МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Аналіз і розроблення спеціалізованої бази даних програмно-технічного комплексу

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 "Комп'ютерна інженерія" (освітня програма - "Системне програмування")

Науковий керівник роботи:	_____ (підпис)	Є.В.Щербаков (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О.Критська (ініціали, прізвище)
Студент:	_____ (підпис)	Д.І. Макаровський (ініціали, прізвище)
Група:		СП-16дм

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 "Комп'ютерна інженерія" (освітня програма - "Системне програмування")
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Макаровському Денису Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз і розроблення спеціалізованої бази даних програмно-технічного комплексу

керівник проекту (роботи) Щербаков Євген Васильович, к.т.н., доц.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «18» 10 2018 р. № 208/48
2. Строк подання студентом роботи 21.01.2018
3. Вихідні дані до роботи Матеріали науково-дослідної практики, структура файлів архівів ОБД МСКУ-М, середовище розробки Qt

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) АНАЛІЗ БАЗОВИХ ОБ'ЄКТІВ ПРОЕКТУ, РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ, ЗАСТОСУВАННЯ РОЗРОБЛЕНОГО ПЗ, ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ, ВИСНОВКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Щербаков Є.В. доцент кафедри КНІ		
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. асистент кафедри КНІ		

7. Дата видачі завдання 10.09.2017

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	10.09.2017-15.09.2017	
2	Аналіз завдання, постановка задачі, підбір і вивчення літератури	16.09.2017-22.09.2017	
3	Аналіз предметної області	23.09.2017-25.09.2017	
4	Розробка графічного інтерфейсу	26.09.2017-06.10.2017	
5	Розробка слотів і функцій	07.10.2017-13.11.2007	
6	Тестування програмного засобу	14.11.2017-30.11.2017	
7	Оформлення графічної частини проекту	01.12.2017-31.12.2017	
8	Оформлення пояснювальної записки	01.01.2018 – 10.01.2018	

Студент _____ Макаровський Д.І.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Щербаков Є.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Макаровський Д.І. Аналіз і розроблення спеціалізованої бази даних програмно-технічного комплексу.

Пояснювальна записка до дипломного проекту містить: 139 сторінок, 26 рисунків, 8 таблиць, 21 джерело, 17 електронних плакатів.

Об'єктами дослідження дипломного проекту є файли архівів AEF_Arch і HDA_Arch узагальненої бази даних МСКУ-М.

Мета роботи полягає в розробці модуля відображення експертної системи аналізу баз даних САОЗ і СУОР-І (програмний засіб).

В основній частині пояснювальної записки описані засоби і методи, використані для реалізації поставлених завдань: середовище розробки екранного інтерфейсу - Qt Designer, використання механізму «сигнал-слот». Там же наведені схеми алгоритмів функцій і слотів проекту. В окремому розділі наведено приклад використання розробленого програмного засобу. Відповідно до завдання було отримано програмний засіб, що надає користувачеві зручний інтерфейс для перегляду і первинного аналізу файлів архівів ОБД МСКУ-М. Проектом реалізований двохсторінковий інтерфейс, що надає можливість аналізувати і зіставляти інформацію з двох архівів: подій і параметрів технологічного процесу. Реалізована можливість відображення інформації зі зв'язаних файлів архівів у вигляді таблиць. У віконця виведення видаються повідомлення про хід аналізу відкриваються файлів, а так само вміст полів заголовка файлів. Виконано оцінку шкідливих і небезпечних факторів, що впливають на безпеку роботи оператора персонального комп’ютера.

Ключові слова: ІНЖЕНЕРНО-ДІАГНОСТИЧНА СТАНЦІЯ, КОМПЛЕКС ТЕХНОЛОГІЧНИХ ЗАХИСТІВ, ОБД МСКУ-М, СУБД, ЕКРАННИЙ ІНТЕРФЕЙС.

THE ABSTRACT

Makarovsky D.I. Analysis and development of a specialized database of software and hardware complex.

Master thesis contains: 139 pages, 26 drawings, 8 tables, 21 sources, 17electronic posters.

The objects of the study of the diploma project are the archive files of AEF_Arch and HDA_Arch of the generalized MSCU-M database.

The purpose of the work is to develop a module for displaying an expert system for analyzing databases of CAOZ and SUOR-I (software).

The main part of the explanatory note describes the tools and methods used to implement the tasks: the environment for developing the on-screen interface - Qt Designer, the use of the signal-slot mechanism. There are also schematics of algorithms of functions and project slots. A separate section provides an example of the use of the developed software. In accordance with the task, a software tool was provided that provides the user with a user-friendly interface for viewing and initial analysis of archive files of the OSB MSCU-M archive. The project implemented a two-page interface, which provides an opportunity to analyze and compare information from two archives: events and parameters of the technological process. The ability to display information from linked archive files in the form of tables is implemented. In the output window, you'll see the progress of the analysis of the opened files, as well as the contents of the fields of the file header. The estimation of harmful and dangerous factors influencing the safety of the operator of a personal computer is executed.

Key words: ENGINEERING-DIAGNOSTIC STATION, COMPLEX OF TECHNOLOGICAL PROTECTORS, DB MSCU-M, DBMS, SCREEN INTERFACE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ....7
ВСТУП....8
1 АНАЛІЗ БАЗОВИХ ОБ'ЄКТІВ ПРОЕКТУ....10
1.1 КТЗ-І. Короткий опис....10
1.1.1 Склад і функції КТЗ-І....10
1.1.2 Технічні характеристики ІДС....12
1.1.3 Загальний аналіз ПЗ КТЗ-І....12
1.2 Значимість розробок, пов'язаних з КТЗ-І....13
1.3 Об'єкти аналізу проекту....15
1.3.1 УБД МСКУ-М як середовище створення архівів....15
1.3.2 Логічна структура файлів архівів....15
1.3.3 Зв'язки і зміст файлів архівів....16
1.4 Технічне завдання на розробку....17
2 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ....27
2.1 Розробка структурної схеми програми....27
2.2 Розробка екранного інтерфейсу програми....28
2.2.1 Особливості середовища розробки....28
2.2.2 Компоненти графічного інтерфейсу програми....29
2.3 Опис глобальних змінних....30
2.4 Розробка алгоритмів функцій і слотів....32
2.4.1 Розробка інтерфейсних функцій і слотів....32
2.4.1.4 Функції виводу поточних повідомлень і повідомлень про помилки аналізу заголовків файлів....36
2.4.2 Розробка основних і допоміжних функцій і слотів....43
2.4.3 Головний модуль програми....68
3 ЗАСТОСУВАННЯ РОЗРОБЛЕНОГО ПЗ....70
3.1 Приклад використання ПЗ МВФА....70
3.2 Керівництво користувача....75
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ....77
4.1 Аналіз потенційно небезпечних і шкідливих виробничих факторів , що впливають на персонал....77
4.2 Заходи щодо техніки безпеки....80
4.3 Заходи, що забезпечують виробничу санітарію і гігієну праці....83

4.4 Рекомендації по пожежній профілактиці 86
ВИСНОВКИ 89
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....90
ДОДАТОК А. Лістинг модулю mainform.ui.h 92
ДОДАТОК Б. Структури заголовків та записів файлів архівів.....126
ДОДАТОК В. Електронні плакати.....130

ПЕРЕЛІК СКОРОЧЕНЬ

АСУ ТП - Автоматизована система управління технологічним процесом

БД - База даних

ІС - Інтелектуальна система

ІДС - Інженерно-діагностична станція

КТЗ-І - Комплекс технологічних захистів

МСКУ - Мікропроцесорний субкомплексу контролю і управління

ОБД - Оперативна база даних

ОС - Операційна система

ПО - Програмне забезпечення

ПТК - Програмно-технічний комплекс

ПЗ - Програмний засіб

ПЕОМ - Персональна електронно-обчислювальна машина

САОЗ - Програмно-технічна система аварійного охолодження активної зони енергетичного реактора ВВЕР-440

ССО – Стійка сервісного обладнання

СУБД - Система управління базою даних

СУОР-І - Система захисту і управління органами регулювання реактора

ТО - Технологічне обладнання

УКЗ - Пристрій контролю та захисту

ЕС - Експертна система

ЕОМ - Електронно-обчислювальна машина

AEF_Arch - Архів подій і повідомлень про порушення ОБД МСКУ-М

HDA_Arch - Архів параметрів технологічного процесу ОБД МСКУ-М

MS-DOS - Дискова операційна система фірми Microsoft

SQL - Мова структурованих запитів

ВСТУП

В даний час на ПрАТ «СНВО «Імпульс» (далі - підприємство) ведуться розробки з модернізації програмно-технічного комплексу технологічних захистів (далі - КТЗ-І).

КТЗ-І орієнтований на застосування в якості обладнання АСКТП при реконструкції діючих і створенні нових управляючих систем безпеки і систем нормальної експлуатації.

Одна з областей практичного застосування КТЗ-І - система аварійного охолодження активної зони (далі - САОЗ) водо-водяного енергетичного реактора ВВЕР-440, призначена для відводу тепла від активної зони реактора з метою екстреного перекладу технологічного обладнання, яке в безпечний стан після спрацювання аварійного захисту.

САОЗ складається з трьох ідентичних незалежних каналів. КТЗ-І застосовується в якості центральної частини обладнання АСКТП кожного каналу САОЗ. На рис 1.1 наведена структурна схема трьох каналів САОЗ.

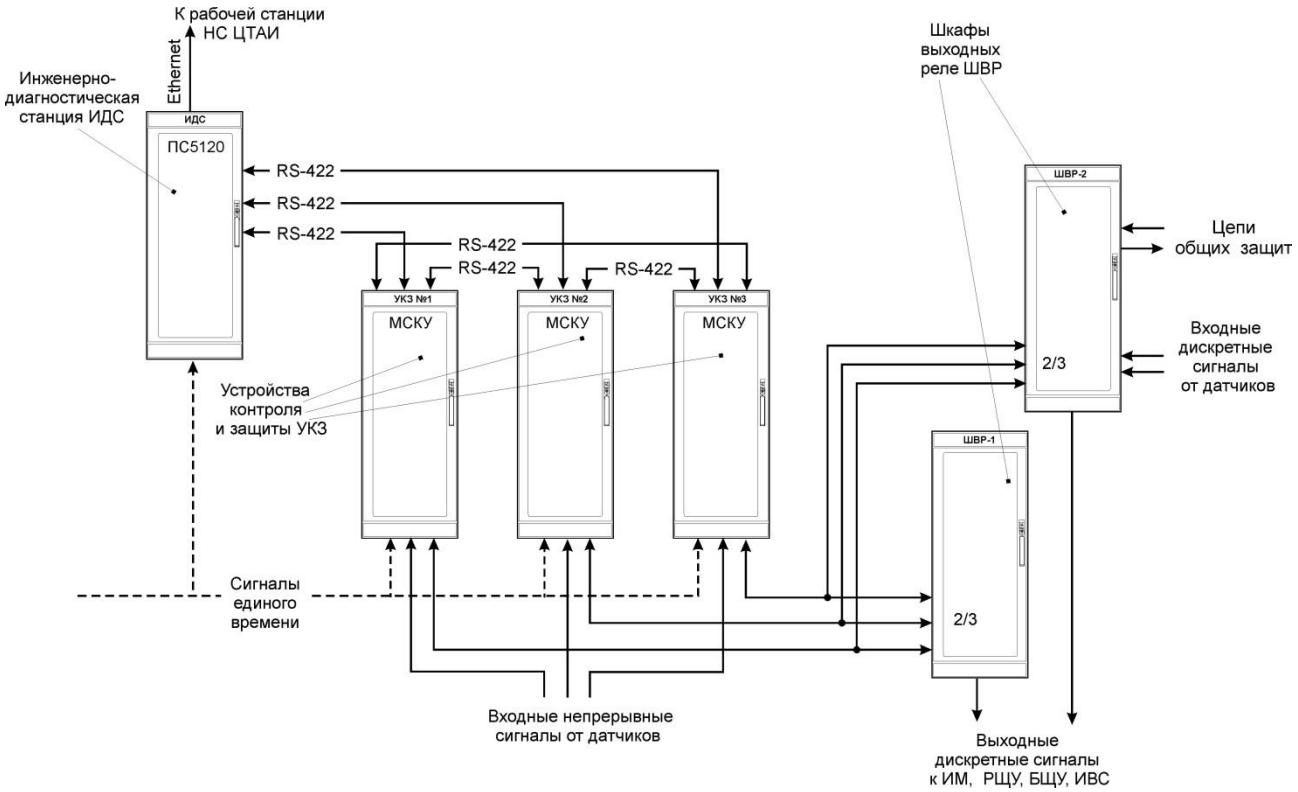


Рисунок 1 - Структура трьох каналів САОЗ

КТЗ-І складається з наступних функціональних підсистем [1]:

– захисту;

– контролю і діагностування;

– випробування і калібрування.

Серед перерахованих підсистем слід відзначити підсистему контролю та діагностування. Дану підсистему утворює інженерно-діагностична станція (далі - ІДС) з програмним забезпеченням (далі - ПЗ).

ІДС (ПС5120) зі складу КТЗ-І забезпечує:

– прийом і обробку даних від пристроїв контролю і захисту (далі - ПКЗ) (МСКУЗ);

– візуалізацію поточної діагностичної та технологічної інформації;

– ведення протоколу подій і порушень по кожному з ПКЗ

– (МСКУ 3);

– ведення протоколів виконання періодичного тестування в режимі «Випробування»;

– архівування інформації;

– перегляд архівних параметрів на екрані монітора;

– передачу інформації в стійку сервісного обладнання (далі - ССО) при проведенні періодичного контролю технічного стану КТЗ-І в період ППР;

– передачу інформації в інші системи по мережі Ethernet.

В даний час, поряд з оновленням всього ПЗ КТЗ-І, розробниками підприємства проводиться модернізація ПЗ ІДС.

На підприємстві розроблено ряд модулів, що функціонують під управлінням ОС MS Linux, що дозволяють виводити на екран вміст файлів архівів узагальненої бази даних (далі - УБД) МСКУ-М. Але, так як ПЗ, яке функціонує під управлінням MS Linux, не повній мірі задовольняє сучасним вимогам до ПЗ ІДС, то перед розробниками підприємства була поставлена задача з модернізації засобів аналізу і візуалізації файлів архівів УБД МСКУ-М - завдання з створенню експертної системи аналізу баз даних САОЗ.

Зазначена експертна система повинна функціонувати під управлінням ОС Linux. Для розробки графічного інтерфейсу системи повинні використовуватися сучасні засоби (середовище розробки Qt Designer).

Метою магістерської роботи є розробка модуля відображення файлів архівів УБД МСКУ-М експертної системи аналізу баз даних САОЗ і СУОР-І.

1 АНАЛІЗ БАЗОВИХ ОБ'ЄКТІВ ПРОЕКТУ

В даному розділі наводиться коротка характеристика КТЗ-І і тієї його частини (ПЗ ІДС), яка безпосередньо відноситься до теми даного дипломного проекту. У пункті 1.3 дається опис об'єктів аналізу проекту. Відповідно до методичних вказівок до дипломного проектування, розділ закінчується докладним завданням на розробку.

1.1 КТЗ-І. Короткий опис

В даному підрозділі наведені основні характеристики КТЗ-І [2]:

– склад і функції КТЗ-І (див. 1.1.1);
– технічні характеристики ІДС (див. 1.1.2);
– загальний аналіз ПЗ КТЗ-І (див. 1.1.3) і склад ПЗ ІДС (див. 1.1.3.1).

1.1.1 Склад і функції КТЗ-І

Склад КТЗ-І із зазначенням приналежності вхідних пристроїв до класів безпеки, груп умов експлуатації та розміщення наведено у таблиці 1.1.

Таблиця 1.1 - Склад КТЗ-І

Найменування	Кіл-ть	Клас безпеки	Група умов експлуатації
Робоча станція ПС5120.21	1	3Н	2.2
Шафа вихідних реле ШВР-1	1	2У	2.2
Шафа вихідних реле ШВР-2	1	2У	2.2
Комплекс керуючий обчислювальний МСКУ 3.01/002 АС	3	2У	2.2
Стійка сервісного обладнання ССО	1	4	2.2

КТЗ-І виконує наступні основні функції:

– прийом і первинну обробку вхідних безперервних і дискретних сигналів, визначення поточних значень технологічних параметрів;

– порівняння поточних значень технологічних параметрів із заданими граничними значеннями (уставками), виконання обчислювальних і логічних операцій відповідно до алгоритмів спрацьовування захистів;

– логічну обробку в кожному каналі (ПКЗ) сигналів, які формуються в ньому самому і в двох інших каналах (ПКЗ) при ідентифікації вихідної події спрацьовування захисту;

– формування і видачу команд захисту при виході контрольованих технологічних параметрів за межі встановлених граничних значень (уставок) і / або за іншими умовами спрацьовування захисту;

– формування і видачу на БЩК і РЩК сигналів при виході поточних значень технологічних параметрів за межі встановлених граничних значень і при виявленні несправностей технічних засобів, які контролюють вихід параметрів за межі уставок;

– формування і видачу в ІОС сигналів про спрацювання САОЗ за окремими видами захистів;

– прийом сигналів від мережі єдиного всесвітнього координованого часу, відлік часу за допомогою вбудованих засобів, синхронізацію результатів незалежного рахунку з мережею єдиного часу.

