFunc;      //характеристики ф-ій з даним ім'ям
                     end;
```

```
type TAlgOperation = array of
                     record
                      PR:  Cardinal;        //пріоритет операції  (крім '0'!)
                      Name:  String;        //ім'я символьної операції
                      NFunc: Cardinal;      //номер масиву відповідної ф-ії
                      GS:  Integer;         //для одноарг. операцій:
розташування
                                            //((-1)спереду;(-2)позаду)
                     end;                   //для багатоарг. операцій: >= 0 !!!

                     //пріоритет кожної одноарг. операції вище кожної
багатоарг. операції !!!



type TAlgObject = array of
                  record
                   Name: String;
                  Type: Cardinal;          //тип змінної
                   Addr: Cardinal;
                  end;

type TAlgLoadFunction = array of
                        record
                         Type: Cardinal;    //тип змінної для завантаження
                          Addr: Cardinal;
                         end;


// Bri = 1  якщо ф-ія встановлена й NFi - номер масиву
type
TAlgBracket = record
               Br1: Integer;    // (,) не використовується !!!
               NF1: Cardinal;
               Br2: Integer;    // [,]
               NF2: Cardinal;
               Br3: Integer;    // {,}
               NF3: Cardinal;
               Br4: Integer;    // <,>
               NF4: Cardinal;
              end;

type
 PFunc      = ^TFunc;



 TFunc = record
               NL: Cardinal;               //порядковий номер команди на рівні
               S:  Cardinal;               //Stack
               N:  Cardinal;               //Node
            Type:  Cardinal;              //повертається тип
              COP:  Cardinal;
            NFunc:  Cardinal;              //номер масиву ф-ій
               NF:  Cardinal;              //номер ф-ії в масиві
             NObj:  Cardinal;              //номер масиву змінних
               FD:    Double;             //константа
              Arg:  Cardinal;              //число змінних (для infinite)
              Id:    Cardinal;
              Id:    Cardinal;

         end;


type
TArrayOfPFunc = array of PFunc;
```

```
type
TFunction = record
                 Interp: TArrayOfPFunc;
                   Comp: TCode;
                  ICode: TAddress;
                  end;

type PFunction = ^TFunction;

type TNStack = record
                S: Cardinal; //поточний номер стека
                N: Cardinal; //число вузлів, що випливають
              end;


type
PNode = ^TNode;
TNode = record
         Stack: TNStack;
         Id:    Cardinal;
         Id:    Cardinal;
         NL:     Cardinal;           //порядковий номер команди (складова
рівня);
         NCOP:   Cardinal;           //операція (номер масиву) (складова вузла);
         NFunc:  Cardinal;           // ф-ія (номер масиву ф-ій)
         NObj:   Cardinal;           //номер масиву змінної
         COP:    Integer;            //COP: = 1( ф-ія) в Func номер масиву ф-ій;
                                     //2(операція) в NCOP номер масиву;
                                     //3(завантаження змінної) в NObj - номер
масиву;
                                     //4(завантаження константи) в FD -
константа;
         FD:     Double;             //константа
         Expression: String;
         Ch: PNode;
         Nb: PNode;
         Pr: PNode;
       end;

type
TDataNode = record
             N:     Cardinal;
             Id:    Cardinal;
             NCOP:  Cardinal;
             NFunc: Cardinal;
             NObj:  Cardinal;
             COP:   Integer;
             FD:    Double;
             FX:    Extended;
             Expression: String;
            end;

type
TLevel = record
          NL:  Cardinal;               //порядковий номер команди (складова
рівня);
          Id: Cardinal;
          Expression: String;
        end;

type PLevel = array of TLevel;

type
TArray = array of String;

type
TParamList = record
               Name: String;
```

```
              Value: TFloatType;
             end;

{type
TArray = array of Integer;
}

type
TString1 = String[1];
type
TAlgebra = record
             Name:  String;
            Types:  TArray;
              Obj:  TAlgObject;
              Opr:  TAlgOperation;
           ExFunc:  TAlgFunction;
           LdFunc:  TAlgLoadFunction;
          Bracket:  TAlgBracket;
           end;


type TExternalException = procedure (TypeError: Integer); {of object;} stdcall;


type

  TForeval = class

  private

    Alg: TAlgebra;
    CurrentExpression: String;      //поточне вираження
    LFunc    : Cardinal;          //довжина скомпільованого вираження
    StackSize: Cardinal;          //розмір стека
    Arg : TArray;               //список аргументів у випадку > 2 арг.(відлік з
'1')
    Func: TFunction;                    //виконується код, що, скомпільованого
вираження
    Tree: PNode;                                //дерево розбору вираження
    //F_SyntaxExtension:      Boolean; //вмик./вимк.: імена ф-ій > 1; '{' ;
'['; '!'; '^'; .
    BeginTree: PNode;                     //покажчик початку дерева (не обов'язково)
    F_VarShift:           Boolean;  //розпізнавання var&param у
верхньому/нижньому/обох регістрах.
    F_FuncShift:          Boolean;
    F_SyntaxErrorCode: Cardinal;    //код помилки при розборі
    F_SyntaxErrorString: String;   //рядок помилки
    F_CalcErrorCode: Integer;    //вичислит. помилка
    F_InternalError: Integer;
    F_ShowException: Boolean;      //вмик./вимк. показ виключень
    F_ExternalException: Boolean;  //вмик./вимк. зовнішнього оброблювача
виключень
    //BeginLinker: Integer; //0,1 - початок/кінець трансляції - використовується
при обробці помилок
    F_BeginCalc: Integer; //0,1 - початок/кінець трансляції - використовується
при обробці помилок



    //Alg:
    F_NumberStack: Cardinal;
    F_Address: Cardinal;
    F_NumberArg: Cardinal;
    F_FConst: TFloatType;
    ICOMP: TCode;
    F_SyntaxErrorString1: String;
    F_Id: Cardinal;
    F_ResultType: Cardinal;
    F_ExtNumberArg: Cardinal;
```

```
    function  FindNumberPR(PR: Cardinal): Cardinal;
    function  FindOperation(S: String): Cardinal;
    procedure FindSingleOP(S: String; var NCop: Integer);
    procedure FindCallFunction(var nf: Integer);
    procedure FindLdFunction(var nf: Integer);
    procedure FindConstLdFunction(var nf: Integer);
    procedure FindNumberLevel(var ANL: TArray);
    procedure InsertBracketOperation(S: String; var S1: String);
    procedure IntToHex(N: TAddress; var AH: TArray4);
    procedure AddAddress(Adr: TAddress);
    procedure AddCommand(I: Integer);
    procedure AddByte(B: Byte);
    procedure AddWord(B1: Byte; B2: Byte);
    procedure Compile;
    procedure CALLA;
    procedure CALLB;
    procedure RET;
    procedure FSTP(N: TAddress);
    procedure FLD(N: TAddress);
    procedure MOVAM(N: TAddress);
    procedure MOVMA(N: TAddress);
    procedure MOVA1(N: TAddress);
    procedure PUSHA;
    procedure POPA;


    procedure FindExternalBracket(S: String; var SS: String);
    procedure FindMainOperation(S:String; var NCOP: Integer; var NPos:
Cardinal);
    procedure StringAnalizator(S:String; var ND: TDataNode; var HS:PLevel);
    procedure WriteCode;
    procedure WriteLevelParam(H: PNode; HS: PLevel; N: Cardinal);
    procedure SubstParam(S1: String;  var C: TFloatType; var SB: Integer);
    function  PointToDec(S: String):String;
    procedure CreateInitData;
    procedure FindVar(S1: String; var ni: Integer);
    procedure SyntaxAnalizator(S:String; var ND: TDataNode; var LV:PLevel;  var
EOFN: Boolean);
    procedure NumberAnalizator(S: String; var ND: TDataNode;  var EOFN:
Boolean);
    procedure WriteNodeParam(ND: TDataNode);
    procedure InitNode(var ND: TDataNode);
    procedure WriteDataCode;
    procedure FindFunction(S: String;  var ND: TDataNode; var LV: PLevel;  var
BH: Boolean);
    procedure XchangeAddingBracket(S: String; var S1: String);
    procedure PrevTreat(S: String; var S1: String);
    //procedure SetExpression(S: String);
    procedure FreeTree;
    procedure FreeFunction;
    procedure Linker(S:String);
    procedure SetStackSize;
    procedure CodeGeneration;
    //function  Calculation: TFloatType;
    procedure FindManyArg(S: String; var Arg: TArray);
    function  Calc: TFloatType;


  protected
   //ExtException: TExternalException; //зовнішній оброблювач виключень
   procedure  CalcException(Sender: TObject; E: Exception);  virtual;


  public

   //procedure InsertBracketOperation(S: String; var S1: String);
   constructor Create;
```