КТЗ-І функціонує в наступних режимах за допомогою переведення перемикача панелі випробування в один з пунктів:

– режим «Робота»;

– режим «Випробування».

У режимі «Робота» КТЗ-І виконує всі основні функції і такі допоміжні функції:

– безперервний автоматичний контроль вхідних і вихідних безперервних і дискретних сигналів, архівування, відображення і реєстрацію даних про поточні значення технологічних параметрів, стан технологічного обладнання, спрацьовування захистів і умовах, що викликали ці спрацьовування;

– безперервний автоматичний контроль технічного стану апаратних засобів (ПКЗ, ШВР-1, ШВР-2) і програмного забезпечення, архівування, відображення і реєстрацію даних про час, місце і характер виявлених дефектів.

У режимі «Випробування» КТЗ-І забезпечує виконання допоміжних функцій і наступних додаткових допоміжних функцій:

– автоматизований періодичний контроль (випробування) технічного стану апаратних і програмних засобів за допомогою перевіряючих тестів (без порушення вхідних ланцюгів);

– автоматизована перевірка справності шляхом імітації умов спрацьовування захистів і автономної перевірки технічних засобів зі складу комплексу технологічних захистів (КТЗ-І) з використанням ССО;

– автоматизована перевірка (калібрування) вимірювальних каналів.

У КТЗ-І передбачена можливість блокування спрацьовування уставок по кожному з вхідних сигналів за допомогою переказу тумблера в положення ВКЛЮЧЕНО на відповідному пристрої блокування технологічних параметрів. При блокуванні вхідного сигналу здійснюється вивід сигналу «Несправність» в схему технологічної сигналізації на БЩК за даним технологічним параметром. Пристрої блокування по кожному вхідному сигналу мають обмежений доступ.

1.1.2 Технічні характеристики ІДС

Так як тема дипломного проекту пов'язана з ПЗ ІДС, то нижче наводяться технічні характеристики ІДС.

Інженерно-діагностична станція ІДС (ПС5120) зі складу КТЗ-І забезпечує:

– прийом і обробку даних від кожного пристрою контролю і захисту ПКЗ (МСКУ 3);

– візуалізацію поточної діагностичної та технологічної інформації;

– ведення протоколу подій і порушень по кожному з пристроїв контролю і захисту ПКЗ (МСКУ 3);

– ведення протоколів виконання періодичного тестування в режимі «Випробування»;

– архівування інформації;

– перегляд архівних параметрів на екрані монітора;

– передачу інформації в стійку сервісного обладнання ССО при проведенні періодичного контролю технічного стану КТЗ-І в період ППР;

– передачу інформації в інші системи по мережі Ethernet.

1.1.3 Загальний аналіз ПЗ КТЗ-І

Програмне забезпечення КТЗ-І відповідає вимогам, що пред'являються до ПЗ щодо забезпечення ядерної та радіаційної безпеки АЕС [3]. ПЗ КТЗ-І складається з:

– програмного забезпечення пристрою контролю і захисту ПКЗ (МСКУ 3) класу безпеки 2 з реалізації функцій захисту і блокування;

– програмного забезпечення інженерно-діагностичної станції ІДС (ПС5120) класу безпеки 3 з реалізації функцій безперервного і періодичного контролю технічного стану КТЗ-І в режимі «Випробування»;

– програмного забезпечення стійки сервісного обладнання ССО класу безпеки 4 з реалізації функцій перевірки справності КТЗ-І та повірки (калібрування) вимірювальних каналів КТЗ-І.

ПЗ пристрою контролю і захисту ПКЗ (МСКУ 3) має модульну структуру. Текст одного модуля містить обмежену кількість операторів, має ясну структуру, легко модифікується і тестується, використання переривань обмежено прийомом сигналів від таймера. ПЗ ПКЗ (МСКУ 3) здійснює безперервний автоматичний контроль технічного стану і визначає несправність до змінної одиниці (блок, модуль) технічних засобів.

Програмне забезпечення забезпечує можливість проведення періодичного контролю технічного стану КТЗ-І і самодіагностику (самоконтроль) програмних засобів, захист від помилок обслуговуючого персоналу і захист від несанкціонованого доступу.

Реалізація програмами безперервного автоматичного контролю не впливає на виконання програм з реалізації функцій захисту.

Прикладне ПЗ ІДС складається з [1]:

– програми контролю цілісності програмного забезпечення;
– програми контролю технічних засобів;
– програми контролю завантажених модулів;
– програми підтримки єдиного часу;
– програми управління джерелом безперебійного живлення;
– програми обслуговування пристроїв зв'язку з ПКЗ;
– програми доступу до ПКЗ по односпрямованому інтерфейсу;
– програми ведення бази даних ІДС, прийому і обробки даних від ПКЗ;
– програми формування та архівування протоколів випробування;
– програми формування та архівування подій і повідомлень про порушення;
– програми архівування параметрів;
– програми відображення інформації.

1.2 Значимість розробок, пов'язаних з КТЗ-І

На сьогоднішній день використання КТЗ-І в якості обладнання АСКТП при реконструкції САОЗ реакторної установки типу ВВЕР-440 дозволяє [2]:

– замінити використане і морально застаріле обладнання АСКТП діючих САОЗ сучасними програмно-технічними засобами;

– підвищити надійність (безвідмовність, ремонтопридатність, довговічність) в порівнянні з обладнанням АСКТП, застосовуваним в діючих САОЗ;

– забезпечити відповідність обладнання АСКТП вимогам діючих в Україні норм і правил з ядерної та радіаційної безпеки;

– підвищити якості (глибини, достовірності, оперативності) діагностування технологічного обладнання САОЗ і власних технічних і програмних засобів КТЗ-І;

– покращити інформаційну підтримку персоналу при використанні за призначенням і обслуговуванні КТЗ-І, зменшення ймовірності помилкових дій персоналу;

– фіксувати зміни станів вхідних і вихідних сигналів до і після виникнення вихідної події, що викликала спрацьовування захисту;

– оперативно відображати дані про стан технологічного обладнання та про роботу САОЗ;

– забезпечити можливість збільшення обсягу даних, які передаються в інформаційно-обчислювальну систему (ІОС) енергоблоку;

– проводити архівування подій і станів у реальному часі (для подальшого аналізу) та зберігання архівних даних за тривалий період;

– контролювати та відображати стан обладнання технологічного захисту (КТЗ-І) у процесі експлуатації;

– видавати сигнали тривоги при виявленні дефектів, визначати їх місце, відображати, архівувати і реєструвати діагностичні повідомлення;

– спростити перевірку готовності САОЗ і КТЗ-І перед пуском енергоблоку і скоротити час перевірки;

– зменшити витрати часу і ризик можливих помилок при технічному обслуговуванні обладнання технологічних захистів;

– автоматизувати операції періодичного контролю (випробування) та перевірки справності КТЗ-І, а також документування їх результатів;

– забезпечити термін служби компонентів обладнання АСКТП в складі САОЗ, що відповідає встановленому терміну служби енергоблоку.

Зі змісту даного і попереднього підрозділів можна зробити висновок, що модернізація і розробки в галузі програмного забезпечення КТЗ-І є важливим завданням, реалізація яких дозволяє удосконалити характеристики КТЗ-І.

1.3 Об'єкти аналізу проекту

В даному розділі наведені й описані об'єкти аналізу даного дипломного проекту - файли архівів AEF_Arch і HDA_Arch. Дана коротка інформація про середовище створення зазначених архівів (див. 1.3.1), логічної та фізичної структури файлів архівів (див. 1.3. 2), описані зв'язки між файлами архівів (див. 1.3. 3).

В даному дипломному проекті файли архівів AEF_Arch і HDA_Arch розглядаються як об'єкти аналізу.

Зазначені архіви створюються в УБД МСКУ-М (дивись 1.3.1.1). У файлах архівів містяться поточні значення ТП, інформація про стан ТО, про спрацювання захистів і умовах, що викликали ці спрацьовування [1].

1.3.1 УБД МСКУ-М як середовище створення архівів

Відповідно до 1.1.3, ПЗ ІДС складається з декількох програм, серед яких слід відзначити програми, що забезпечують прийом, обробку, зберігання та доступ додатків (локальних і віддалених) до інформації, прийнятої від ПКЗ. Зазначені програми призначені для організації в шлюзовий ПЕОМ узагальненої бази даних МСКУ - М. Тут інформація, яка приймається від ПКЗ, записується в область, доступну прикладним процесам і архівується для подальшого використання. Таким чином, створюються архіви AEF_Arch і HDA_Arch.

1.3.2 Логічна структура файлів архівів

Логічна структура файлів архівів приведена на малюнку 1.1 [1].

<table>
<tr><td colspan="4">Заголовок файлу</td><td colspan="8">Послідовність записів</td></tr>
<tr><td colspan="4">Поля заголовку</td><td colspan="4">Запис 1</td><td>…</td><td colspan="4">Запис m</td></tr>
<tr><td>Поле 1</td><td>Поле 2</td><td>…</td><td>Поле n</td><td>Поле 1</td><td>Поле 2</td><td>…</td><td>Поле k</td><td>…</td><td>Поле 1</td><td>Поле 2</td><td>…</td><td>Поле p</td></tr>
</table>

Рисунок 1.1 – Логічна структура файлів архівів AEF_Arch и HDA_Arch

Як показано на рис. 1.1, файли складаються з заголовку, що складається з інформаційних полів, і послідовності записів. У полях заголовку міститься ключова інформація про походження файлу, його стан і зв'язках з іншими файлами архіву.

1.3.3 Зв'язки і зміст файлів архівів

Коли в УБД МСКУ-М сформовано повідомлення про порушення, то інформація про дане повідомлення і вміст повідомлення розміщується в декількох пов'язаних файлах архіву AEF_Arch. У таблиці 1.2 подано інформацію про файлах вказаного архіву [1].

Таблиця 1.2 - Файли архіву AEF_Arch

Ім’я файлу	Призначення
Event_Common.aef	Файл опису взаємозв'язків між частинами зберігання повідомлень (основний)
DDMMYYYY_Event.aef	Файл зберігання основної частини повідомлення

Зауваження. В імені файлу DDMMYYYY_Event.aef: DDMMYYYY - дата і рік створення файлу, де DD - число, MM місяць, YYYY - рік.

У таблиці 1.3 наведена інформація про фізичну структуру файлів архівів AEF_Arch, де вказуються імена структур, які використовуються для зберігання інформації.

Таблиця 1.3 - Фізична структура файлів архіву AEF_Arch

Ім’я файлу	Ім'я структури заголовку	Ім'я структури записи
Event_Common.aef	struct tag_TDA_AEF_HeadFV	struct tag_TDA_AEF_EventDescrWrite
DDMMYYYY_Event.aef	struct tag_TDA_AEF_DataHeadFV	struct tag_TDA_AEF_EventDinWrite

Зауваження. Зазначені в таблиці 1.3 структури описані в додатку Б.

Файли архіву AEF_Arch пов'язані за полем заголовку IdentCode (unsigned long).

В архівах HDA_Arch розміщується інформація про параметри технологічного процесу. Інформація розподіляється по декільком пов'язаним файлам.

У таблиці 1.4 наведено перелік і призначення файлів HDA_Arch [1].

Таблиця 1.4 - Файли архіву HDA_Arch

Ім’я файлу	Призначення
Source_Common.hda	файл опису атрибутів зберігання даних
DD_MM_YYYY_Value.hda	файл даних параметрів технологічного процесу

Зауваження. В імені файлу DD_MM_YYYY_Value.hda: DD_MM_YYYY - дата і рік створення файлу, де DD - число, MM місяць, YYYY - рік.

У таблиці 1.5 наведена інформація про фізичну структуру файлів HDA_Arch.

Таблиця 1.5 - Фізична структура файлів архіву HDA_Arch

Ім’я файлу	Структура заголовку	Структура записів
Source_Common.hda	struct tag_TDA_HDA_HeadFV	struct tag_TDA_HDA_ATTRIBUTEFV
DD_MM_YYYY_Value.hda	struct tag_TDA_HDA_DataHeadFV	struct tag_TDA_HDA_DataFV

Зауваження. Зазначені в таблиці 1.5 структури описані в додатку Б.

Файли архіву HDA_Arch пов'язані по полю заголовка IdentCode (unsigned long).

1.4 Технічне завдання на розробку

1.4.1 Дані вимоги поширюються на розробку програмного засобу модуля відображення файлів архівів (далі - ПЗ МВФА) узагальненої бази даних МСКУ-М експертної системи аналізу баз даних САОЗ і СУОР-І.

1.4.2 ПЗ МВФА призначений для виконання наступних завдань:

– ідентифікації файлів архівів УБД МСКУ-М (далі - файлів архівів)
– надання персоналу інформації про взаємодію між файлами архівів;
– інформаційної підтримки персоналу при виконанні аналізу файлів архівів.

1.4.3 Розробка виконується в зв'язку з модернізацією ПЗ КТЗ-І.

1.4.4 Розроблюваний ПЗ МВФА направлено на реалізацію наступних основних цілей або завдань:

– проектування програми у вигляді ієрархічної деревоподібної структури;
– проектування екранного інтерфейсу на основі системи спливаючих меню;

– розробка двохсторінкового екранного інтерфейсу, що забезпечує персонал інформацією про стан і зміст файлів архіву AEF (далі - AEF_Arch) на одній сторінці, а файлів архіву HDA (далі - HDA_Arch) - на інший;

– надання персоналу поточної інформації про хід аналізу файлів архівів;

– ідентифікація файлів архівів і виведення на екран їх вмісту у вигляді таблиць (по дві таблиці для кожної сторінки додатка);

– надання персоналу пов'язаної інформації з декількох файлів архіву (для обох архівів);

– забезпечення декодування зчитаних з файлів архівів даних;

– використання при розробці коду програми методів, що дозволяють легко проводити модифікацію програми.

1.4.5 Для забезпечення зазначених цілей і завдань ПЗ МВФА має виконувати такі основні функції:

– функціонування двохсторінкового екранного інтерфейсу, побудованого на основі системи спливаючого меню;

– відкриття файлів архівів;

– аналіз і читання заголовків файлів архівів;

– візуалізацію вмісту файлів архівів;

– виведення текстових повідомлень.

1.4.5.1 Двохсторінковий екранний інтерфейс повинен забезпечувати:

– розміщення інформації про кожного з архівів (AEF_Arch і HDA_Arch) на окремій сторінці додатка, з можливістю почергового перегляду сторінок;

– виклик сторінки додатка AEF_Arch;

– виклик сторінки додатка HDA_Arch;

– перемикання сторінок додатку;

– приведення до початкового стану таблиці №1 AEF_Arch;

– приведення до початкового стану таблиці №2 AEF_Arch;

– приведення до початкового стану таблиці №1 HDA_Arch;

– приведення до початкового стану таблиці №2 HDA_Arch;

– завершення роботи програми.

1.4.5.2 Відкриття файлів архівів має забезпечувати:

– відкриття файлів архіву AEF_Arch;

– відкриття файлів архіву HDA_Arch;

– завантаження файлів архіву AEF_Arch;

– завантаження пов'язаного файлу архіву AEF_Arch;

– завантаження файлів архіву HDA_Arch;

– завантаження пов'язаного файлу архіву HDA_Arch.

1.4.5.3 Аналіз і читання заголовків файлів повинні забезпечувати:

– читання і ідентифікацію заголовка основного файлу архіву AEF_Arch;

– читання і ідентифікацію заголовка основного файлу архіву HDA_Arch;

– читання і ідентифікацію заголовка пов'язаного файлу архіву AEF_Arch;

– читання і ідентифікацію заголовка пов'язаного файлу архіву HDA_Arch.

1.4.5.4 Візуалізація вмісту файлів архівів повинна забезпечувати:

– читання основного файлу архіву AEF_Arch;

– читання основного файлу архіву HDA_Arch;

– читання зв'язаного файлу архіву AEF_Arch;

– читання зв'язаного файлу архіву HDA_Arch;

– отримання і установку в поточну позицію пов'язаних даних AEF_Arch;

– отримання і установку в поточну позицію пов'язаних даних HDA_Arch.

1.4.5.5 Виведення текстових повідомлень повинно забезпечувати:

– виведення на екран поточних повідомлень по ходу відкриття основного файлу AEF_Arch;

– виведення на екран поточних повідомлень по ходу відкриття основного файлу HDA_Arch;

– виведення на екран інформації про помилки, що виникли по ходу аналізу заголовків файлів обох архівів;

– виведення на екран повідомлення про відсутність пов'язаних даних.

1.4.6 Для забезпечення зазначених в пункті 1.4.4 цілей і завдань

– ПЗ МВФА має виконувати такі допоміжні функції:

– ініціалізація змінних і вказівників на об'єкти;

– декодування лічених з файлів архівів даних;

– інші.

1.4.6.1 Ініціалізація вказівників на об'єкти і змінних повинна забезпечувати:

– ініціалізацію змінних і вказівників, формування початкових установок для екранного інтерфейсу при запуску програми.

1.4.6.2 Декодування зчитаних з файлів архівів даних має забезпечувати:

– декодування даних в Unicode;

– декодування даних з Unicode.

1.4.6.3 Інші функції повинні забезпечувати:

– копіювання об'єкта QString в масив типу char;

– копіювання об'єктів типу unsigned char в об'єкт типу QString.

1.4.7 ПЗ МВФА має функціонувати в ПЕОМ, що має в своєму складі:

– процесор Pentium з тактовою частотою 1,4 GHz;

– оперативна пам'ять загальною ємністю 256 Mbyte;

– накопичувач на ЖМД з ємністю дискової пам'яті 40 Gbyte;

– накопичувач ГМД (3.5 ", 1,44 Mbyte);

– алфавітно-цифрова клавіатура;

– монітор TFT 17";

– пристрій безперебійного живлення.

1.4.8 Програмний засіб МВФА має включати в себе:

– прикладне ПЗ, що забезпечує виконання функцій МВФА.

1.4.9 Швидкість читання даних з файлів і виведення інформації в таблиці - не менше ніж для базового комплексу МСКУ 3.

1.4.10 ПЗ МВФА має бути розроблено в середовищі Qt Designer.

1.4.11 ПЗ МВФА має бути розроблено на мові програмування С ++.

1.4.12 ПЗ МВФА має функціонувати під управлінням ОС Linux з ядром RadHat версії 2.4.9 або ядром RadHat версії 2.4.18.

1.4.13 Склад, найменування, позначення, тип значення функцій і слотів ПЗ МВФА повинні відповідати, наведеними в таблиці 1.6.

Таблиця 1.6 - Склад функцій і слотів ПЗ МВФА

Пункт завдання	Найменування функції або слота	Тип значення, що повертається, позначення функції або слота, формальні параметри	Вхідні дані	Вихідні дані
1	2	3	4	5
Слоти				
1.4.5.2	Слот відкриття файлів архіву AEF_Arch	void fileOpen_AEF()	-	Отримання імені файлу, що відкривається.Зав антаження файлу

Продовження табл. 1.6

1	2	3	4	5
1.4.5.4	Слот отримання і установки в поточну позицію пов'язаних даних AEF_Arch	void getLinkedData()	Поточний осередок головної таблиці (AEF)	Знайдений осередок, який став поточним у таблиці пов'язаних даних AEF_Arch
1.4.5.5	Слот виведення на екран повідомлення про відсутність пов'язаних даних	void AEF_Mes_NoLinked-Data(int val, char * namefile)	Файл даних; вміст осередку	Повідомлення про відсутність пов'язаних даних в підпорядкованій таблиці
1.4.5.1	Слот виклику сторінки додатка AEF_Arch	void raise_AEF()	-	Промальовування сторінки додатка AEF_Arch
1.4.5.1	Слот виклику сторінки додатка HDA_Arch	void raise_HDA()	-	Промальовування сторінки додатка HDA_Arch
1.4.5.1	Слот «очищення» таблиці №1HDA_Arch	void clear_HDA_t1()	Кількість рядків таблиці	Таблиця №1 HDA_Arch з порожніми осередками
1.4.5.1	Слот «очищення» таблиці №2 HDA_Arch	void clear_HDA_t2()	Кількість рядків таблиці	Таблиця №2 HDA_Arch Arch з порожніми осередками
1.4.5.4	Слот отримання і установки в поточну позицію пов'язаних данихHDA_Arch	void getLinked-Data_HDA()	Поточний осередок головної таблиці (HDA)	Знайдений осередок, який став поточним у таблиці пов'язаних даних HDA_Arch

Продовження табл. 1.6

1	2	3	4	5
1.4.5.1	Слот «перемикання» сторінок додатку	void changeView(QAction * action)	Ім'я об'єкту QAction	Промальовування на екрані відповідної до імені об'єкта сторінки додатка
1.4.4.2	Слот відкриття файлів архіву HDA_Arch	void fileOpen_HDA()	-	Отримання імені файлу, що відкривається. Завантаження файлу
1.4.5.1	Слот завершення роботи програми	void fileExit()	-	Припинення роботи програми
1.4.5.1	Слот «очищення» таблиці №1 AEF_Arch	void clear_AEF_t1()	Кількість рядків таблиці	Таблиця №1 AEF_Arch з порожніми осередками
1.4.5.1	Слот «очищення» таблиці №2 AEF_Arch	void clear_AEF_t2()	Кількість рядків таблиці	Таблиця №2 AEF_Arch з порожніми осередками
Функції				
1.4.6.1	Функція-конструктор	void init()	-	Ініціалізовані глобальні змінні та вказівники
1.4.5.2	Функція завантаження основного файлу архіву AEF_Arch	void load()	Ім'я файлу, що відкривається	Заповнені осередки таблиць з інформацією із файлів
1.4.5.2	Функція завантаження пов'язаного файлу архіву AEF_Arch	void loadEventCommon()	Позиція курсора для вставки повідомлення	Виведена інформація в підпорядковану таблицю; виведені повідомлення в віконця виведення інформації

Продовження табл. 1.6

1	2	3	4	5
1.4.5.4	Функція читання основного файлу архіву AEF_Arch	int read_AEF_DataFile()	Позиція курсору для вставки повідомлення	Виведені повідомлення в віконця виведення інформації
1.4.5.4	Функція читання пов'язаних даних архіву AEF_Arch	int read_AEF_Event-Common()	Позиція курсору для вставки повідомлення	Виведені повідомлення в віконця виведення інформації
1.4.5.3	Функція читання заголовка основного файлу архіву AEF_Arch	int read_AEF_Header(char * namefile)	Ім'я аналізованого файлу	Проаналізований заголовок і виведені відповідні повідомлення в віконця виведення
1.4.5.5	Функція виведення поточних повідомлень по ходу відкриття файлів архіву AEF_Arch	void MESSAGE(int kodd)	Код повідомлення	Виведене в віконце виводу повідомлення
1.4.5.5	Функція виводу повідомлення про помилку при аналізі заголовка основного файлу архіву AEF_Arch	void ERROR_Message(int k, tag_TDA_AEF_Data-HeadFV * pHeadBuf)	Код повідомлення, вказівник на структуру заголовку файлу	Виведене в віконце виводу повідомлення
1.4.6.3	Функція отримання імені файлу, що відкривається в масив char	void setINNAME()	Довжина стр. шляху доступу до що відкривається файлу	Масив char, що містить ім'я файлу

Продовження табл. 1.6

1	2	3	4	5
1.4.6.2	Функція декодування даних в Unicode	QString& TM_CharToQString(const char *st, const char* code)	Масив елементів char з інформацією із файлу даних і назва кодування	Глоб. рядок типу QString, зі скопійованими з масиву елементів char розкодуваними даними
1.4.5.5	Функція виводу повідомлення про помилку по ходу відкриття основного файлу архіву HDA_Arch	void HDA_ERROR_Message(int er, tag_TDA_HDA_DataHeadFV * pHeadBuf_HDA)	Код повідомлення, вказівник на структуру заголовку файлу	Виведене в віконце виводу повідомлення
1.4.5.2	Функція завантаження основного файлу архіву HDA_Arch	void load_HDA()	Ім'я файлу, що відкривається	Заповнені осередки таблиць з інформацією із файлів
1.4.5.4	Функція читання основного файлу архіву HDA_Arch	int read_HDA_DataFile()	Позиція курсору для вставки повідомлення	Виведені повідомлення в віконця виводу інформації
1.4.5.2	Функція завантаження пов'язаних даних архіву HDA_Arch	void loadSourceCommon()	Позиція курсору для вставки повідомлення	Виведена інформація в підпорядковану таблицю; виведені повідомлення в віконця виводу інформації
1.4.5.3	Функція читання заголовку основного файлу архіву HDA_Arch	int read_HDA_Header(char * namefile)	Ім'я аналізованого файлу	Проаналізований заголовок і виведені відповідні повідомлення в віконця виводу

Продовження табл. 1.6

1	2	3	4	5
1.4.5.5	Функція виводу поточних повідомлень по ходу відкриття основного файлу архіву HDA_Arch	void MESSAGE_HDA(int mk)	Код повідомлення	Виведене в віконце виводу повідомлення
1.4.5.4	Функція читання пов'язаних даних архіву HDA_Arch	int read_HDA_Source-Common()	Позиція курсору для вставки повідомлення	Виведені повідомлення в віконця виводу інформації
1.4.6.2	Функція декодування даних з Unicode	char* TM_QStringTo-Char(QString *st, char *buf, const int sz, const char* code)	Оригінальний об'єкт QString, адреса пам'яті для вихідного рядку, розмір пам'яті під вихідний рядок, кодування, в якому буде виведено вихідний рядок	Масив елементів типу char
1.4.5.3	Функція читання заголовка файлу пов'язаних даних архіву HDA_Arch	int read_HDA_SCom_Header(char * namefile)	Ім'я аналізованого файлу	Проаналізований заголовок і виведені відповідні повідомлення в віконця виводу

Продовження табл. 1.6

1	2	3	4	5
1.4.5.5	Функція виводу повідомлення про помилку при аналізі заголовка файлів пов'язаних даних архіву HDA_Arch	void ERROR_SCom_Message(int er, tag_TDA_HDA_HeadFV * pHeadBufSCom)	Код повідомлення, вказівник на структуру заголовка файлу	Виведене в віконце виводу повідомлення
1.4.6.3	Функція копіювання об'єкта типу unsigned char в об'єкт типу QString	void ucharToQString(unsigned char buf)	Об'єкт типу unsigned char	Глобальна змінна зі скопійованим значенням з об'єкта unsigned char
1.4.5.3	Функція читання заголовка пов'язаних даних архіву AEF_Arch	int read_AEF_EvCom_Header(char* namefile)	Ім'я аналізованого файлу	Проаналізований заголовок і виведені відповідні повідомлення в віконця виводу
1.4.5.5	Функція виводу повідомлення про помилку при аналізі заголовка пов'язаних даних архіву AEF_Arch	void ERROR_EvCom_Message(int k, tag_TDA_AEF_HeadFV * pHeadBufEvCom)	Код повідомлення, вказівник на структуру заголовка файлу	Виведене в віконце виводу повідомлення

2 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

В даному розділі наводиться структурна схема проектованого модуля (див. рис.2.1), описані особливості середовища розробки ПЗ, компоненти графічного інтерфейсу програми, наведена інформація про глобальних змінних, використовуваних в проекті (див. Таблицю 2.2), а так же наведені схеми алгоритмів слотів і функцій, розроблених проектом.

2.1 Розробка структурної схеми програми

Структурна схема програми визначає в основних рисах і зовнішній вигляд проектованого модуля, і принципи взаємодії з користувачем.

Відповідно до завдання на розробку дипломного проекту (далі - завдання) (див. 1.4.4), схема проектованого модуля повинна являти собою ієрархічну деревоподібну структуру, що описує процедури введення, обробки і виведення даних. Побудова програм інформаційно-довідкового класу за таким принципом дозволяє досить легко робити модифікацію системи в цілому полегшує сприйняття і розуміння принципу роботи програми.

Для побудови структурної схеми необхідно визначити ієрархію і зв'язок зазначених в завданні (див. 1.4.13) функцій і процедур обробки даних.

Структурна схема програми представлена на рис. 2.1.

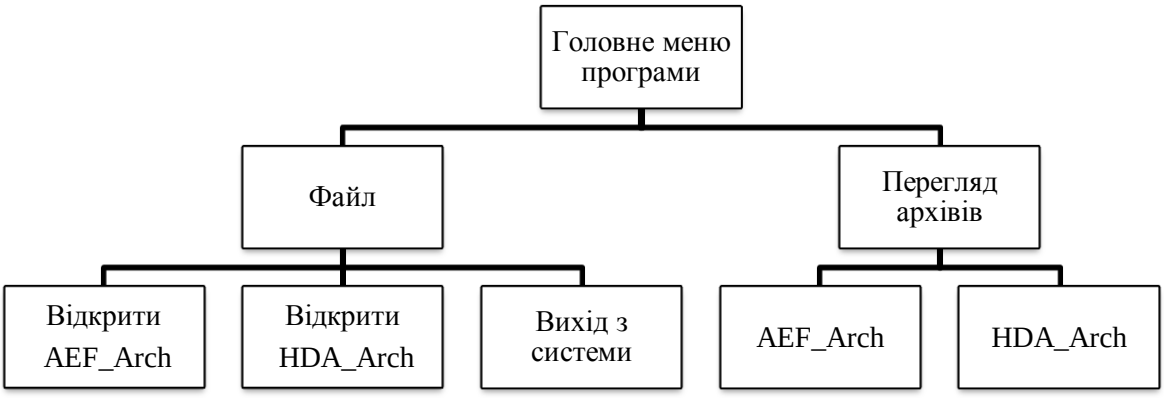


Рисунок 2.1 - Структурна схема програми

2.2 Розробка екранного інтерфейсу програми

Екранний інтерфейс програми багато в чому визначає зручність роботи користувача і є одним з важливих факторів, що впливають на ефективність його праці. Програма, що виконує всі покладені на неї функції, що володіє високою швидкодією може бути повністю непридатною для роботи через неприйнятний інтерфейс з користувачем.

Найбільш практичними і зручними з точки зору користувача можна вважати системи, що мають екранний інтерфейс, побудований на основі системи спливаючих меню. Відповідно до завдання (див. 1.4.4), при розробці даного дипломного проекту був використаний вказаний вище підхід.

2.2.1 Особливості середовища розробки

Відповідно до завдання (див. 1.4.10), проект розроблений в середовищі програмування Qt Designer.

Проекти, розроблені в даному середовищі, функціонують без зміни коду як під управлінням ОС Linux, так і під управлінням ОС Windows.

Система проектування Qt Designer надає розробнику досить велику гаму засобів, що дозволяють здійснювати розробку екранного інтерфейсу [4].

Зокрема, є можливість створення стандартного Головного Вікна (Qt Designer \ New \ Main Window), зі стандартним меню, яке в подальшому можна редагувати.

Необхідно відзначити особливості проектування екранного меню в даній системі.

В Qt Designer є кілька панелей, що відповідають за маніпуляцію тими чи іншими засобами програмування в середовищі. Зокрема, для створення пункту меню необхідно використовувати Action Editor, в якому створюється новий об'єкт, Action, є аналогом «активатора події». Далі зазначений об'єкт з вкладки Action Editor «перетягується» мишею на робочу форму, де і встановлюється в потрібному місці MenuBar. Під час роботи програми, активація пункту меню викликає на виконання обробник події - slot. Слід також зазначити, що для того, щоб створити зв'язок між активацією пункту меню і виконанням відповідної процедури (слота), використовується спеціальний механізм, властивий даному середовищі програмування - Connection. На спеціальній вкладці створюється «з'єднання», в якому Sender, в даному випадку - об'єкт Action, посилає Signal «одержувачу» (Receiver). Одержувачем в даному випадку є об'єкт Main Form, всередині якого визначено вказаний слот. Після того, як сигнал випущений (emit), він приймається одержувачем (в загальному випадку відправник - екземпляр будь-якого класу, що містить сигнали, а одержувач -

об'єкт, що містить сигнали або слоти) і на виконання запускається вказаний слот. Пов'язані сигнали і слоти повинні мати однакові параметри. В даному випадку слід зазначити, що сигнал, що випускається об'єктом Action, називається activated() і визначено всередині класу QAction. Середовище програмування Qt Designer має досить невеликий набір візуальних компонент, розташованих на спеціальній вкладці Toolbox. На даній вкладці є 8 панелей, в яких містяться 45 компонент, які, вибравши на панелі і клацнувши мишкою в потрібному місці форми, можна додати в свою програму. При цьому зазначеним об'єктам в Property Editor розробником встановлюються різні властивості. Додані на форму з палітри компоненти визначаються в проекті як вказівники на об'єкти відповідних типів.

2.2.2 Компоненти графічного інтерфейсу програми

У загальному випадку, якщо необхідні візуальні компоненти графічного інтерфейсу відсутні в зазначених у підрозділі 2.2.1 панелях інструментів, об'єкти створюються за допомогою оператора new.

У таблиці 2.1 наведені візуальні компоненти, використані для створення графічного інтерфейсу розробленої проектом програми.