```
   destructor  Destroy;
   //function  ReadErrorCode: Integer;
   //function  CalculateExpression(S: String): TFloatType;
   procedure SetExtExpression(S: String;  var E_Func: TFunction; var Stack:
Cardinal);
   procedure CalcExtFunc(E_Func: TFunction);
   function  CalcExtFunc(Addr: TAddress): TFloatType;
   procedure FreeExtFunc(E_Func: TFunction);
   property  VarShift: Boolean read F_VarShift write F_VarShift default False;
   property  FuncShift: Boolean read F_FuncShift write F_FuncShift default
False;
   property  ShowException: Boolean read F_ShowException write F_ShowException
default True;
   property  SyntaxError: Cardinal read  F_SyntaxErrorCode write
F_SyntaxErrorCode default 0;
   property  SyntaxErrorString: String read F_SyntaxErrorString write
F_SyntaxErrorString;
   property  CalcError: Integer read F_CalcErrorCode write F_CalcErrorCode
default 0;
   property  InternalError: Integer read F_InternalError write F_InternalError
default 0;
   property  ExternalException: Boolean read F_ExternalException write
F_ExternalException default False;
   //property  SetExternalException: TExternalException write ExtException;
   property  BeginCalc: Integer read F_BeginCalc write F_BeginCalc default 0;



   //Algebra:
   property  GetStack: Cardinal read F_NumberStack;
   property  GetAddress: Cardinal read F_Address;
   property  ResultType: Cardinal read F_ResultType;
   property  GetNumberArg: Cardinal read F_NumberArg;   //для Infinite ф-ій:
число аргументів
   property  ExtNumberArg: Cardinal read F_ExtNumberArg write F_ExtNumberArg;
//для Infinite ф-ій:
                                                        //min число аргументів
для виклику ф-ії
   procedure SetType(IdType: Cardinal);
   procedure SetObject(Name: String; Addr: Cardinal; IdType: Cardinal);
   procedure SetFunction(Name: String; Addr: Cardinal; TF: TArray; IdType:
Cardinal);
   procedure SetFunctionEx(Name: String; Addr: Cardinal; Type: Cardinal; IdType:
Cardinal);
   procedure SetLoadFunction(IdType: Cardinal; Addr: Cardinal);
   procedure SetOperation(Name: String; Func: String; PR: Cardinal; GS:
Integer);
   procedure SetBracketOperation(Br: Integer; Name: String);

  published

  end;

var
  ak,bk: array[1..7] of TFloatType;
  C_2dqrPi: extended;

  Foreval: TForeval;

implementation
{$IFDEF TEXTOUT}
uses
Test8;
{$ENDIF}

procedure TForeval.SetType(IdType: Cardinal);
label endp;
var
i: Integer;
```

```
begin
{
for i:=0 to High(Alg.Types) do
begin
 if Alg.Types[i] = IdType then goto endp;
end;
}
SetLength(Alg.Types,Length(Alg.Types)+1);
Alg.Types[High(Alg.Types)]:=IdType;

endp:
end;

procedure TForeval.SetObject(Name: String; Addr: Cardinal; IdType: Cardinal);
label endp;
var
i: Integer;
S: String;
begin
{
S:=Copy(Name,1,Length(Name));
for i:=0 to High(Alg.Obj) do
begin
 if Alg.Obj[i].Type = IdType then
 begin
  if F_VarShift = False then S:=LowerCase(S));
  if S = Alg.Obj[i].Name then goto endp;
 end;
end;
}
SetLength(Alg.Obj,Length(Alg.Obj)+1);
Alg.Obj[High(Alg.Obj)].Type:=IdType;
Alg.Obj[High(Alg.Obj)].Addr:=Addr;
Alg.Obj[High(Alg.Obj)].Name:=Copy(Name,1,Length(Name));

endp:
end;


procedure TForeval.SetFunction(Name: String; Addr: Cardinal; TF: TArray; IdType:
Cardinal);
label endp;
var
N,K,i,j: Integer;
begin
  for i:=0 to High(Alg.ExFunc) do
  begin
   if Name = Alg.ExFunc[i].Name then
   begin
    K:=i;
    N:=High(Alg.ExFunc[K].Func);
    SetLength(Alg.ExFunc[K].Func,Length(Alg.ExFunc[K].Func)+1);
    N:=N+1;
    Alg.ExFunc[K].Func[N].Type:=IdType;
    Alg.ExFunc[K].Func[N].Addr:=Addr;
    Alg.ExFunc[K].Func[N].N:=Length(TF);
    SetLength(Alg.ExFunc[K].Func[N].TV,Length(TF));
    for j:=0 to High(TF) do
    begin
     Alg.ExFunc[K].Func[N].TV[j]:=TF[j];
    end;

    goto endp;
   end;
  end;

  SetLength(Alg.ExFunc,Length(Alg.ExFunc)+1);
  Alg.ExFunc[High(Alg.ExFunc)].Name:=Copy(Name,1,Length(Name));
```

```
  K:=High(Alg.ExFunc);
  N:=High(Alg.ExFunc[K].Func);
  SetLength(Alg.ExFunc[K].Func,Length(Alg.ExFunc[K].Func)+1);
  N:=N+1;
  Alg.ExFunc[K].Func[N].Type:=IdType;
  Alg.ExFunc[K].Func[N].Addr:=Addr;
  Alg.ExFunc[K].Func[N].N:=Length(TF);
  SetLength(Alg.ExFunc[K].Func[N].TV,Length(TF));
  for j:=0 to High(TF) do
  begin
   Alg.ExFunc[K].Func[N].TV[j]:=TF[j];
  end;

endp:
end;

procedure TForeval.SetFunctionEx(Name: String; Addr: Cardinal; Type: Cardinal;
IdType: Cardinal);
label endp;
var
N,K,i,j: Integer;
begin
  for i:=0 to High(Alg.ExFunc) do
  begin
   if Name = Alg.ExFunc[i].Name then
   begin
    K:=i;
    N:=High(Alg.ExFunc[K].Func);
    SetLength(Alg.ExFunc[K].Func,Length(Alg.ExFunc[K].Func)+1);
    N:=N+1;
    Alg.ExFunc[K].Func[N].Type:=IdType;
    Alg.ExFunc[K].Func[N].Addr:=Addr;
    Alg.ExFunc[K].Func[N].N:=-1;
    SetLength(Alg.ExFunc[K].Func[N].TV,1);
    Alg.ExFunc[K].Func[N].TV[0]:=Type;
    goto endp;
   end;
  end;

  SetLength(Alg.ExFunc,Length(Alg.ExFunc)+1);
  Alg.ExFunc[High(Alg.ExFunc)].Name:=Copy(Name,1,Length(Name));

  K:=High(Alg.ExFunc);
  N:=High(Alg.ExFunc[K].Func);
  SetLength(Alg.ExFunc[K].Func,Length(Alg.ExFunc[K].Func)+1);
  N:=N+1;
  Alg.ExFunc[K].Func[N].Type:=IdType;
  Alg.ExFunc[K].Func[N].Addr:=Addr;
  Alg.ExFunc[K].Func[N].N:=-1;
  SetLength(Alg.ExFunc[K].Func[N].TV,1);
  Alg.ExFunc[K].Func[N].TV[0]:=Type;

endp:
end;

procedure TForeval.SetLoadFunction(IdType: Cardinal; Addr: Cardinal);
begin
 SetLength(Alg.LdFunc,Length(Alg.LdFunc)+1);
 Alg.LdFunc[High(Alg.LdFunc)].Type:=IdType;
 Alg.LdFunc[High(Alg.LdFunc)].Addr:=Addr;
end;
procedure TForeval.SetOperation(Name: String; Func: String; PR: Cardinal; GS:
Integer);
label 1;
var
i,nf: Integer;                              //Операції використовують тільки 1,2
- аргументні ф-ії
begin
nf:=-1;
```