Таблиця 2.1 - Візуальні компоненти графічного інтерфейсу

Палітра компонент	Клас	Ім'я об'єкту	Призначення у проекті
1	2	3	4
Containers	QWidgetStack	AEF_HDA_WidgetStack	Об'єкт, що дозволив створити двохсторінковий інтерфейс зі згрупованої інформацією про кожний архів на окремій сторінці
-	QWidget	AEF_StackPage	«Сторінка» AEF_Архіву
-	QWidget	HDA_StackPage	«Сторінка» HDA_Архіву
Display	QLabel	textLabel5	Об'єкт, що містить напис «AEF_Arch»
Display	QTextBrowser	textBrr3	Віконце виведення поточної інформації про файли AEF_Архіву
Views	QTable	table1	Таблиця ***_Event.aef
Views	QTable	table2	Таблиця Event_Common.aef
Display	QLabel	textLabel1	Об'єкт, що містить напис «HDA_Arch»
Display	QLabel	textLab2	Об'єкт, що містить напис над віконцем виведення інформації про заголовки файлів HDA_Архіву
Display	QLabel	textLab5	Об'єкт, що містить напис над таблицями HDA_Архіву

Продовження табл. 2.1

1	2	3	4
Display	QLabel	textLab6	Об'єкт, що містить напис над віконцем виведення поточної інформації про файли HDA_Архіву
Display	QTextBrowser	textBrr4	Віконце виведення інформації про заголовки файлів HDA_Архіву
Display	QTextBrowser	textBrr5	Віконце виведення поточної інформації про файли HDA_Архіву
Views	QTable	table3	Таблиця ***_Value.hda
Views	QTable	table4	Таблиця Source_Common.hda
Display	QLabel	textLab4	Об'єкт, що містить напис над таблицями AEF_Архіву
Display	QLabel	textLab1	Об'єкт, що містить напис над віконцем виведення поточної інформації про файли AEF_Архіву
Display	QTextBrowser	textBrr1	Віконце виведення інформації про заголовки файлів AEF_Архіву
Display	QLabel	textLab3	Об'єкт, що містить напис над віконцем виведення інформації про заголовки файлів AEF_Архіву

2.3 Опис глобальних змінних

У таблиці 2.2 наведена інформація про всі глобальні змінні проекту.

Таблиця 2.2 - Глобальні змінні проекту

Тип та ім'я змінної	Опис
1	2
tag_TDA_HDA_HeadFV* fileHead-SCom	Вказівник на структуру заголовку файлу зберігання даних Source_Common.hda
tag_TDA_HDA_DataHeadFV* pHeadBuf_HDA	Вказівник на структуру заголовку файлу зберігання даних ********_Value.hda
bool showLinkedData_HDA	Змінна, яка містить інформацію про завантаження пов'язаних даних (HDA)
bool showLinkedData	Змінна, яка містить інформацію про завантаження пов'язаних даних (AEF)
tag_TDA_AEF_HeadFV* pHead-BufEvCom	Вказівник на структуру заголовку файлу зберігання даних Event_Common.aef
QString TempForChar	Рядок для тимчасового розміщення інформації з полів структур даних
tag_TDA_AEF_DataHeadFV* fileHead	Вказівник на структуру заголовку файлу зберігання даних ********_Event.aef
int setCurs1	Змінна для установки позиції курсору для виведення повідомлення
int insPos1	Змінна для вставки параграфа повідомлення в поточну позицію
int hourseek	Змінна, яка отримує значення поля структури tag_TDA_AEF_DataHeadFV - SeekFileOfUTCTime[24]

Продовження табл. 2.2

1	2
int setCurs3	Змінна для установки позиції курсору для виведення повідомлення
int insPos3	Змінна для вставки параграфа повідомлення в поточну позицію
QString fileName	Змінна для зберігання імені файлу, що відкривається
tag_TDA_AEF_DataHeadFV * pHeadBuf	Вказівник на структуру заголовку файлу зберігання даних ********_Event.aef
tag_TDA_AEF_HeadFV* file-HeadEvCom	Вказівник на структуру заголовку файлу зберігання даних Event_Common.aef
int insPos4	Змінна для вставки параграфа повідомлення в поточну позицію
int setCurs4	Змінна для установки позиції курсору для виведення повідомлення
int insPos5	Змінна для вставки параграфа повідомлення в поточну позицію
int setCurs5	Змінна для установки позиції курсору для виведення повідомлення
tag_TDA_HDA_DataHeadFV* file-Head_HDA	Вказівник на структуру заголовку файлу зберігання даних ********_Value.hda
tag_TDA_HDA_HeadFV* pHeadBuf-SCom	Вказівник на структуру заголовку файлу зберігання даних Source_Common.hda
unsigned long bufReadCount	Змінна, що зберігає кількість прочитаних записів з файлу
unsigned long bufMaxCount	Змінна, що зберігає максимальну кількість записів, які можуть бути прочитані з файлу
char inname [255]	Масив зберігання імені файлу, що відкривається
char prtime[255]	Масив зберігання інформації з полів структур даних
char prtime1[255]	Масив зберігання інформації з полів структур даних
char prtime2[255]	Масив зберігання інформації з полів структур даних
char prtime3[255]	Масив зберігання інформації з полів структур даних
char prtime4[255]	Масив зберігання інформації з полів структур даних
char DataSource [80]	Масив зберігання інформації з полів структур даних
char MessageStat [255]	Масив зберігання інформації з полів структур даних
char nameEvCom[]	Масив, що містить ім'я файлу даних із пов'язаною з основним файлом інформацією
int kod	Змінна, яка містить код повідомлення
char nameSCom[]	Масив, що містить ім'я файлу даних із пов'язаною з основним файлом інформацією
char descrServ[80]	Масив зберігання інформації з полів структур даних
char nameServ[80]	Масив зберігання інформації з полів структур даних
char nameUnit[80]	Масив зберігання інформації з полів структур даних
QString qstringtmp[10]	Масив рядків для розміщення розкодованої інформації з файлів
int currqstr	Змінна, яка містить індекс рядка в масиві елементів QString

2.4 Розробка алгоритмів функцій і слотів

При проектуванні даної системи весь процес був розбитий на кілька частин:

– проектування меню програми;

– створення функцій і слотів екранного інтерфейсу;

– розробка функцій і слотів, за допомогою яких вирішуються всі завдання, які стоять перед проектом;

– підключення до проекту необхідних бібліотек;

– комплексне налагодження та тестування програми в процесі дослідної експлуатації.

Створення в першу чергу процедур і функцій меню і екранного інтерфейсу пояснюється тим, що ці функції складають «скелет» програми, до якого в подальшому були підключені інші процедури і функції. Таким чином, велася розробка методом «З гори до низу». Цей метод дозволяє за короткий термін отримати працюючу систему з обмеженим (і в процесі роботи все повнішим) набором функціональних можливостей.

2.4.1 Розробка інтерфейсних функцій і слотів

В даному пункті описані розроблені проектом функції і слоти, які використовуються для управління елементами графічного інтерфейсу програми:

– активації пунктів меню;

– виклику діалогового вікна відкриття файлів;

– виведення текстових повідомлень в віконця виведення;

– приведення таблиць до вихідного стану («очищення» таблиць при відкритті декількох файлів).

2.4.1.1 Слоти виклику на екран певної сторінки додатка

Відповідно до завдання (див. 1.4.5.1), розроблені два слоти, призначені для виклику на екран певної сторінки додатка.

У тілі слотів відбувається звернення до вказівника на об'єкт класу QWidgetStack, розміщеного на формі і виклик певного в даному класі слота raiseWidget (int), в якості параметра до якого вказується ціле число, відповідне властивості об'єкта - currentPage. У першому випадку (AEF_Arch) вказується 0, а в другому (HDA_Arch) - 1.

2.4.1.2 Слот перемикання сторінок додатку

Відповідно до ТЗ (дивись 1.4.5.1), на даний слот покладається завдання виклику на екран відповідної сторінки додатка при активації відповідного пункту меню. Як параметр слот отримує вказівник на об'єкт типу QAction.

Коли відбувається вибір якогось пункту меню, то відповідний об'єкт типу QAction випускає (emit) сигнал selected (...), і в якості параметра передає покажчик на себе. Слот приймає як параметр цей покажчик і аналізує об'єкт, на який вказує покажчик. Залежно від результатів аналізу «викликається» відповідна сторінка додатка.

Зауваження. У об'єкта QWidgetStack може бути кілька «сторінок» (об'єк-тів класу QWidget), але тільки одна з них може бути промальована на формі. Для цього її потрібно зробити об'єктом верхнього рівня, тобто викликати слот raiseWidget (int) с відповідним номером сторінки в якості параметра.

На рис. 2.2 наводиться схема алгоритму даного слоту

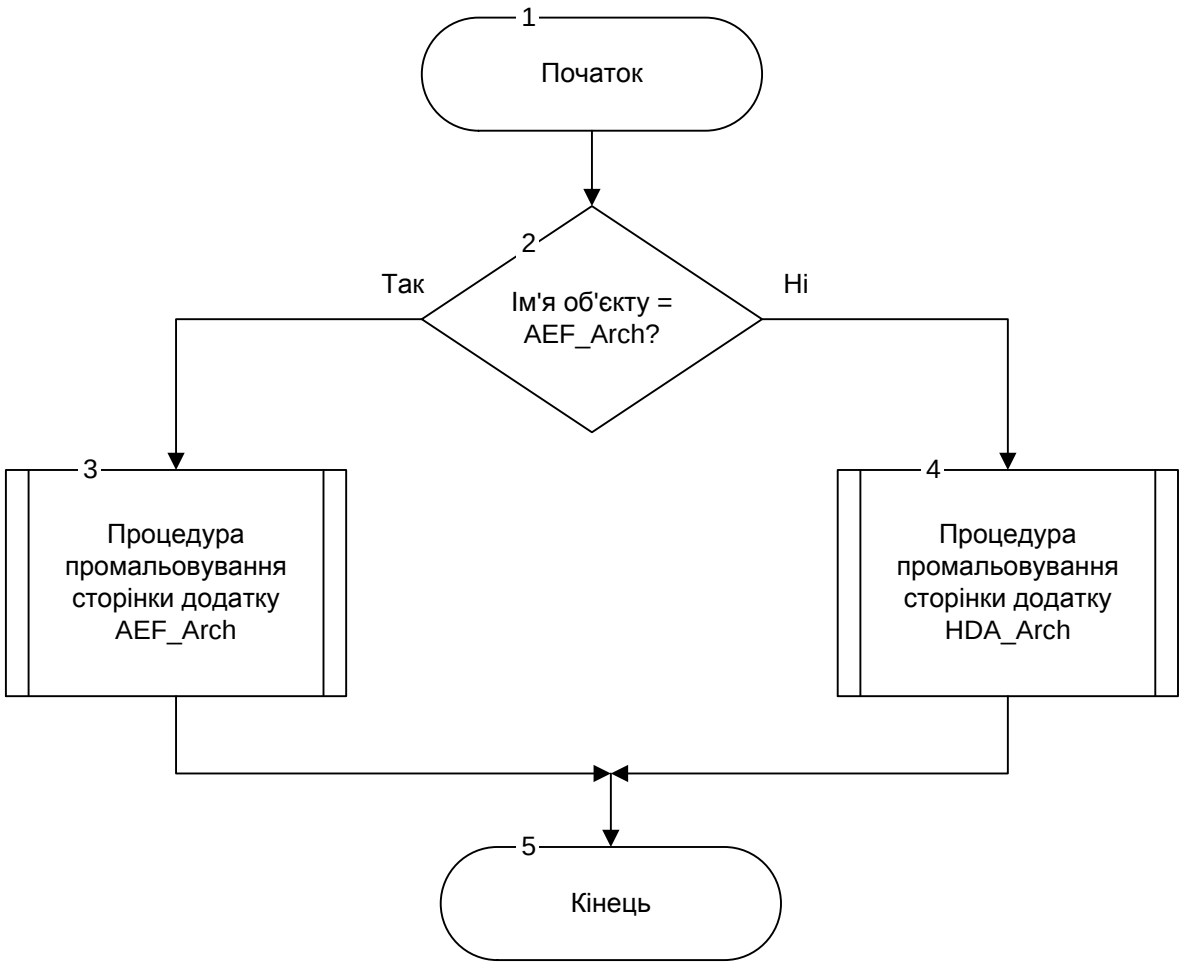


Рисунок 2.2 - Блок-схема алгоритму функціонування слоту, виклику відповідної сторінки додатку

2.4.1.3 Слоти відкриття файлів

Відповідно до завдання (див. 1.4.5.2), були створені два слота, в тілі яких відбувається виклик стандартного діалогового вікна відкриття файлів. Це завдання реалізується за допомогою виклику функції-члена класу QFileDialog getOpenFileName(...). Після вдалого відкриття файлу здійснюється виклик слота завантаження файлу, який буде описаний в наступних підрозділах. На рис. 2.3 приведена схема алгоритму функціонування слота відкриття файлів.

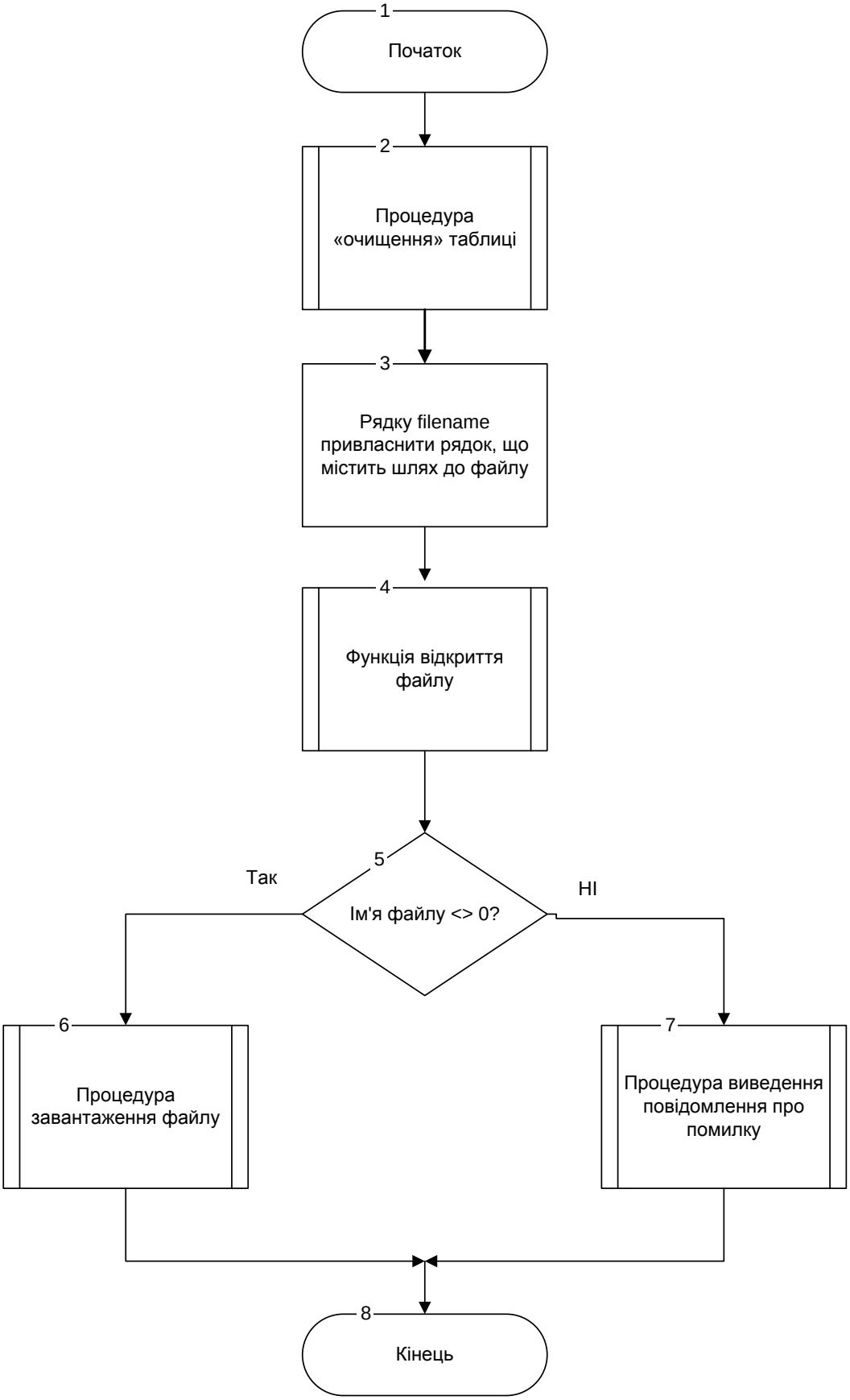


Рисунок 2.3 - Схема алгоритму функціонування слота відкриття файлів

2.4.1.4 Функції виводу поточних повідомлень і повідомлень про помилки аналізу заголовків файлів

Відповідно до завдання (див. 1.4.5.5), проектом розроблено ряд функцій, які виконують виведення повідомлень в віконця виведення.

Функція виведення повідомлення про відсутність пов'язаних даних виводить в віконце виведення повідомлення: «Зміни, пов'язані дані відсутні».

Функція виведення поточних повідомлень по ходу аналізу основних файлів виводить в віконце виведення поточної інформації повідомлення про хід процесу аналізу відкриваються файлів. Як параметр дана функція отримує ціле число, якому ставиться у відповідність певний повідомлення, яке виводиться на екран. Розроблено дві функції для обох архівів. Функції мають подібні схеми алгоритмів.

На рис. 2.4 приведена схема алгоритму функції виведення поточних повідомлень по ходу аналізу основних файлів.

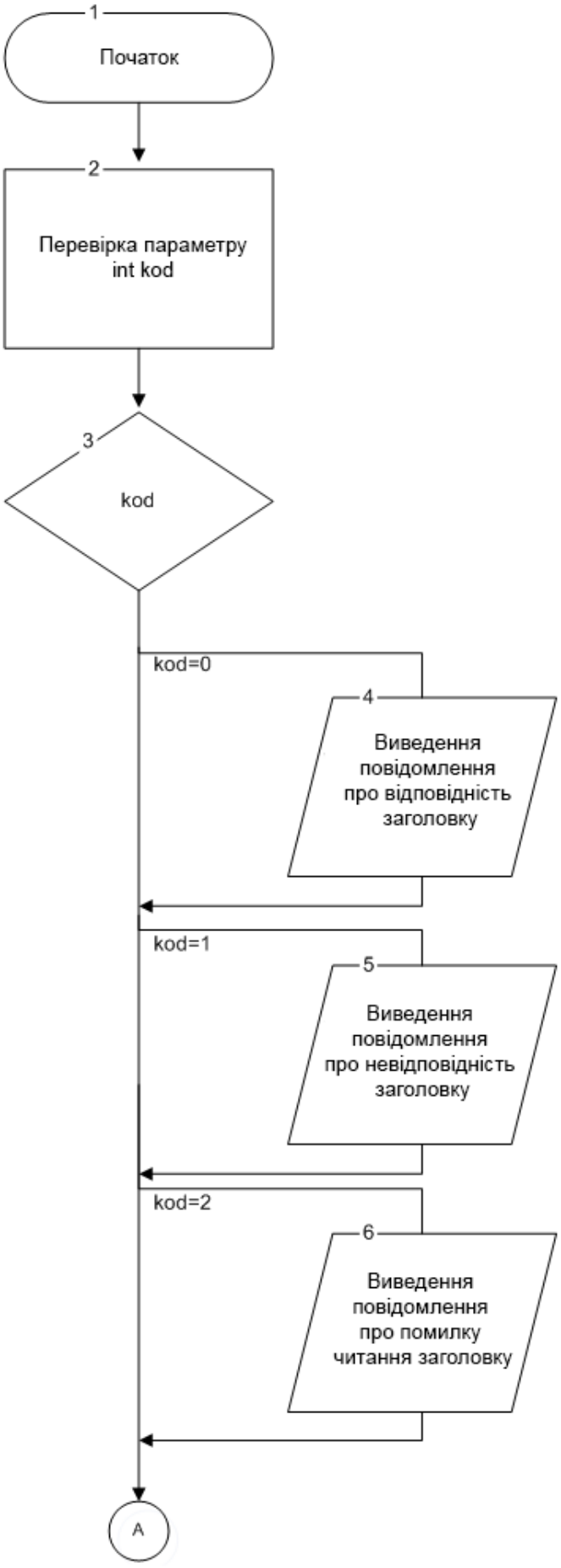


Рисунок 2.4 – Схема алгоритму функції виведення поточних повідомлень

Рисунок 2.4. Аркуш 2

Рисунок 2.4. Аркуш 3

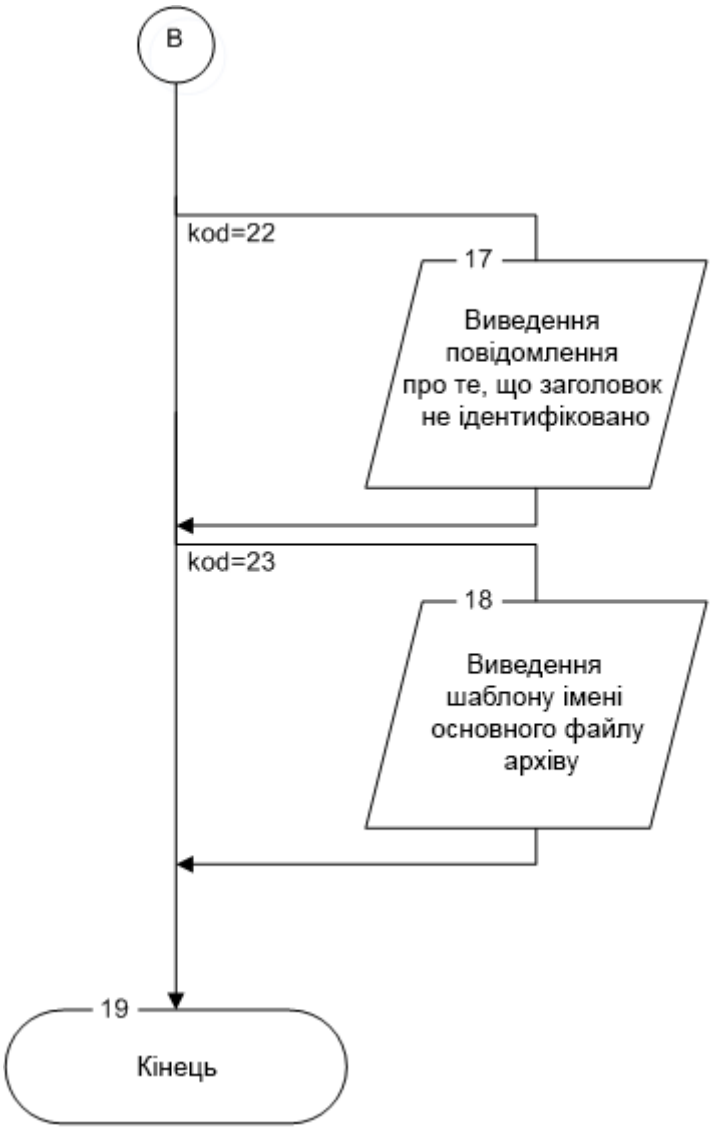


Рисунок 2.4. Аркуш 4

Наступна функція, функція виведення повідомлення про помилки, що виникли по ходу аналізу заголовку файлу, за своєю структурою не відрізняється від наведеної вище. Проектом розроблено дві подібних функції для аналізу заголовків файлів обох архівів.

На рис. 2.5 приведена схема алгоритму функції виведення повідомлення про помилки, що виникли по ходу аналізу заголовку файлу.

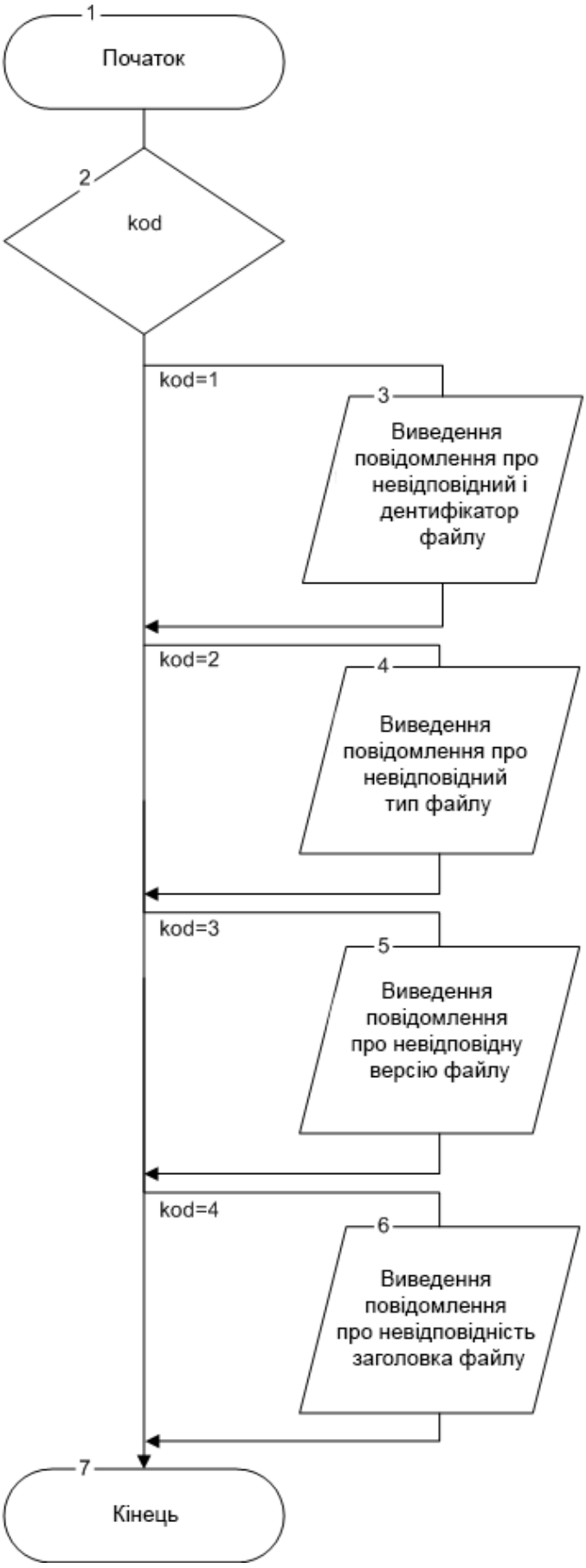


Рисунок 2.5 – Схема алгоритму функції виведення повідомлень при помилці заголовка файлу

2.4.1.5 Функції приведення таблиць в початковий стан

Відповідно до завдання (див. 1.4.4), для виведення на екран вмісту файлів в програмі використовувалися компоненти типу QTable (таблиці).

У певний момент роботи програми таблиці «очищаються». І з цією метою проектом розроблено чотири функції (див. 1.4.5.1).

Їх алгоритми функціонування подібні. На рис. 2.6 приведена одна схема алгоритму функції приведення таблиць в початковий стан.

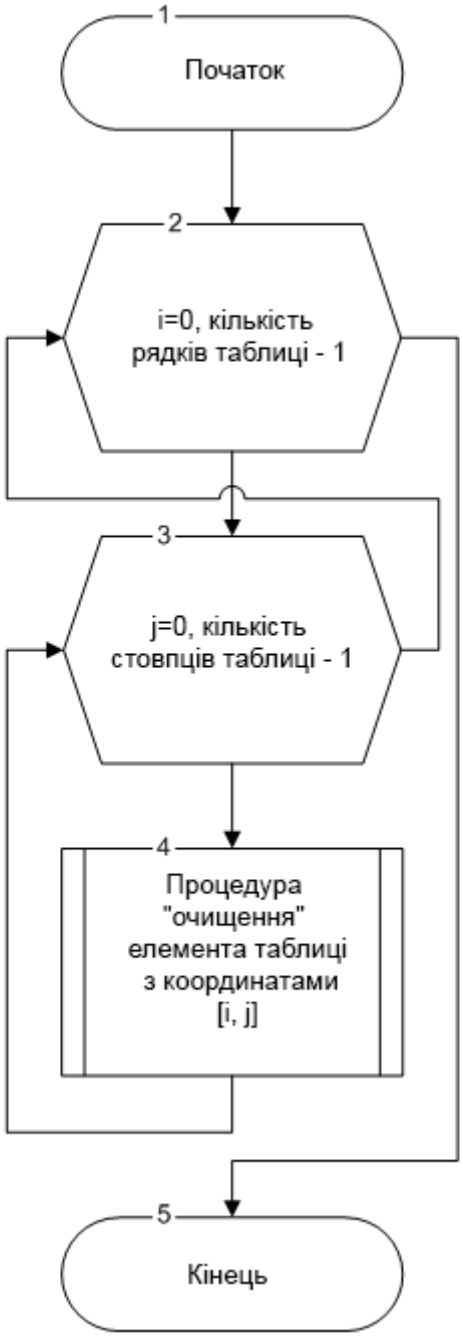


Рисунок 2.6 – Схема алгоритму приведення таблиць в початковий стан

2.4.1.6 Слот закриття програми

Слот розроблений відповідно до завдання (див. 1.4.5.1).

Слот, за допомогою механізму Connection, пов'язаний з сигналом activated() від action з ім'ям fileExitAction. Під час роботи програми, після активації кнопки меню «Вихід», викликається член-функція класу QApplication - exit(0) і відбувається завершення програми з кодом 0 (успішне завершення).

2.4.2 Розробка основних і допоміжних функцій і слотів

У цій частині пояснювальної записки наводиться опис та схеми алгоритмів основних й допоміжних функцій і слотів, розроблених проектом для вирішення поставлених завдань.

2.4.2.1 Приватна функція-конструктор init()

У середовищі Qt Designer є можливість виконання певних дій до того, як ще не створена графічна форма додатка. Зазвичай такими діями є різні присвоєння значень глобальних змінних і завдання параметрів створюваних об'єктів.

Відповідно до завдання (див. 1.4.6.1), проектом розроблена функція, що має стандартне ім'я init() і визначається системою як конструктор.

2.4.2.2 Функції завантаження файлів

Відповідно до завдання (див. 1.4.5.2), розроблені дві функції завантаження файлів для обох архівів. Функції розроблені для надання програмі читабельності, тобто за допомогою даних функцій в програмі проглядається послідовність і логіка дій. Функції корисні при модифікації програми.

На рис. 2.7 представлена схема алгоритму функцій завантаження файлів.

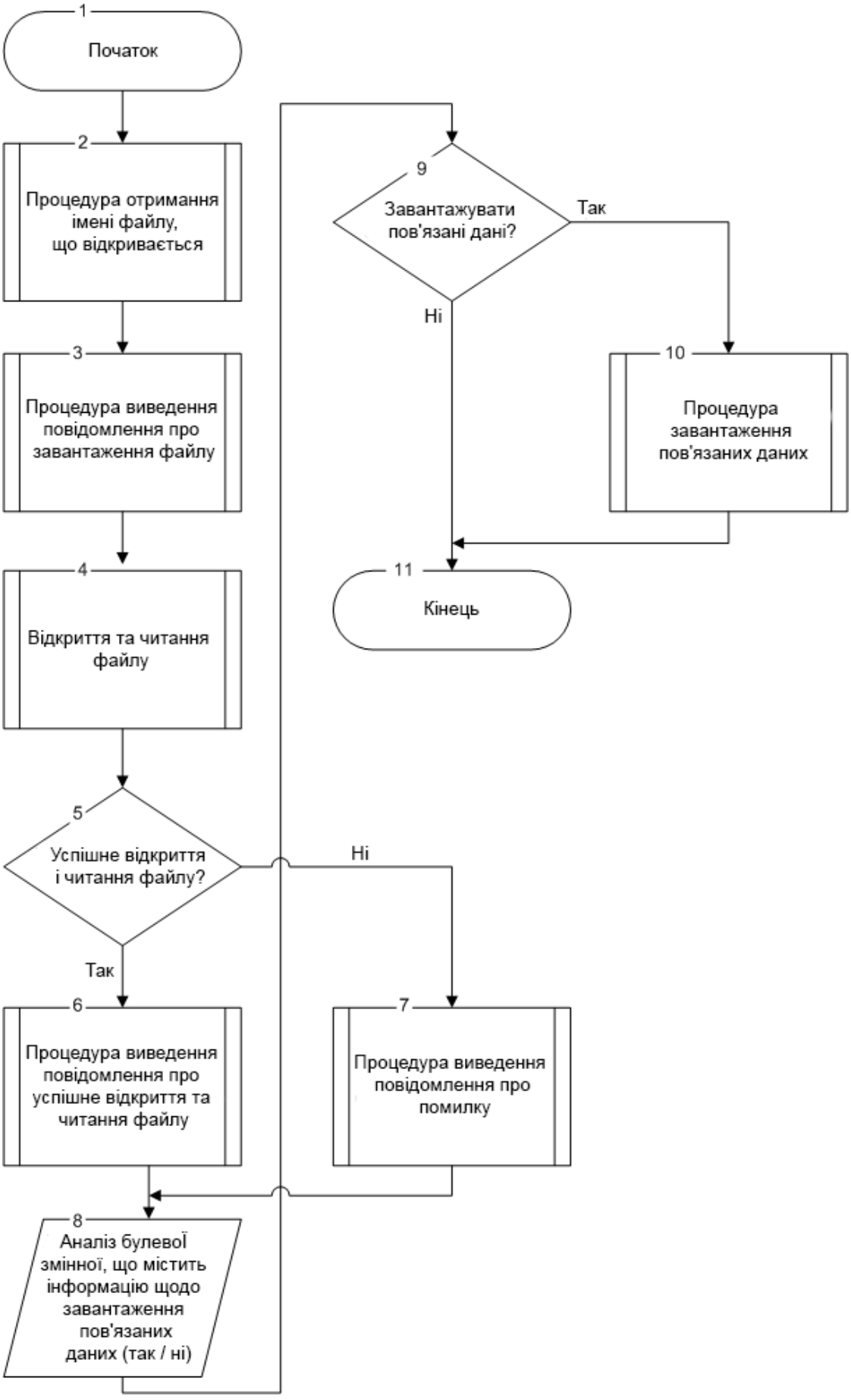


Рисунок 2.7 – Схема алгоритму функції завантаження файлів

2.4.2.3 Функції читання основних файлів

Відповідно до завдання (див. 1.4.5.4), проектом розроблено дві функції, призначені для читання даних з основних файлів архівів.

Після того, як за допомогою процедури аналізу заголовку файлу (див. 2.4.2.4) проаналізовано заголовок, в разі відповідності - дані зчитуються з файлу, декодуються і записуються в осередки таблиці. У разі невідповідності - здійснюється вихід з функції з кодом 1.