```
for i:=0 to High(Alg.ExFunc) do
begin
 if Func = Alg.ExFunc[i].Name then
 begin
  nf:=i; goto 1;
 end;
end;

1:
if nf = -1 then begin {Error} end;
if PR <= 0 then begin {Error} end;

SetLength(Alg.Opr,Length(Alg.Opr)+1);
Alg.Opr[High(Alg.Opr)].Name:=Name[1];
Alg.Opr[High(Alg.Opr)].NFunc:=nf;
Alg.Opr[High(Alg.Opr)].PR:=PR;
Alg.Opr[High(Alg.Opr)].GS:=GS;

end;

procedure TForeval.SetBracketOperation(Br: Integer; Name: String);
label 1;
var
i,nf: Integer;
begin                                          //Br = 1 (); не використовується
!!!
                                               //Br = 2 []; 3 {}; 4 <>;
nf:=-1;
for i:=0 to High(Alg.ExFunc) do
begin
 if Name = Alg.ExFunc[i].Name then
 begin
  nf:=i; goto 1;
 end;
end;

1:
if (nf = -1) or (Br < 2) or (Br > 4) then begin {Error} end;


{if Br = 1 then
begin
 Alg.Bracket.Br1:=1;
 Alg.Bracket.NF1:=nf;
end
else}
if Br = 2 then
begin
 Alg.Bracket.Br2:=1;
 Alg.Bracket.NF2:=nf;
end
else
if Br = 3 then
begin
 Alg.Bracket.Br3:=1;
 Alg.Bracket.NF3:=nf;
end
else
if Br = 4 then
begin
 Alg.Bracket.Br4:=1;
 Alg.Bracket.NF4:=nf;
end;

end;

procedure TForeval.SetStackSize;
label endp;
begin
```

```
if F_SyntaxErrorCode <> 0 then goto endp;
Tree:=BeginTree;   //не обов'язково, тому що читання дерева - циклічно

while Tree^.Ch <> nil do
begin
Tree:=Tree^.Ch;
end;
StackSize:=Tree^.Stack.S;

while Tree^.Pr <> nil do
begin
 while Tree^.Nb <> nil  do
 begin
 Tree:=Tree^.Nb;
  while Tree^.Ch <> nil do
  begin
  Tree:=Tree^.Ch;
  end;
if Tree^.Stack.S > StackSize then StackSize:=Tree^.Stack.S;
 end;
 Tree:=Tree^.Pr;
if Tree^.Stack.S > StackSize then StackSize:=Tree^.Stack.S;
end;

endp:
end;

procedure TForeval.Linker(S: String);
label endp;
var
HS: PLevel;
ND: TDataNode;
H,H1: PNode;
EOFN: Boolean;
i: Cardinal;
begin
if F_SyntaxErrorCode <> 0 then goto endp;
New(Tree);
Tree^.Ch:=nil;
Tree^.Nb:=nil;
Tree^.Pr:=nil;
BeginTree:=Tree;
//вершина дерева:
Tree^.Expression:=Copy(S,1,Length(S));
Tree^.Stack.S:=1;
EOFN:=True;

SyntaxAnalizator(Copy(Tree^.Expression,1,Length(Tree^.Expression)),ND,HS,EOFN);
//спуск:
//COP<>NOP:
while EOFN <> True do
begin
if F_SyntaxErrorCode <> 0 then goto endp;

New(H);
Tree^.Ch:=H;

WriteNodeParam(ND);
WriteLevelParam(H,HS,0);

H^.Stack.S:=Tree^.Stack.S;
H^.Pr:=Tree;
H1:=H;
H^.Ch:=nil;
H^.Nb:=nil;

for i:=1 to High(HS) do
begin
 New(H);
```

```
 H^.Ch:=nil;
 H^.Nb:=nil;
 WriteLevelParam(H,HS,i);
 H^.Stack.S:=H1^.Stack.S+1;
 H1.Nb:=H;
 H^.Pr:=Tree;
 //HS:=HS^.Next;
 H1:=H;
end;
Tree:=H^.Pr^.Ch;
HS:=nil; //??? (звільнення масиву рівня)
SyntaxAnalizator(Copy(Tree^.Expression,1,Length(Tree^.Expression)),ND,HS,EOFN);
end;
//EOFN=True: остання команда вузла (нижній лівий)
WriteNodeParam(ND);

//підйом:
while Tree^.Pr <> nil do
begin
 while Tree^.Nb <> nil  do
 begin
 Tree:=Tree^.Nb;
 SyntaxAnalizator(Copy(Tree^.Expression,1,Length(Tree^.Expression)),ND,HS,EOFN);
//COP<>NOP:
while EOFN<>True do
 begin
 if F_SyntaxErrorCode <> 0 then goto endp;

 New(H);
 Tree^.Ch:=H;
 WriteNodeParam(ND);
 WriteLevelParam(H,HS,0);

 H^.Stack.S:=Tree^.Stack.S;
 H^.Pr:=Tree;
 H1:=H;
 H^.Ch:=nil;
 H^.Nb:=nil;
 for i:=1 to High(HS) do
 begin
  New(H);
  H^.Ch:=nil;
  H^.Nb:=nil;
  WriteLevelParam(H,HS,i);

  H^.Stack.S:=H1^.Stack.S+1;
  H1.Nb:=H;
  H^.Pr:=Tree;
  H1:=H;
 end;
 Tree:=H^.Pr^.Ch;
 HS:=nil; //??? (звільнення масиву рівня)
 SyntaxAnalizator(Copy(Tree^.Expression,1,Length(Tree^.Expression)),ND,HS,EOFN);
 end;
//EOFN=True:остання команда вузла
WriteNodeParam(ND);
end;
Tree:=Tree^.Pr;
end;

Tree:=BeginTree;

endp:
end;

procedure TForeval.FreeFunction;
var
i: Integer;
begin
```

```
if F_SyntaxErrorCode = 0 then
for i:=1 to Length(Func.Interp)-1 do
begin
dispose(Func.Interp[i]);
end;
Func.Comp:=nil;
Func.Interp:=nil;

end;

procedure TForeval.WriteNodeParam(ND: TDataNode);
begin
Tree^.Stack.N:=ND.N;
Tree^.NCOP:=ND.NCOP;
Tree^.FD:=ND.FD;
Tree^.NFunc:=ND.NFunc;
Tree^.COP:=ND.COP;
Tree^.NObj:=ND.NObj;
Tree^.Id:=ND.Id;
end;

procedure TForeval.WriteLevelParam(H: PNode; HS: PLevel; N: Cardinal);
begin
H^.Expression:=Copy(HS[N].Expression,1,Length(HS[N].Expression));
H^.NL:=HS[N].NL;
H^.Id:=HS[N].Id;
end;

procedure TForeval.FindExternalBracket(S: String; var SS: String);
label 1,endp;
var
b,i,z,a:Integer;
begin
SS:=Copy(S,1,Length(S));



1:
if SS = '' then
begin
goto endp;
end;

if (SS[1] = '(') and (SS[Length(SS)] = ')') then
begin
  b:=-1;
  for i:=2 to Length(SS)-1 do
  begin
   if SS[i] = '(' then b:= b-1;
   if SS[i] = ')' then b:=b+1;
   if b = 0 then goto endp;
  end;

 Delete(SS,Length(SS),1);
 Delete(SS,1,1);
 goto 1;
end;

endp:
end;
function TForeval.FindNumberPR(PR: Cardinal): Cardinal;
label 1;
var
i: Integer;
N: Cardinal;
begin
for i:=0 to High(Alg.Opr) do
begin
```

```
 if PR = Alg.Opr[i].PR then
 begin
  N:=i; goto 1;
 end;
end;

1:
FindNumberPR:=N;
end;

function TForeval.FindOperation(S: String): Cardinal;
label 1;
var
i,NCOP: Integer;
begin
NCOP:=-1;
for i:=0 to High(Alg.Opr) do
begin
    if S = Alg.Opr[i].Name then
    begin
     NCOP:=i;
     goto 1;
    end;
end;

1:
FindOperation:=NCOP;
end;

procedure TForeval.FindSingleOP(S: String; var NCop: Integer);
var
i,j,PR1,PR2: Integer;
NCop1,NCop2: Integer;
begin
PR1:=0;
PR2:=0;
NCop1:=-1; NCop2:=-1;  NCop:=-1;

for i:=0 to High(Alg.Opr) do
begin
 if Alg.Opr[i].GS < 0 then    //тільки одноарг. операції
 begin
    if (S[1] = Alg.Opr[i].Name[1]) and (Alg.Opr[i].GS = -1) then
    begin
     NCop1:=i;
    end;

    if (S[Length(S)] = Alg.Opr[i].Name[1]) and  (Alg.Opr[i].GS = -2) then
    begin
     NCop2:=i;
    end;
  end;
end;


if (NCop1 <> -1) and (NCop2 <> -1) then
begin
 if  Alg.Opr[NCop1].PR < Alg.Opr[NCop2].PR then NCop:=NCop1 else NCop:=NCop2;
end
else
if NCop1 <> -1 then NCop:=NCop1
else
if NCop2 <> -1 then NCop:=NCop2;

end;

procedure TForeval.FindMainOperation(S:String; var NCOP: Integer; var NPos:
Cardinal);
label endp;
```

```
var
Pl,Ml: Boolean;
i,b: Integer;
NCOP1,NCOP2,NPos1: Integer;
begin
if S = '' then
begin
F_SyntaxErrorCode:=5;
goto endp;
end;

NCOP1:=-1; NCOP2:=-1;  NCOP:=-1;
b:=0; Pl:=False; Ml:=False;

if S[1] = '(' then b:=-1
 else
  if S[1] = ')' then
   begin
   F_SyntaxErrorCode:=2;
   goto endp;
   end;


for i:=2 to Length(S)-1 {тому що 1-ий і останній символи м.б. одноарг.
символами}  do
begin
 if S[i] = '(' then b:= b-1;
 if S[i] = ')' then b:=b+1;

 if b = 0 then
 begin
  NCOP2:=FindOperation(S[i]);
  if Alg.Opr[NCOP2].GS >= 0 then //тільки багатоарг. операції
  begin
   if (NCOP2 <> -1) and (NCOP1 <> -1) then
   begin
    {обов'язково <=, щоб взяти останній доданок !!!}
    if (Alg.Opr[NCOP2].PR <= Alg.Opr[NCOP1].PR)  then
    begin
     NCOP1:=NCOP2;
     NPos1:=i;
    end;
   end
   else
   if (NCOP2 <> -1) and (NCOP1 = -1) then
   begin
    NCOP1:=NCOP2;
    NPos1:=i;
   end;
  end;
 end;

NPos:=NPos1;
NCOP:=NCOP1;
end;


//корекція виражень на одноарг. оп. типу -x^y,    (neg)
//FindSingleOP(S,NCop,NPos1);
NCop1:=FindOperation(S[1]);
if (NCOP1 <> -1) and (Alg.Opr[NCOP1].PR < Alg.Opr[NCOP].PR)  then NCOP:=-1;

endp:
end;

procedure TForeval.StringAnalizator(S:String; var ND: TDataNode; var HS:PLevel);
label endp;
var
SS,S1: String;
```

```
PR,NPR: Cardinal;
BH,IH,IH1: PLevel;
i,p,b: Cardinal;
PR1,N,NCop: Integer;
ni: Integer;
NPos: Cardinal;
begin
if F_SyntaxErrorCode <> 0 then goto endp;

FindExternalBracket(S,SS);

InitNode(ND);
PR:=0;
FindMainOperation(SS,NCop,NPos);
N:=0;
if NCop <> -1 then
begin
 inc(F_Id);
 ND.Id:=F_Id;
 SetLength(HS,2);
 HS[0].Expression:=Copy(SS,1, NPos-1);
 HS[0].NL:=1;
 HS[0].Id:=F_Id;
 HS[1].Expression:=Copy(SS,NPos+1,Length(SS)-NPos+1);
 HS[1].NL:=2;
 HS[1].Id:=F_Id;
 ND.N:=2;
 ND.COP:=_COP;
 ND.NFunc:=Alg.Opr[NCop].NFunc;
 ND.NCOP:=NCOP;  //для помилок
end
else
begin
 FindSingleOp(SS,NCop);
 if NCop <> -1 then
 begin
  ND.COP:=_FUNC;
  ND.N:=1;
  ND.NFunc:=Alg.Opr[NCop].NFunc;
  SetLength(HS,1);
  if Alg.Opr[NCop].GS = -1 then HS[0].Expression:=Copy(SS,2,Length(SS)-1)
  else
  if Alg.Opr[NCop].GS = -2 then HS[0].Expression:=Copy(SS,1,Length(SS)-1);
  HS[0].NL:=1;
  ND.NCOP:=NCOP;  //для помилок
  inc(F_Id);
  ND.Id:=F_Id;
  HS[0].Id:=F_Id;
 end
 else
 begin
  ND.COP:=0;
  SetLength(HS,1);
  ND.N:=1;
  HS[0].NL:=1;
  HS[0].Expression:=Copy(SS,1,Length(SS));
 end;
end;

endp:
end;

procedure TForeval.SubstParam(S1: String;  var C: TFloatType; var SB: Integer);
label endp;
var
N,i: Cardinal;
S1h,S3: String;
begin
if F_SyntaxErrorCode <> 0 then goto endp;
```

```
C:=0;
if S1 = '' then
begin
F_SyntaxErrorCode:=5;
goto endp;
end;
N:=1; S1h:=Copy(S1,1,Length(S1));  SB:=0;

try
C:=StrToFloat(PointToDec(S1h));
SB:=1;
except
SB:=0;
end;

endp:
end;

procedure TForeval.NumberAnalizator(S: String;  var ND: TDataNode; var EOFN:
Boolean);
label 1;
var
S2: String;
ni,SB: Integer;
C: TFloatType;
begin
EOFN:=False;
InitNode(ND);
FindVar(S,ni);
if ni >= 0 then
 begin
  ND.COP:=_LDSF;
  ND.NObj:=ni;
  EOFN:=True;
  goto 1;
 end;

SubstParam(S,C,SB);
if SB = 1 then
begin
ND.FD:=C;
ND.COP:=_LDSC;
EOFN:=True;
inc(F_Id);
ND.Id:=F_Id;
goto 1;
end;

F_SyntaxErrorCode:=1;         //невідомий символ
F_SyntaxErrorString:=Copy(S,1,Length(S));
EOFN:=False;

1:
end;

procedure TForeval.FindVar(S1: String; var ni: Integer);
label endp;
var
i: Cardinal;
S2,S3: String;
begin                                          //ni - номер масиву знайденої
змінної
if F_SyntaxErrorCode <> 0 then goto endp;      //ni = -1: змінних не знайдена
ni:=-1;

S2:=Copy(S1,1,Length(S1));
if F_VarShift = False then S2:=LowerCase(S2);

for i:=0 to High(Alg.Obj) do
```

```
begin
S3:=Alg.Obj[i].Name;
if F_VarShift = False then S3:=LowerCase(S3);
 if S2 = S3 then
  begin
   ni:=i;
   goto endp;
  end;
end;

endp:
end;

procedure TForeval.FindFunction(S: String;  var ND: TDataNode; var LV: PLevel;
var BH: Boolean);
label 1,2,3,4,endp;
var
i,k,l,bb,pf,pr,b1,b2,nf: Integer;
S1,Sa,Sf,Sf1,Sfn: String;
Neg: Integer;
begin
if F_SyntaxErrorCode <> 0 then goto endp;
BH:=False;