Алгоритм функціонування даних функцій наведено на рис. 2.8.

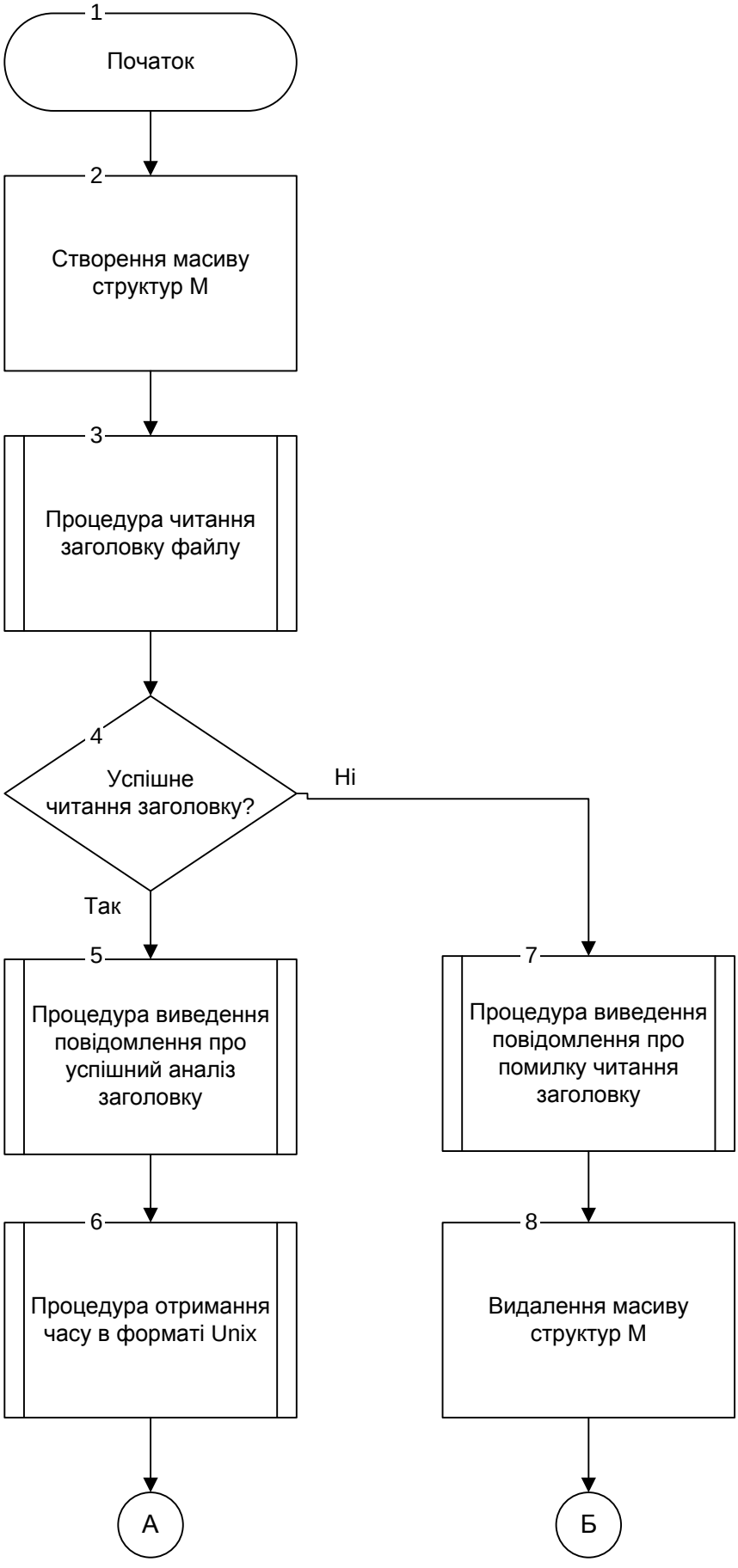


Рисунок 2.8 – Схема алгоритму функції читання основних файлів

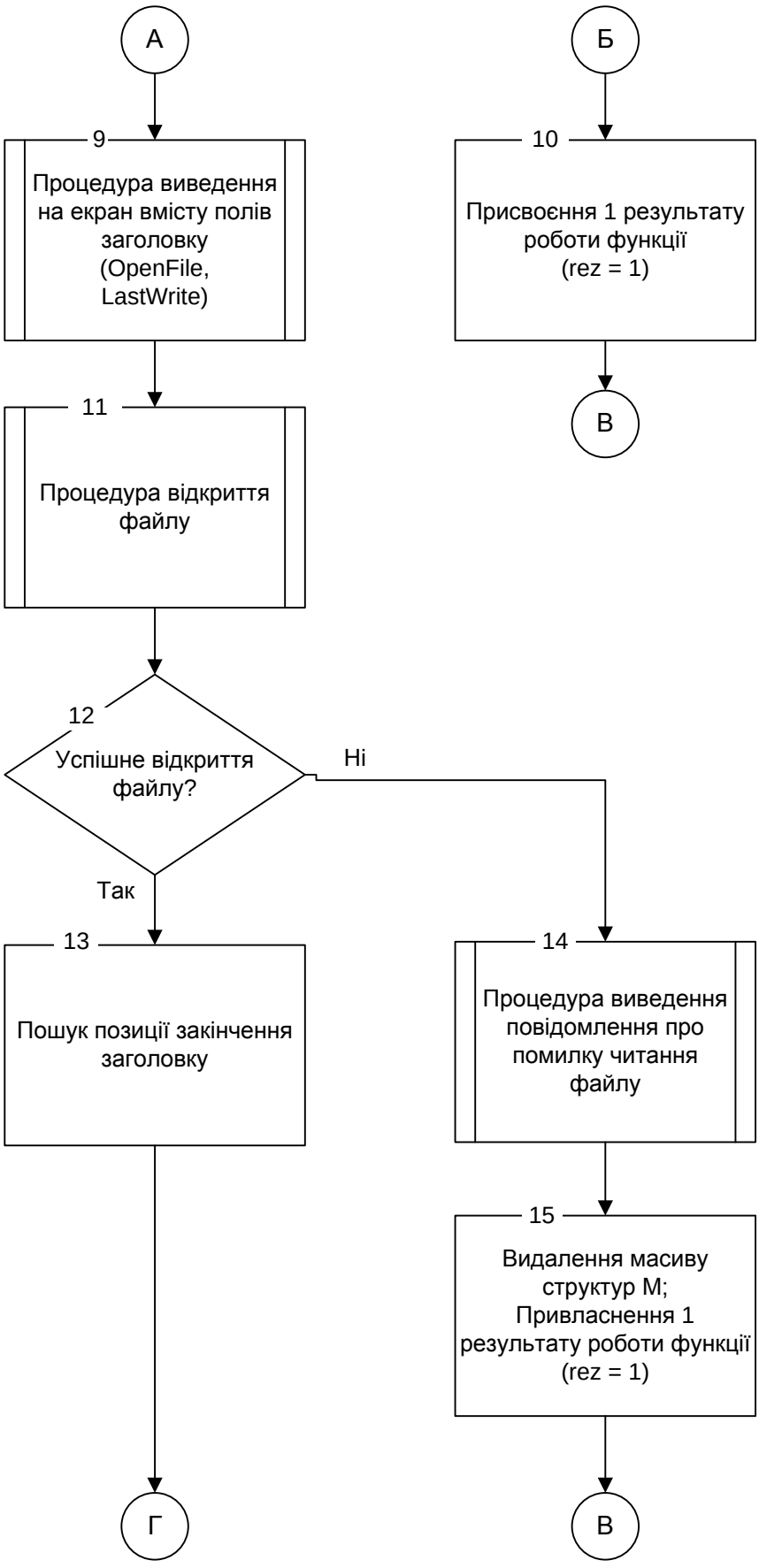


Рисунок 2.8. Аркуш 2

Рисунок 2.8. Аркуш 3

2.4.2.4 Функції читання та ідентифікації заголовків основних файлів

Відповідно до завдання (див. 1.4.5.3), проектом розроблено дві функції, призначені для ідентифікації заголовків основних файлів. Вміст полів заголовку перевіряється на відповідність певним значенням. У разі невідповідності - на екран виводиться повідомлення про помилку, яке вказує неправильне значення поля заголовку файлу. Так само в віконце виведення видається еталонне значення для даного поля.

На рис. 2.9 приведена схема алгоритму функції читання та ідентифікації заголовку основного файлу.

Рисунок 2.9 – Схема алгоритму функції читання та ідентифікації заголовку основного файлу

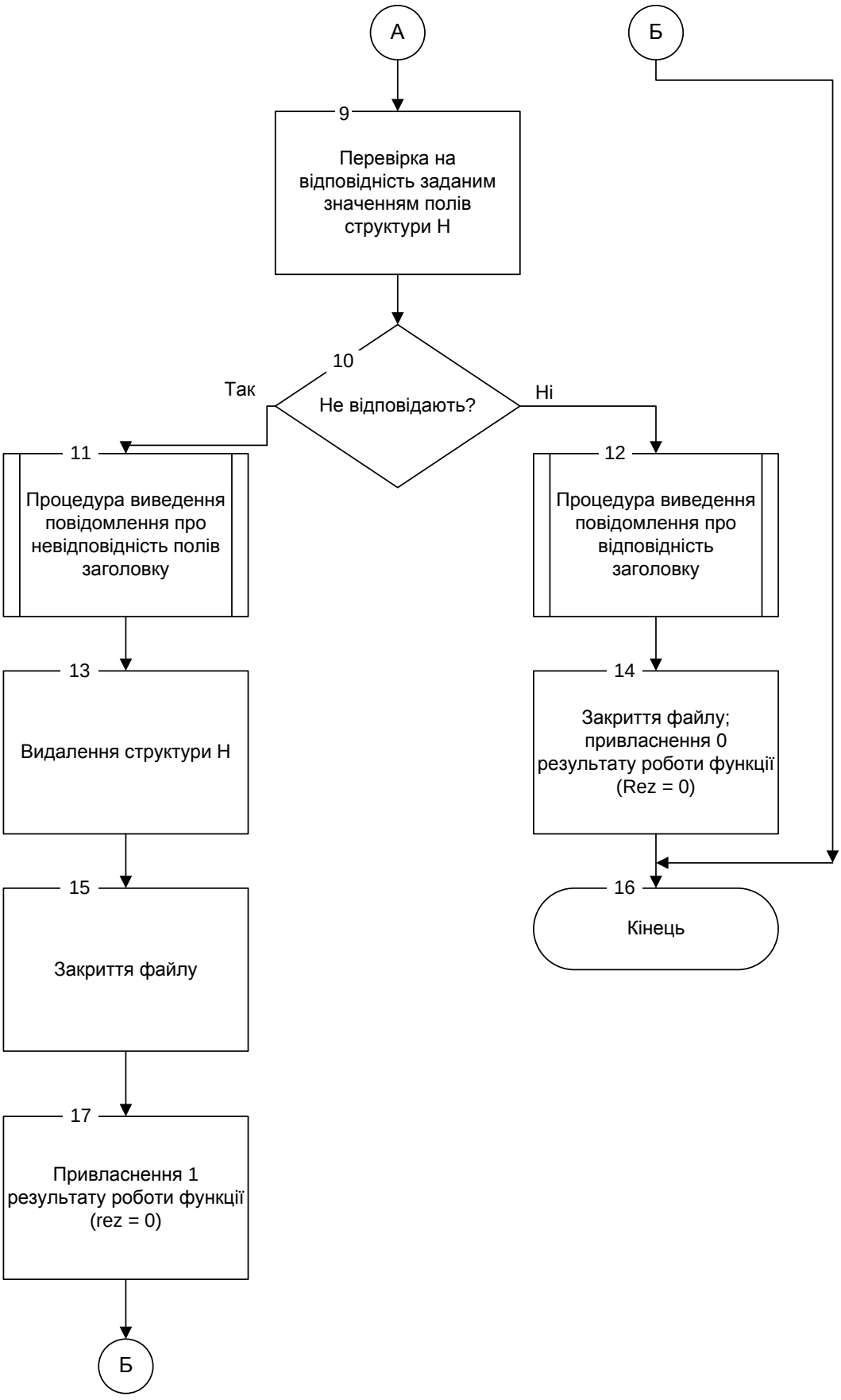


Рисунок 2.9. Аркуш 2

2.4.2.5 Функції читання пов'язаних файлів

Відповідно до завдання (див. 1.4.5.4), розроблені дві функції читання пов'язаних файлів.

На рис. 2.10 наведена схема алгоритму функцій читання пов'язаних файлів.