//пошук функції:
nf:=0; bb:=0;  pf:=0; b1:=0;
for i:=1 to Length(S) do
begin
 if S[i] = '(' then begin b1:=i; goto 1; end;
end;
if b1 = 0 then goto endp;

1:
b2:=0; bb:=-1;
for i:=b1+1 to Length(S) do
begin
 if S[i] = '(' then bb:= bb-1;
 if S[i] = ')' then bb:=bb+1;
 if (i <> Length(S)) and (bb = 0) then begin {Internal Error} end;
end;


Sf:=Copy(S,1,b 1-1);
Sa:=Copy(S,b1+1,Length(S)-b 1-1);


if F_FuncShift = False then Sf1:=LowerCase(Sf) else Sf1:=Sf;

nf:=-1;
for i:=0 to High(Alg.ExFunc) do
begin
  Sfn:=Alg.ExFunc[i].Name;
  if F_FuncShift = False then
  begin
   Sfn:=LowerCase(Sfn);
  end;
  if Sf1 = Sfn then  begin nf:=i;  goto 4; end;
end;


F_SyntaxErrorCode:=6; //UknownFunction
F_SyntaxErrorString:=Copy(Sf,1,Length(Sf));
goto endp;


4:
 if nf <> -1 then
```

```
 begin
    FindManyArg(Sa,Arg);
    if Length(Arg) <= 1 then
    begin
     { ф-ія без аргументів}
     F_SyntaxErrorCode:=4;
     F_SyntaxErrorString:=Copy(Sf,1,Length(Sf));
     goto endp;
    end;

    inc(F_Id);
    ND.Id:=F_Id;

    LV[0].Expression:=Copy(Arg[1],1,Length(Arg[1]));
    LV[0].NL:=1;
    LV[0].Id:=F_Id;
    for i:= 2 to High(Arg) do
    begin
     SetLength(LV,Length(LV)+1);
     LV[ i-1].Expression:=Copy(Arg[i],1,Length(Arg[i]));
     LV[ i-1].NL:=i;
     LV[ i-1].Id:=F_Id;
    end;

    ND.N:=High(Arg);
    ND.COP:=_Func;
    ND.NFunc:=nf;
    BH:=True;
 end;


endp:
end;



procedure TForeval.FindManyArg(S: String; var Arg: TArray);
label endp;
var
i,N,bb: Cardinal;
PP: array of Cardinal;
begin

bb:=0;
N:=1; SetLength(PP,N+1); PP[1]:=0;

for i:=1 to Length(S) do
begin
 if S[i] = '(' then bb:= bb-1;
 if (bb = 0) and (S[i] = ',') then begin inc(N); SetLength(PP,N+1); PP[N]:=i;
end;
 if S[i] = ')' then bb:=bb+1;
end;

inc(N);  SetLength(PP,N+1); PP[N]:=Length(S)+1;

SetLength(Arg,N);

for i:=1 to N-1 do
begin
Arg[i]:=Copy(S,PP[i]+1,PP[i+1]-PP[i]-1);
if (Arg[i] = '')  then
begin
F_SyntaxErrorCode:=4; F_SyntaxErrorString:=Copy(S,1,Length(S)); goto endp;
end;
end;

endp:
```

```
end;

procedure TForeval.XchangeAddingBracket(S: String; var S1: String);
var
i: Cardinal;
begin
S1:=Copy(S,1,Length(S));
for i:=1 to Length(S1) do
 begin
  if (S1[i] = '[') or  (S1[i] = '{') or  (S1[i] = '<') then
   begin Delete(S1,i,1); Insert('(',S1,i); end;
  if (S1[i] = ']') or  (S1[i] = '}') or  (S1[i] = '>') then
   begin Delete(S1,i,1); Insert(')',S1,i); end;
 end;
end;

procedure TForeval.InsertBracketOperation(S: String; var S1: String);
label 1,2,3,4,5,endp;
var
i,b,b1,k,j: Integer;
BH: Boolean;
begin
S1:=Copy(S,1,Length(S));
k:=0;

//круглі дужки для операцій не використовуються !!!
(*
i:=1;
while i <= Length(S1) do
 begin

  if (S1[i] = '(') and (Alg.Bracket.Br1 = 1) then
  begin
   b:=0;
   b1:=-1;
   BH:=False;
   for j:=i+1 to Length(S1) do
   begin
    if (S1[j] = '(') or (S1[j] = '{') or (S1[j] = '[') or (S1[j] = '<') then b:=
b-1;
    if (S1[j] = ')') or (S1[j] = '}') or (S1[j] = ']') or (S1[j] = '>')  then
b:=b+1;
    if S1[j] = '(' then b1:=b 1-1;
    if S1[j] = ')' then b1:=b1+1;
    if (b = 0) and (S1[j] = ',') then BH:=True;
    if b1 = 0 then begin k:=j; goto 1; end;
   end;

   F_SyntaxErrorCode:=2;
   F_SyntaxErrorString:=')';
   goto endp;

   1:
   if BH = True then
   begin
    Delete(S1,k,1);
    Insert(')',S1,k);
    Delete(S1,i,1);
    Insert(Alg.ExFunc[Alg.Bracket.NF1].Name+'(',S1,i);
    i:=i+Length(Alg.ExFunc[Alg.Bracket.NF1].Name)+1;
   end
   else inc(i);
  end
  else inc(i);

end;
*)

5:
```

```
for i:=1 to Length(S1) do
 begin
  if (S1[i] = '[') and (Alg.Bracket.Br2 = 1) then
  begin
   b:=0;
   b1:=-1;
   BH:=False;
   for j:=i+1 to Length(S1) do
   begin
    if (S1[j] = '(') or (S1[j] = '{') or (S1[j] = '[') or (S1[j] = '<') then b:=
b-1;
    if (S1[j] = ')') or (S1[j] = '}') or (S1[j] = ']') or (S1[j] = '>')  then
b:=b+1;
    if S1[j] = '[' then b1:=b 1-1;
    if S1[j] = ']' then b1:=b1+1;
    //if (b = 0) and (S1[j] = ',') then BH:=True;   {для будь-якого числа
змінних}
    if b1 = 0 then begin k:=j; goto 2; end;
   end;

   F_SyntaxErrorCode:=2;
   F_SyntaxErrorString:=']';
   goto endp;

   2:
   //if BH = True then   {для будь-якого числа змінних}
   begin
    Delete(S1,k,1);
    Insert(')',S1,k);
    Delete(S1,i,1);
    Insert(Alg.ExFunc[Alg.Bracket.NF2].Name+'(',S1,i);
    goto 5;
   end;
  end;


  if (S1[i] = '{') and (Alg.Bracket.Br3 = 1) then
  begin
   b:=0;
   b1:=-1;
   BH:=False;
   for j:=i+1 to Length(S1) do
   begin
    if (S1[j] = '(') or (S1[j] = '{') or (S1[j] = '[') or (S1[j] = '<') then b:=
b-1;
    if (S1[j] = ')') or (S1[j] = '}') or (S1[j] = ']') or (S1[j] = '>')  then
b:=b+1;
    if S1[j] = '{' then b1:=b 1-1;
    if S1[j] = '}' then b1:=b1+1;
    //if (b = 0) and (S1[j] = ',') then BH:=True;
    if b1 = 0 then begin k:=j; goto 3; end;
   end;

   F_SyntaxErrorCode:=2;
   F_SyntaxErrorString:='}';
   goto endp;

   3:
   //if BH = True then
   begin
    Delete(S1,k,1);
    Insert(')',S1,k);
    Delete(S1,i,1);
    Insert(Alg.ExFunc[Alg.Bracket.NF3].Name+'(',S1,i);
    goto 5;
   end;
  end;

  if (S1[i] = '<') and (Alg.Bracket.Br4 = 1) then
```

```
  begin
   b:=0;
   b1:=-1;
   BH:=False;
   for j:=i+1 to Length(S1) do
   begin
    if (S1[j] = '(') or (S1[j] = '{') or (S1[j] = '[') or (S1[j] = '<') then b:=
b-1;
    if (S1[j] = ')') or (S1[j] = '}') or (S1[j] = ']') or (S1[j] = '>')  then
b:=b+1;
    if S1[j] = '<' then b1:=b 1-1;
    if S1[j] = '>' then b1:=b1+1;
    //if (b = 0) and (S1[j] = ',') then BH:=True;
    if b1 = 0 then begin k:=j; goto 4; end;
   end;

   F_SyntaxErrorCode:=2;
   F_SyntaxErrorString:='>';
   goto endp;

   4:
   //if BH = True then
   begin
    Delete(S1,k,1);
    Insert(')',S1,k);
    Delete(S1,i,1);
    Insert(Alg.ExFunc[Alg.Bracket.NF4].Name+'(',S1,i);
    goto 5;
   end;
  end;

end;

endp:
end;


procedure TForeval.PrevTreat(S: String; var S1: String);
label endp;
var
b,i: Integer;
begin
if S = '' then S:='0';
//S:=LowerCase(S);

InsertBracketOperation(S,S);

if F_SyntaxErrorCode <> 0 then goto endp;

//додаткові дужки можуть використовуватися тільки в операціях !!!
{if F_SyntaxExtension = True then
begin
 XchangeAddingBracket(S,S);
end;
}
F_SyntaxErrorCode:=0;
b:=0;
for i:=1 to Length(S)-1 do
begin
 if S[i] = '(' then b:= b-1;
 if S[i] = ')' then b:=b+1;
end;

if S[Length(S)] = ')' then b:=b+1;
if b <> 0 then
begin            // внутрішня не відповідність відкр. і закр. дужок
F_SyntaxErrorCode:=2;
if b > 0 then F_SyntaxErrorString:='('
else
```

```
if b < 0 then F_SyntaxErrorString:=')'
end;

//L1.Caption:=S[3];
S1:=Copy(S,1,Length(S));
endp:
end;

procedure TForeval.SyntaxAnalizator(S:String; var ND: TDataNode; var LV:PLevel;
var EOFN: Boolean);
label endp;
var
COP,Arg1,Arg2: String;
nf: Cardinal;
BH: Boolean;
a,b: TFloatType;
begin
if F_SyntaxErrorCode <> 0 then goto endp;

EOFN:=False;
InitNode(ND);

StringAnalizator(Copy(S,1,Length(S)),ND,LV);
If ND.COP = 0 then
 begin
  FindFunction(LV[0].Expression,ND,LV,BH);
  if BH = True then
  begin
   EOFN:=False;
  end
  else
  NumberAnalizator(LV[0].Expression,ND,EOFN);
 end;

endp:
end;

procedure TForeval.CodeGeneration;
label endp;
var
E: Exception;
begin
if F_SyntaxErrorCode <> 0 then goto endp;

LFunc:=1;
SetLength(Func.Interp,LFunc);
new(Func.Interp[0]);
Tree:=BeginTree; //не обов'язково, тому що читання дерева - циклічно

while Tree^.Ch <> nil do
begin
Tree:=Tree^.Ch;
end;
//COP=NOP:
WriteCode;

while Tree^.Pr <> nil do
begin
 while Tree^.Nb <> nil  do
 begin
 Tree:=Tree^.Nb;
  while Tree^.Ch <> nil do
  begin
  Tree:=Tree^.Ch;
  end;
//COP=NOP:
WriteCode;
end;
```

```
//COP<>NOP:
Tree:=Tree^.Pr;
WriteCode;
end;
// 1-ий елемент в WriteCode випадає

endp:

if F_SyntaxErrorCode <> 0 then
 begin
  CalcException(Foreval as TObject,E);
 end
end;

function TForeval.Calc: TFloatType; assembler;
asm
 call [eax+Func.ICode];
end;

procedure TForeval.WriteCode;
label endp;
var
i: Cardinal;
begin
SetLength(Func.Interp,LFunc+1);
new(Func.Interp[LFunc]);
WriteDataCode;

endp:
{-----}
//out:
{$IFDEF TEXTOUT}
if Test8.Form1.CB_Cod.Checked = True then
begin
if Tree^.COP = _LDSC then
begin
 Test8.Form1.M2.Lines[ LFunc-1]:='LDSC:
'+String(StrUpper(PChar(FloatToStr(Tree^.FD))));
end
else
if Tree^.COP = _LDSF then
begin
 Test8.Form1.M2.Lines[ LFunc-1]:='LDSF:
'+String(StrUpper(PChar(Alg.Obj[Tree^.NObj].Name)));
end
else
if (Tree^.COP = _Func) or (Tree^.COP = _COP) then
begin
 Test8.Form1.M2.Lines[ LFunc-
1]:=String(StrUpper(PChar(Alg.ExFunc[Tree^.NFunc].Name)));
end;




end;
{$ENDIF}
{-----}
inc(LFunc);

end;



procedure TForeval.WriteDataCode;
label endp;
var
fa,i,L,j: Cardinal;
S: String;
```

```
nf: Integer;
ANL: TArray;
begin
if F_SyntaxErrorCode <> 0 then goto endp;
if Tree^.COP = _LDSC then
begin
 FindConstLdFunction(nf);
 if nf = -1 then
 begin {Error: немає ф-ії завантаження типу}
  F_SyntaxErrorCode:=7;
  F_SyntaxErrorString:='1';
 end;
 Func.Interp[LFunc]^.FD:=Tree^.FD;
 Func.Interp[LFunc]^.NF:=NF;
 Func.Interp[LFunc]^.COP:=_LDSC;
 Func.Interp[LFunc]^.Type:=1;
end
else
if Tree^.COP = _LDSF then
begin
 FindLdFunction(nf);
 if nf = -1 then
 begin {Error: немає ф-ії завантаження типу}
  F_SyntaxErrorCode:=7;
  F_SyntaxErrorString:=IntToStr(Alg.Obj[Tree^.NObj].Type);
 end;
 Func.Interp[LFunc]^.NObj:=Tree^.NObj;
 Func.Interp[LFunc]^.NF:=nf;
 Func.Interp[LFunc]^.Type:=Alg.Obj[Tree^.NObj].Type;
 Func.Interp[LFunc]^.COP:=_LDSF;
end
else
if (Tree^.COP = _FUNC) or (Tree^.COP = _COP) then
begin
 FindCallFunction(nf);
 if nf = -1 then
 begin {Error: немає ф-ії}
  //для операцій
  if Tree^.COP = _COP then
  begin
   F_SyntaxErrorCode:=8;
   F_SyntaxErrorString1:=Alg.Opr[Tree^.NCOP].Name;
  end
  else
  //для ф-ій
  begin
   F_SyntaxErrorCode:=9;
   F_SyntaxErrorString1:=Alg.ExFunc[Tree^.NFunc].Name;
  end;

  F_SyntaxErrorString:='';
  FindNumberLevel(ANL);
  for j:=0 to Tree^.Stack.N-1 do
  begin
   if j > 0 then  F_SyntaxErrorString:=F_SyntaxErrorString+',';
   F_SyntaxErrorString:=F_SyntaxErrorString+IntToStr(Func.Interp[ANL[j]].Type);
  end;

 end;

 Func.Interp[LFunc]^.COP:=_FUNC;
 Func.Interp[LFunc]^.NFunc:=Tree^.NFunc;
 Func.Interp[LFunc]^.Arg:=Tree^.Stack.N;
 Func.Interp[LFunc]^.NF:=nf;
 Func.Interp[LFunc]^.Type:=Alg.ExFunc[Tree^.NFunc].Func[nf].Type;
end;

Func.Interp[LFunc]^.S:=Tree^.Stack.S-1;  //починається з '0'
Func.Interp[LFunc]^.N:=Tree^.Stack.N;
```