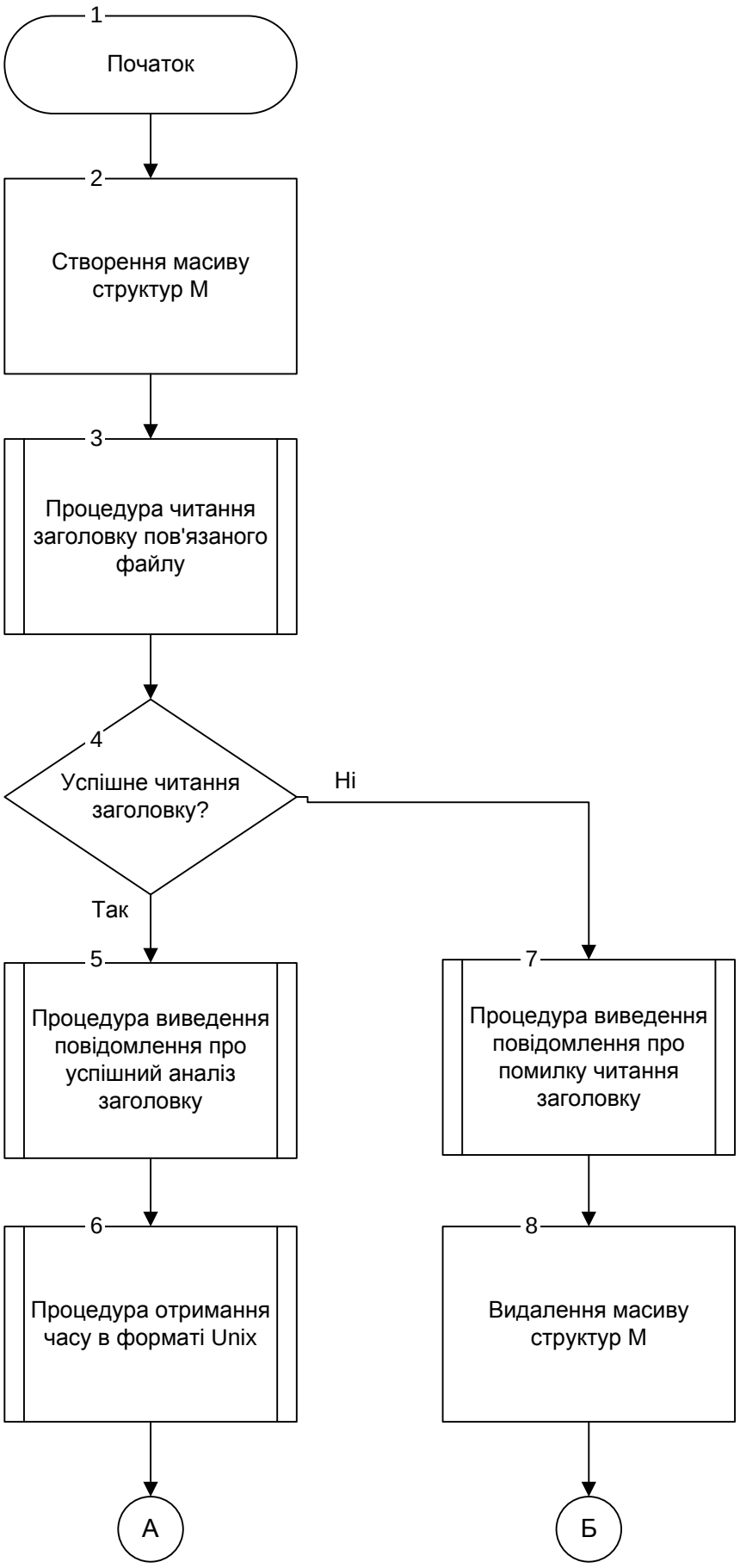


Рисунок 2.10 – Схема алгоритму функцій читання пов'язаних файлів

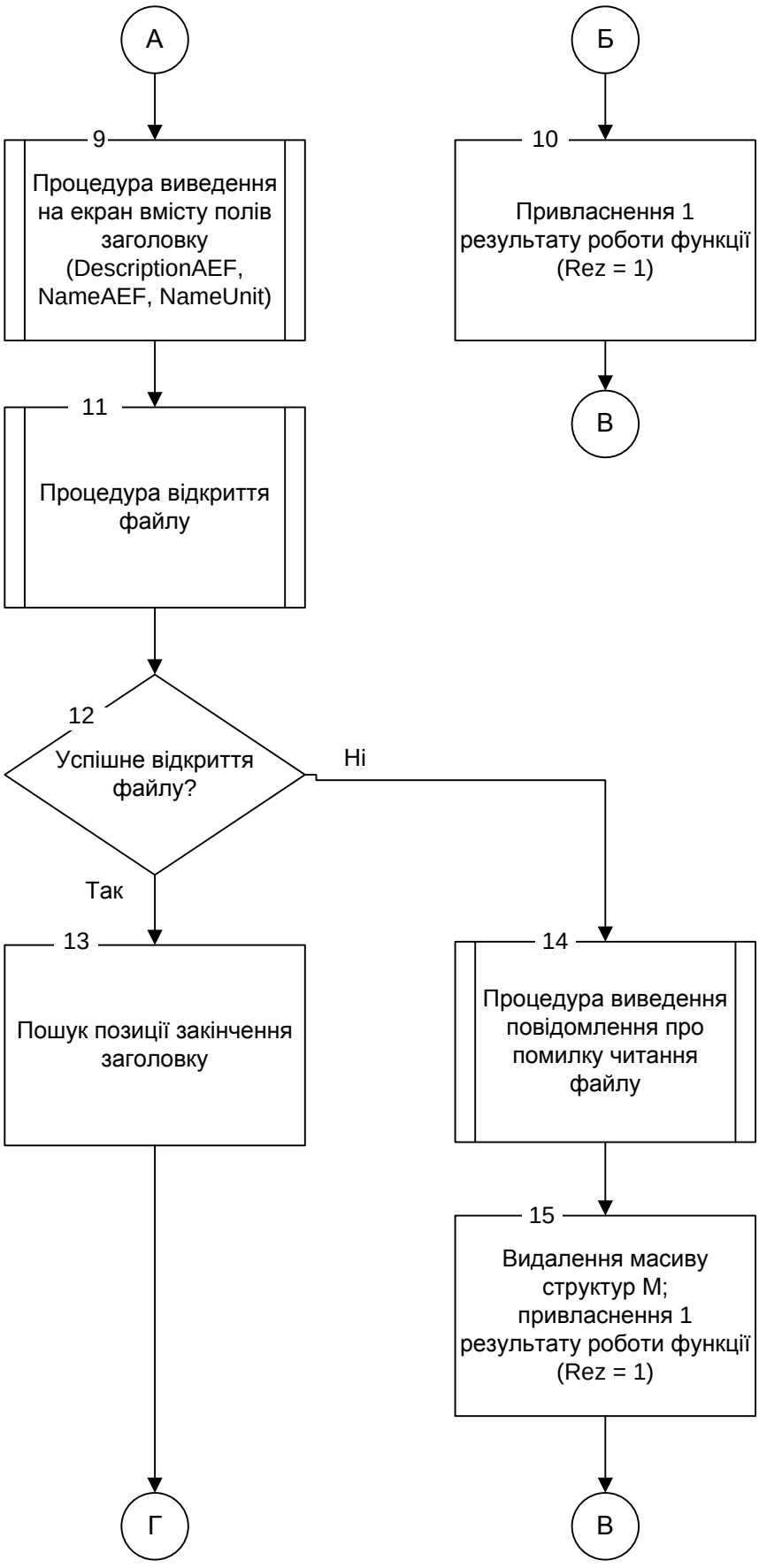


Рисунок 2.10. Аркуш 2

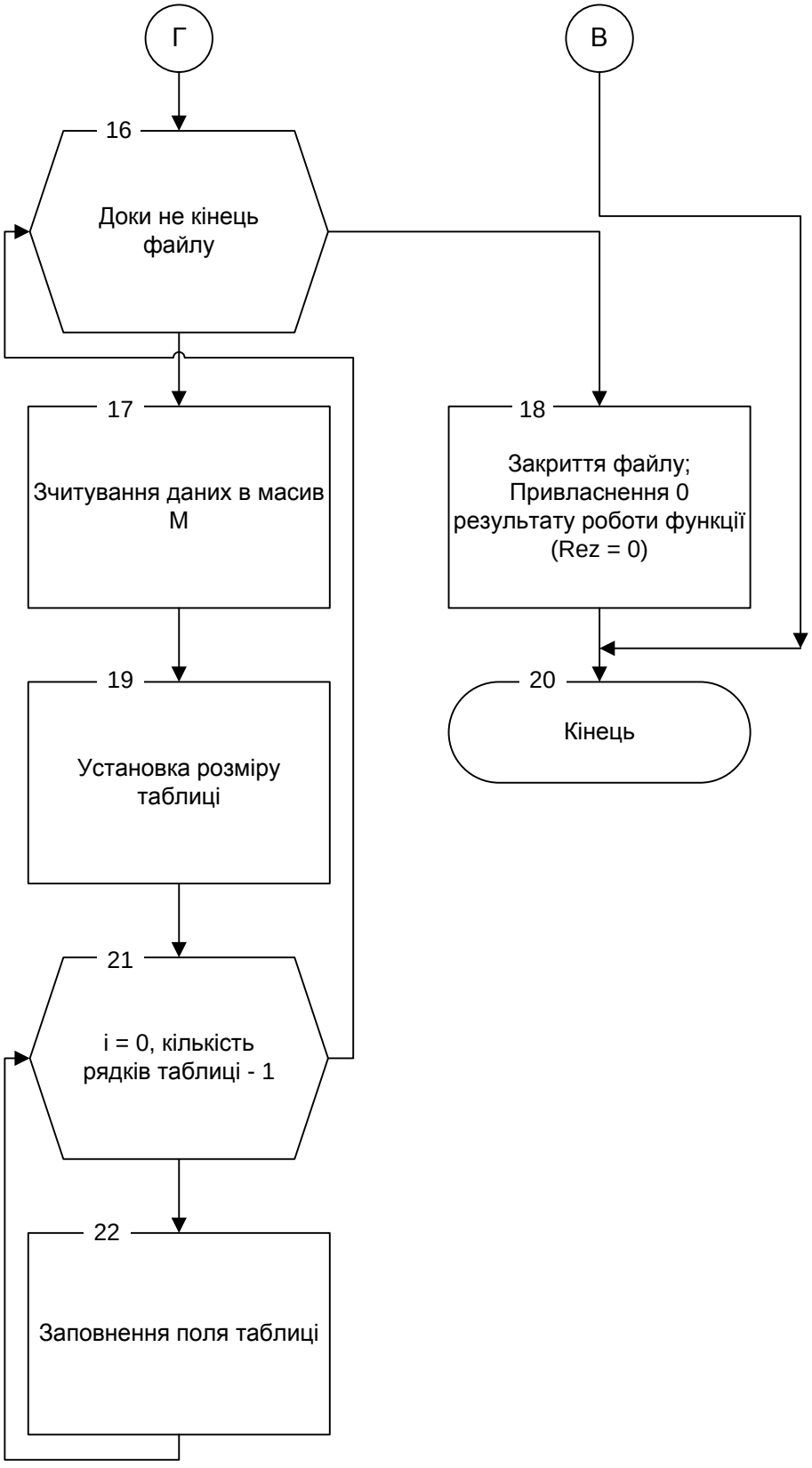


Рисунок 2.10. Аркуш 3

2.4.2.6 Функції читання та ідентифікації заголовків пов'язаних файлів

Відповідно до завдання (див. 1.4.5.3), розроблені дві функції для роботи з обома архівами.

На рис. 2.11 наведена схема алгоритму функціонування зазначених функцій.

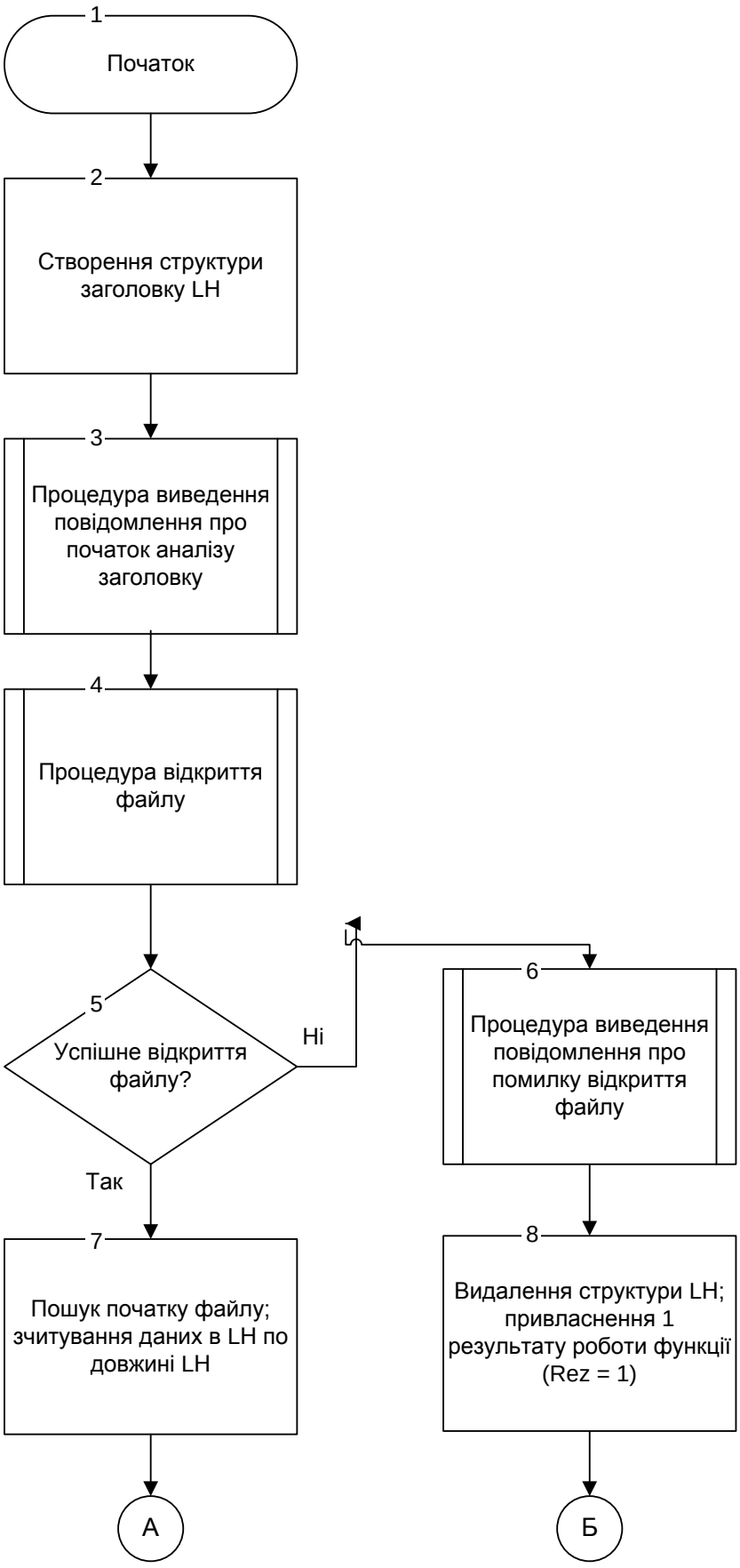


Рисунок 2.11 - Схема алгоритму функції читання та ідентифікації заголовків пов'язаних файлів

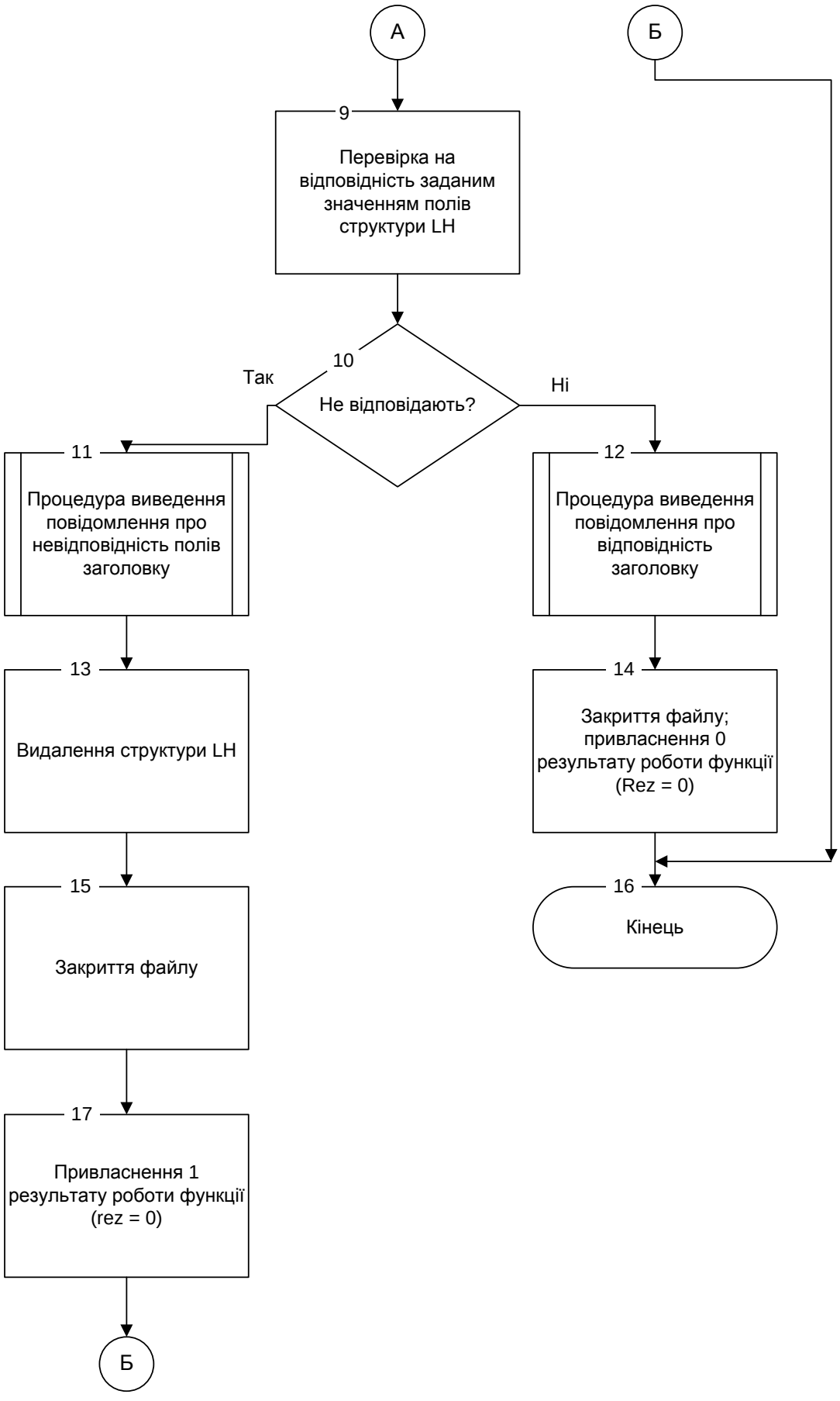


Рисунок 2.11. Аркуш 2

2.4.2.7 Функції завантаження пов'язаних даних

Відповідно до завдання (див. 1.4.5.2), розроблені дві функції завантаження пов'язаних даних.

Ці функції слугують оболонкою для об'єднання в логічне ціле дій, пов'язаних з ідентифікацією та виведенням інформації з пов'язаних файлів разом з основним файлом. Функції слугують підвищенню читабельність програми.

На рис. 2.12 наведена схема алгоритму функціонування зазначених функцій.