```
Func.Interp[LFunc]^.NL:=Tree^.NL;
Func.Interp[LFunc]^.Id:=Tree^.Id;
Func.Interp[LFunc]^.Id:=Tree^.Id;

endp:
end;

procedure TForeval.FindCallFunction(var nf: Integer);
label 1,2,endp;
var
Fnc: TIntFunc;
i,N,T,j,S: Integer;
ANL: TArray;
begin
nf:=-1;
i:=Tree^.NFunc;
N:=Tree^.Stack.N;
S:=Tree^.Stack.S;
Fnc:=Alg.ExFunc[Tree^.NFunc].Func;

FindNumberLevel(ANL);

//спочатку шукати серед Infinite
if (F_ExtNumberArg > 0)  and (Length(ANL) >= F_ExtNumberArg) then
begin

 for i:=0 to High(Fnc) do
 begin

  if  Fnc[i].N = -1 then //перевірити збіг типів:
  begin

   for j:=0 to High(ANL) do
   begin
    if Func.Interp[ANL[j]].Type <> Fnc[i].TV[0] then goto 2;
   end;

   nf:=i;
   goto endp;
  end;

 2:
 end;

end;

for i:=0 to High(Fnc) do
begin

 if N = Fnc[i].N then //перевірити збіг типів:
 begin

  for j:=0 to N-1 do
  begin
   if Func.Interp[ANL[j]].Type <> Fnc[i].TV[j] then goto 1;
  end;

  nf:=i;
  goto endp;
 end;

1:
end;


endp:
end;

procedure TForeval.FindNumberLevel(var ANL: TArray);
```

```
var   //визначення номерів рівнів, що випливають із даного вузла
i,j: Integer;
E: Exception;
begin

for j:=1 to LFunc-1 do
begin
   if Tree^.Id = Func.Interp[j]^.Id then
   begin
    SetLength(ANL,Length(ANL)+1);
    ANL[High(ANL)]:=j;
   end;
end;

if Tree^.Stack.N <> Length(ANL) then
begin {InternalError}
 F_InternalError:=1;
 CalcException(Foreval as TObject,E);
end;

end;

procedure TForeval.FindLdFunction(var nf: Integer);
label endp;
var
i: Integer;
begin
nf:=-1;
  for i:=0 to High(Alg.LdFunc) do
  begin
   if Alg.Obj[Tree^.NObj].Type = Alg.LdFunc[i].Type then
   begin
    nf:=i;
    goto endp;
   end;
  end;

endp:
end;

procedure TForeval.FindConstLdFunction(var nf: Integer);
label endp;
var
i: Integer;
begin
nf:=-1;
  for i:=0 to High(Alg.LdFunc) do
  begin
   if Alg.LdFunc[i].Type = 1 then
   begin
    nf:=i;
    goto endp;
   end;
  end;

endp:
end;

procedure TForeval.InitNode(var ND: TDataNode);
var
i: Integer;
begin
ND.N:=0;
ND.NCOP:=0;
ND.NFunc:=0;
ND.NObj:=0;
ND.COP:=0;
ND.FD:=0;
ND.FX:=0;
```

```
ND.Id:=0;
end;

function TForeval.PointToDec(S: String):String;
label
1;
var
i:Cardinal;
SS: String;
begin
SS:=Copy(S,1,Length(S));
for i:=1 to Length(S) do
 begin
  if SS[i] = '.' then
  begin
   Delete(SS,i,1);
   Insert(',',SS,i);
   goto 1;
 end
end;

1:
PointToDec:=Copy(SS,1,Length(SS));
end;

procedure TForeval.CreateInitData;
begin                              //порядок не міняти

F_SyntaxErrorCode:=0;
LFunc:=0;

F_ShowException:=False;
F_ExternalException:=False;
F_VarShift:=False;
F_FuncShift:=False;
Alg.Bracket.Br1:=0;
Alg.Bracket.Br2:=0;
Alg.Bracket.Br3:=0;
Alg.Bracket.Br4:=0;

Application.OnException:=CalcException;

SetType(1);
end;

procedure TForeval.SetExtExpression(S: String;  var E_Func: TFunction; var
Stack: Cardinal);
var
I_Func: TFunction;
I_LFunc: Cardinal;
E: Exception;
begin
try
I_LFunc:=LFunc;
I_Func:=Func;
Func:=E_Func;
F_SyntaxErrorCode:=0;
F_CalcErrorCode:=0;
F_InternalError:=0;
F_SyntaxErrorString:='';
F_Id:=0;
CurrentExpression:=Copy(S,1,Length(S));
PrevTreat(S,S);
Linker(S);
FreeFunction;
CodeGeneration;
if Length(Func.Interp) <> 0 then
F_ResultType:=Func.Interp[High(Func.Interp)]^.Type;
SetStackSize;
```

```
Compile;
Func.ICode:=TAddress(@Func.Comp[0]);
FreeTree;
Stack:=StackSize;
E_Func:=Func;
LFunc:=I_LFunc;
Func:=I_Func;

except
 on E: EZeroDivide      do F_InternalError:=1;
 on E: EInvalidOp       do F_InternalError:=2;
 on E: EOverFlow        do F_InternalError:=3;
 on E: EUnderFlow       do F_InternalError:=4;
 on E: EIntOverFlow     do F_InternalError:=5;
 on E: EAccessViolation do F_InternalError:=6;
 on E: EOutOfMemory     do F_InternalError:=7;
 on E: EStackOverFlow   do F_InternalError:=8;
end;
if F_InternalError <> 0 then CalcException(Foreval as TObject,E);
//BeginLinker:=1;
end;


procedure TForeval.CalcExtFunc(E_Func: TFunction); assembler;
asm
 //call [E_Func.ICode]
end;

function TForeval.CalcExtFunc(Addr: TAddress): TFloatType; assembler;
asm
 call  Addr
 //fstp  @Result
end;

procedure TForeval.FreeExtFunc(E_Func: TFunction);
var
i: Cardinal;
begin

for i:=1 to Length(E_Func.Interp)-1 do
begin
dispose(E_Func.Interp[i]);
end;
E_Func.Comp:=nil;
E_Func.Interp:=nil;

end;

constructor TForeval.Create;
begin
//inherited Create(AOwner);
CreateInitData;
end;

destructor TForeval.Destroy;
begin
FreeFunction;
inherited Destroy;
end;

procedure TForeval.FreeTree;
label endp;
var
H: PNode;
begin
if F_SyntaxErrorCode <> 0 then goto endp;
Tree:=BeginTree;

while Tree^.Ch <> nil do
```

```
begin
Tree:=Tree^.Ch;
end;
//COP=NOP

while Tree^.Pr <> nil do
begin
 while Tree^.Nb <> nil  do
 begin
  H:=Tree; Tree:=Tree^.Nb;  H^.Nb:=nil;
  dispose(H);
  while Tree^.Ch <> nil do
  begin
  Tree:=Tree^.Ch;
  end;
 //COP=NOP
 end;

//COP<>NOP
H:=Tree; Tree:=Tree^.Pr; H^.Pr:=nil;
dispose(H);
end;
dispose(Tree);

endp:
end;

procedure TForeval.CalcException(Sender: TObject; E: Exception);
var
S: String;
T: Integer;
begin

if F_BeginCalc = 0 then
begin
 {if E.ClassType  = EZeroDivide       then F_InternalError:=1
 else
 if E.ClassType =  EInvalidOp        then F_InternalError:=2
 else
 if E.ClassType =  EOverFlow         then F_InternalError:=3
 else
 if E.ClassType =  EUnderFlow        then F_InternalError:=4
 else
 if E.ClassType =  EIntOverFlow      then F_InternalError:=5
 else
 if E.ClassType =  EAccessViolation  then F_InternalError:=6
 else
 if E.ClassType =  EOutOfMemory      then F_InternalError:=7
 else
 if E.ClassType =  EStackOverFlow    then F_InternalError:=8;
 }
end

else
if F_SyntaxErrorCode = 0 then
begin
 if E.ClassType  = EZeroDivide       then F_CalcErrorCode:=1
 else
 if E.ClassType =  EInvalidOp        then F_CalcErrorCode:=2
 else
 if E.ClassType =  EOverFlow         then F_CalcErrorCode:=3
 else
 if E.ClassType =  EUnderFlow        then F_CalcErrorCode:=4
 else
 if E.ClassType =  EIntOverFlow      then F_CalcErrorCode:=5
 else
 if E.ClassType =  EAccessViolation  then F_CalcErrorCode:=6;
 F_BeginCalc:=0;
end;
```

```
if F_ShowException then
begin
 MessageBeep(MB_ICONHAND);
 if F_CalcErrorCode <> 0 then
 begin
  MessageDlg('Calculation Error',mtError,[mbOk],0)
 end
 else
 if F_SyntaxErrorCode <> 0 then
 begin
  if F_SyntaxErrorCode = 1 then S:='UNKNOWN SYMBOL:';
  if F_SyntaxErrorCode = 2 then S:='MISSING BRACKET:';
  if F_SyntaxErrorCode = 3 then S:='MISSING OPERATION:';
  if F_SyntaxErrorCode = 4 then S:='INCOINCIDENCE NUMBER OF ARGUMENTS:';
  if F_SyntaxErrorCode = 5 then S:='MISSING EXPRESSION:';
  if F_SyntaxErrorCode = 6 then S:='UNKNOWN FUNCTION:';
  if F_SyntaxErrorCode = 7 then S:='ABSENT FUNCTION FOR LOAD TYPE:';
  if F_SyntaxErrorCode = 8 then S:='NOT DEFINED OPERATION
'+'"'+F_SyntaxErrorString1+'"'+' FOR TYPES:';
  if F_SyntaxErrorCode = 9 then S:='NOT DEFINED FUNCTION
'+'"'+F_SyntaxErrorString1+'"'+' FOR TYPES:';

  MessageDlg(S+#13+#10+'"'+F_SyntaxErrorString+'"',mtError,[mbOk],0)
 end
 else
 if F_InternalError <> 0  then
 MessageDlg('Internal Error',mtError,[mbOk],0)
end;

{
if F_ExternalException then
begin
 if F_SyntaxErrorCode <> 0 then T:=1000
 else
 if F_CalcErrorCode <> 0 then T:=1001
 else
 if F_InternalError <> 0 then T:=1002;

 ExtException(T);
end;
}

end;

procedure TForeval.Compile;
label 1;
var
i,j,N: Integer;
AH,AB: TArray4;
R: TFloatType;
P: Pointer;
X: TFloatType;
begin

LFunc:=Length(Func.Interp);

SetLength(ICOMP,0);

PUSHA;
for i:=1 to LFunc-1 do
begin
 AddCommand(i);
end;
POPA;
RET;

SetLength(Func.Comp,Length(ICOMP));
Func.Comp:=Copy(ICOMP,0,Length(ICOMP));
```

```
ICOMP:=nil;
end;

procedure TForeval.RET;
begin
AddByte($C3);
end;

procedure TForeval.AddCommand(I: Integer);
var
N,NF: Integer;
begin
//стек:
MOVMA(TAddress(@Func.Interp[I]^.S));
MOVAM(TAddress(@F_NumberStack));

if Func.Interp[I]^.COP = _LDSC then
begin
 MOVA1(TAddress(@Func.Interp[I]^.FD));
 MOVAM(TAddress(@F_Address));
 N:=Func.Interp[I]^.NF;
 MOVA1(TAddress(Alg.LdFunc[N].Addr));
 CALLA;
end
else
if Func.Interp[I]^.COP = _LDSF then
begin
 N:=Func.Interp[I]^.NObj;
 MOVA1(Alg.Obj[N].Addr);
 MOVAM(TAddress(@F_Address));
 N:=Func.Interp[I]^.NF;
 MOVA1(TAddress(Alg.LdFunc[N].Addr));
 CALLA;
end
else
if Func.Interp[I]^.COP = _FUNC then
begin
 N:=Func.Interp[I]^.NFunc;
 NF:=Func.Interp[I]^.NF;

 //infinite:
 if Alg.ExFunc[N].Func[NF].N = -1 then
 begin
  MOVA1(TAddress(Func.Interp[I]^.Arg));
  MOVAM(TAddress(@F_NumberArg));
 end;

 MOVA1(Alg.ExFunc[N].Func[NF].Addr);
 CALLA;
end;

end;

procedure TForeval.IntToHex(N: TAddress; var AH: TArray4);
var
i,K,N1,j: Integer;
A8: array [1..8] of Integer;
begin
N1:=N;  j:=1;
for i:=7 downto 1 do
begin
 K:=Trunc(N1/IntPower(2,i*4));
 A8[j]:=(K);
 N1:=N 1-Trunc(IntPower(2,i*4))*K;
 inc(j);
end;
A8[j]:=N1;

//у зворотному порядку:
```

```
j:=1;
for i:=4 downto 1 do
begin
 AH[i]:=A8[j]*16+A8[j+1];
 j:=j+2;
end;
end;

procedure TForeval.AddByte(B: Byte);
var
j: Integer;
begin
j:=High(ICOMP)+1;
SetLength(ICOMP,Length(ICOMP)+1);
ICOMP[j]:=B;
end;

procedure TForeval.AddWord(B1: Byte; B2: Byte);
var
j: Integer;
begin
j:=High(ICOMP)+1;
SetLength(ICOMP,Length(ICOMP)+2);
ICOMP[j]:=B1; ICOMP[j+1]:=B2;
end;

procedure TForeval.AddAddress(Adr: TAddress);
var
AB: TArray4;
j: Integer;
begin
IntToHex(Adr,AB);
j:=High(ICOMP)+1;
SetLength(ICOMP,Length(ICOMP)+4);
ICOMP[j]:=AB[1];  ICOMP[j+1]:=AB[2]; ICOMP[j+2]:=AB[3]; ICOMP[j+3]:=AB[4];
end;

procedure TForeval.CALLA;
begin
AddWord($FF,$D0);
end;
procedure TForeval.CALLB;
begin
AddWord($FF,$D3);
end;
procedure TForeval.MOVAM(N: TAddress);
begin
//EAX->MEM
AddWord($89,$05);
AddAddress(N);
end;
procedure TForeval.MOVMA(N: TAddress);
begin
//MEM->EAX
AddWord($8B,$05);
AddAddress(N);
end;
procedure TForeval.MOVA1(N: TAddress);
begin
//N - число
AddByte($B8);
AddAddress(N);
end;
procedure TForeval.FLD(N: TAddress);
begin
AddWord($DD,$05);
AddAddress(N);
end;
procedure TForeval.FSTP(N: TAddress);
```

```
begin
AddWord($DD,$1D);
AddAddress(N);
end;
procedure TForeval.PUSHA;
begin
AddByte($50);
end;
procedure TForeval.POPA;
begin
AddByte($58);
end;
end.
```

Файл About.pas - вікно «Про програму...»

```
unit uAbout;

interface
uses
  Classes,
  Graphics,
  Controls,
  Forms,
  StdCtrls,
  ExtCtrls,
  ShellApi,
  jpeg, Buttons;

type
  TfmAbout = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    BitBtn1: TBitBtn;
    Image1: TImage;
    procedure Label7Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private    { Private declarations }
  public    { Public declarations }
  end;

var fmAbout: TfmAbout;

implementation

{$R *.DFM}
procedure TfmAbout.BitBtn1Click(Sender: TObject);
begin
fmAbout.Close;
end;

procedure TfmAbout.FormCreate(Sender: TObject);
begin
Label1.Caption:='МАГІСТЕРСЬКА РОБОТА';
Label2.Caption:='на тему:';
Label3.Caption:='Дослідження та програмна реалізація системи управління
технологічною безпекою на основі імовірнісних структурно-логічних моделей
небезпек виробництв';
Label5.Caption:='Розробив: студент  Макеєв Андрій Віталійович';
Label6.Caption:='гр. КІ-22МЗ';
Label7.Caption:='Керівник: Смірнов О.А.';
Label8.Caption:='м. Кропивницький 2023';
end;
end.
```