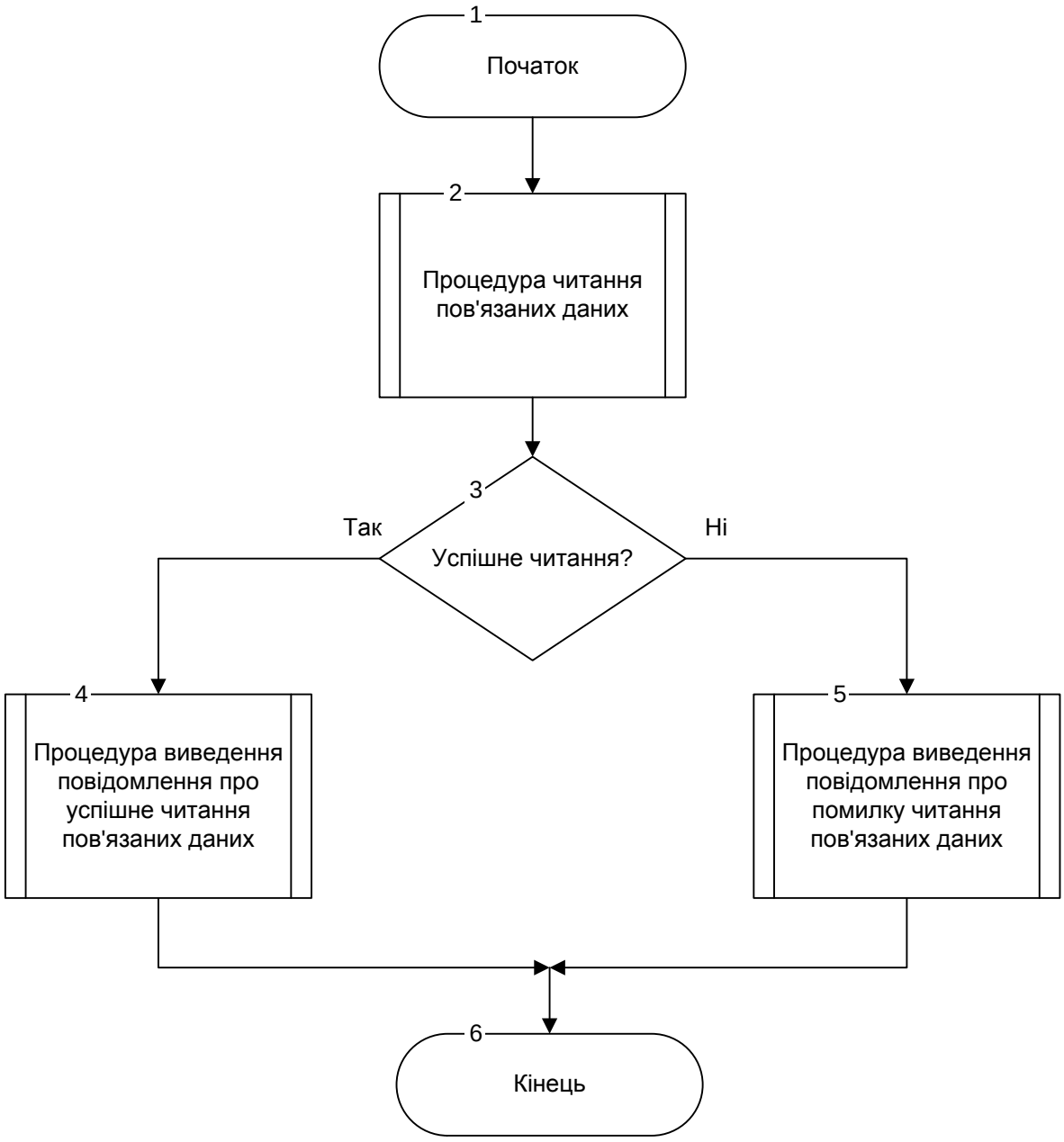


Рисунок 2.12 - Схема алгоритму функції завантаження пов'язаних даних

2.4.2.8 Слоти відображення пов'язаних даних

Відповідно до завдання (див. 1.4.5.4), проектом розроблено два слота для відображення пов'язаних даних.

Зазначені слоти викликаються на виконання, коли відбувається виділення якого-небудь осередку в основній таблиці або переміщення по рядках таблиці. Слід нагадати, що відповідно до завдання (див. 1.4.4), на кожній сторінці додатка розміщено по дві таблиці: одна для відображення даних з основного файлу, а інша - для даних зі зв'язаного файлу.

Також слід зазначити, що для кожного з зазначених слотів в середовищі проектування створено по два з'єднання (connection), що зв'язують сигнал головної таблиці (clicked(...) - натискання на будь-яку клітинку) з даними слотом і сигнал головної таблиці (currentChanged(...) - зміна поточної осередки таблиці) з даними слотом. Зазначені сигнали визначені в класі QTable.

На рис. 2.13 зображена схема алгоритму функціонування слота відображення на екрані пов'язаних даних.

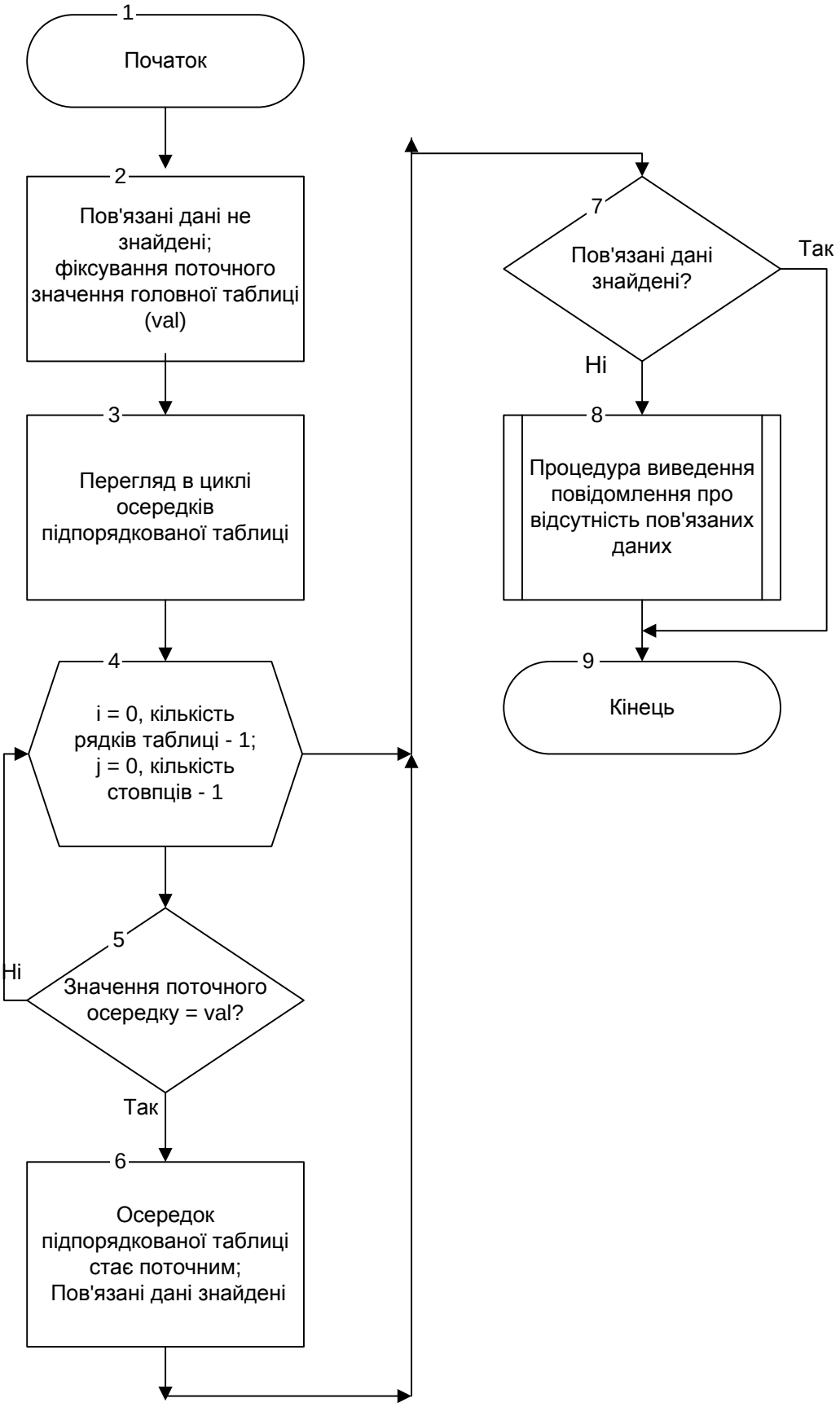


Рисунок 2.13 - Схема алгоритму слота відображення на екрані пов'язаних даних

2.4.2.9 Функція копіювання об'єкта unsigned char в об'єкт QString

Функція розроблена відповідно до завдання (див. 1.4.6.3).

Так як поля структур даних, які використовуються в програмі і інформація з яких пишеться в таблиці робочого додатку, містять дані типу unsigned char, то виникає необхідність в перетворенні даного типу до типу, елементи якого можна записувати в осередки таблиць класу QTable наявними засобами (за допомогою функції–члену класу QTable - setText (...)).

Функція як параметр отримує елемент unsigned char для копіювання в рядок QString.

2.4.2.10 Функція копіювання об'єкта QString в масив типу чар

Ця функція визначена відповідно до завдання (дивись 1.4.6.3).

У програмі для створення діалогового вікна відкриття файлу використовується статична функція класу QFileDialog - getOpenFileName (...), в результаті виклику повертає рядок типу QString, що містить повний шлях доступу до файлу, що відкривається. Але, так як в даному проекті використані бібліотечні функції C (модуль stdio.h), то виникає необхідність в розміщенні імені файлу, що відкривається в масиві типу char, для того, щоб використовувати вказівник на даний масив при виклику функцій: fopen(...), fseek(...), fread(...).

На рис. 2.14 наведена блок–схема алгоритму функціонування даної функції.

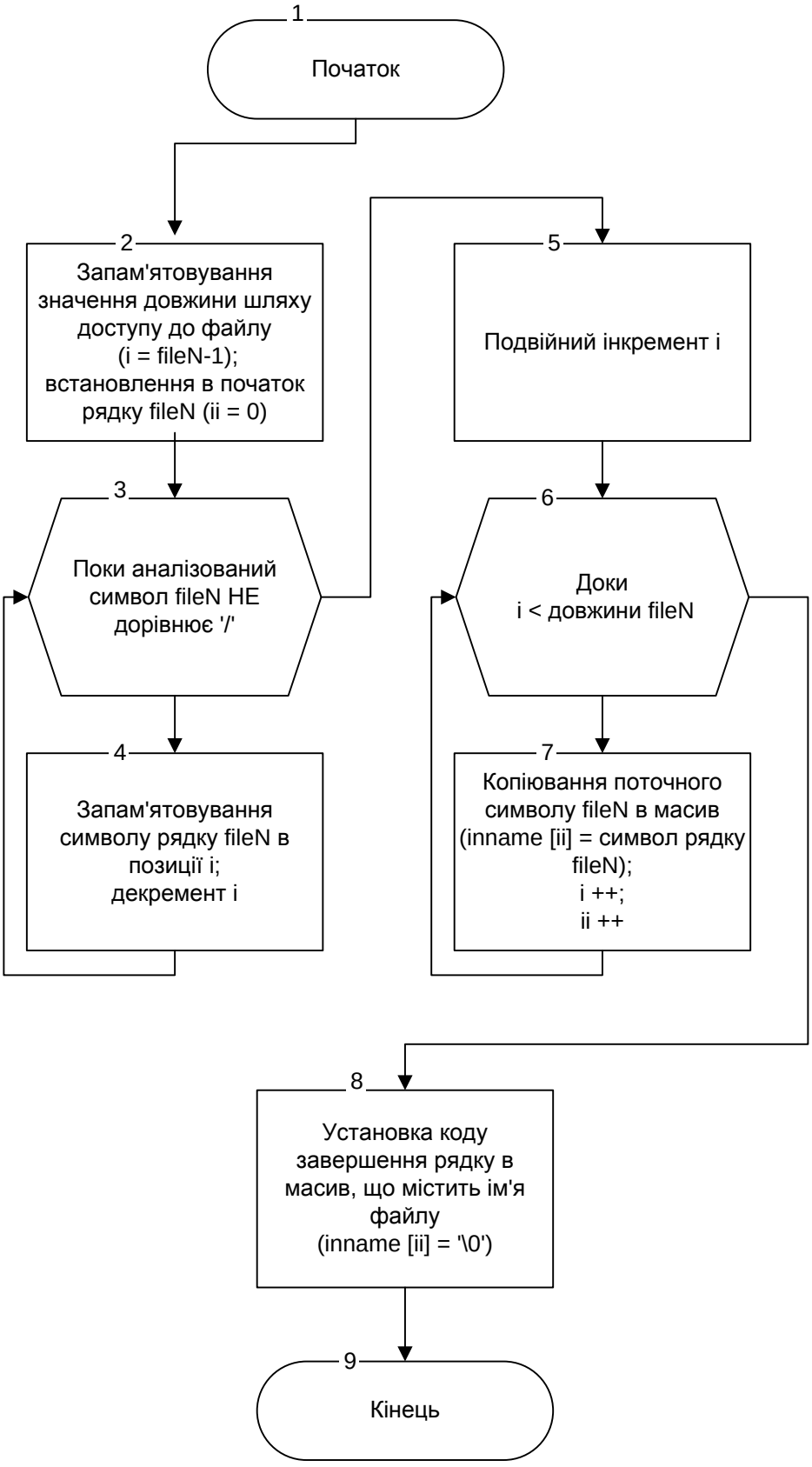


Рисунок 2.14 - Схема алгоритму функції копіювання об'єкта QString в масив типу char

2.4.2.11 Функція декодування даних в Unicode

Функція розроблена відповідно до завдання (дивись 1.4.6.2).

Параметри:

st - вихідна рядок символів;

buf - адреса змінної QString, в яку буде занесена вихідна рядок;

code - кодування, в якій міститься текст вхідного рядка.

Функція перетворює рядок символів типу char до типу QString. При цьому дані, лічені з файлів архівів, переводяться з кодування KOI8-R в Unicode.

Допустимі кодування:

"KOI8-R" (або "KOI8");

"CP 1251" (або "CP1251");

"CP 866" (або "CP866").

На рис. 2.15 наведена схема алгоритму даної функції.

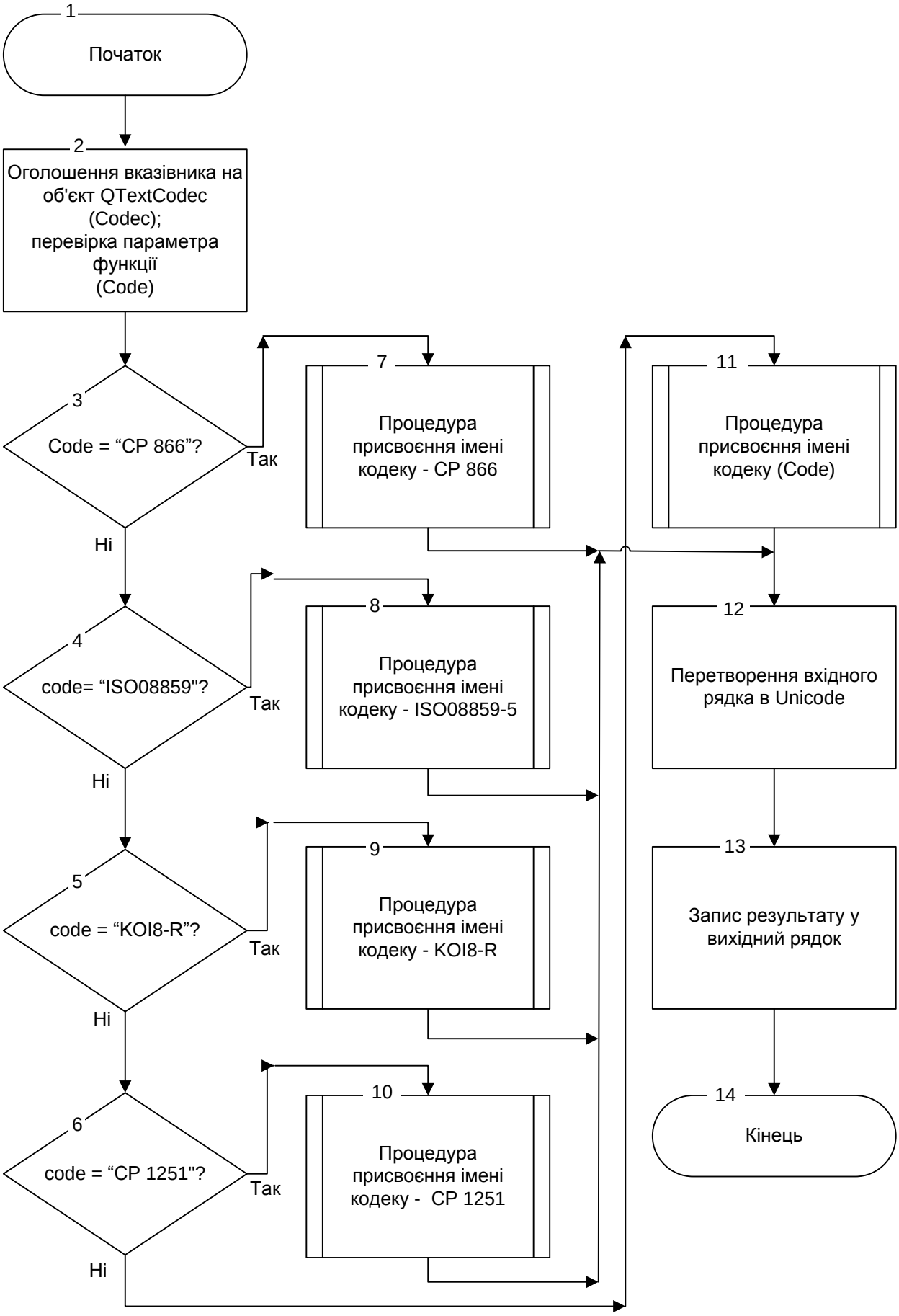


Рисунок 2.15 - Схема алгоритму функції перетворення даних в Unicode

2.4.2.12 Функція декодування даних з Unicode

Функція розроблена відповідно до завдання (див. 1.4.6.2).

Параметри:

st - вихідний об'єкт QString;

buf - адреса пам'яті, в яку буде занесена вихідна рядок;

sz - розмір пам'яті, виділеної під вихідний рядок;

code - кодування, в якій буде записаний текст вихідний рядки.

Допустимі кодування:

"KOI8-R" (або "KOI8");

"CP 1251" (або "CP1251");

"CP 866" (або "CP866");

"ISO08859-5".

На рис. 2.16 наведена схема алгоритму даної функції.

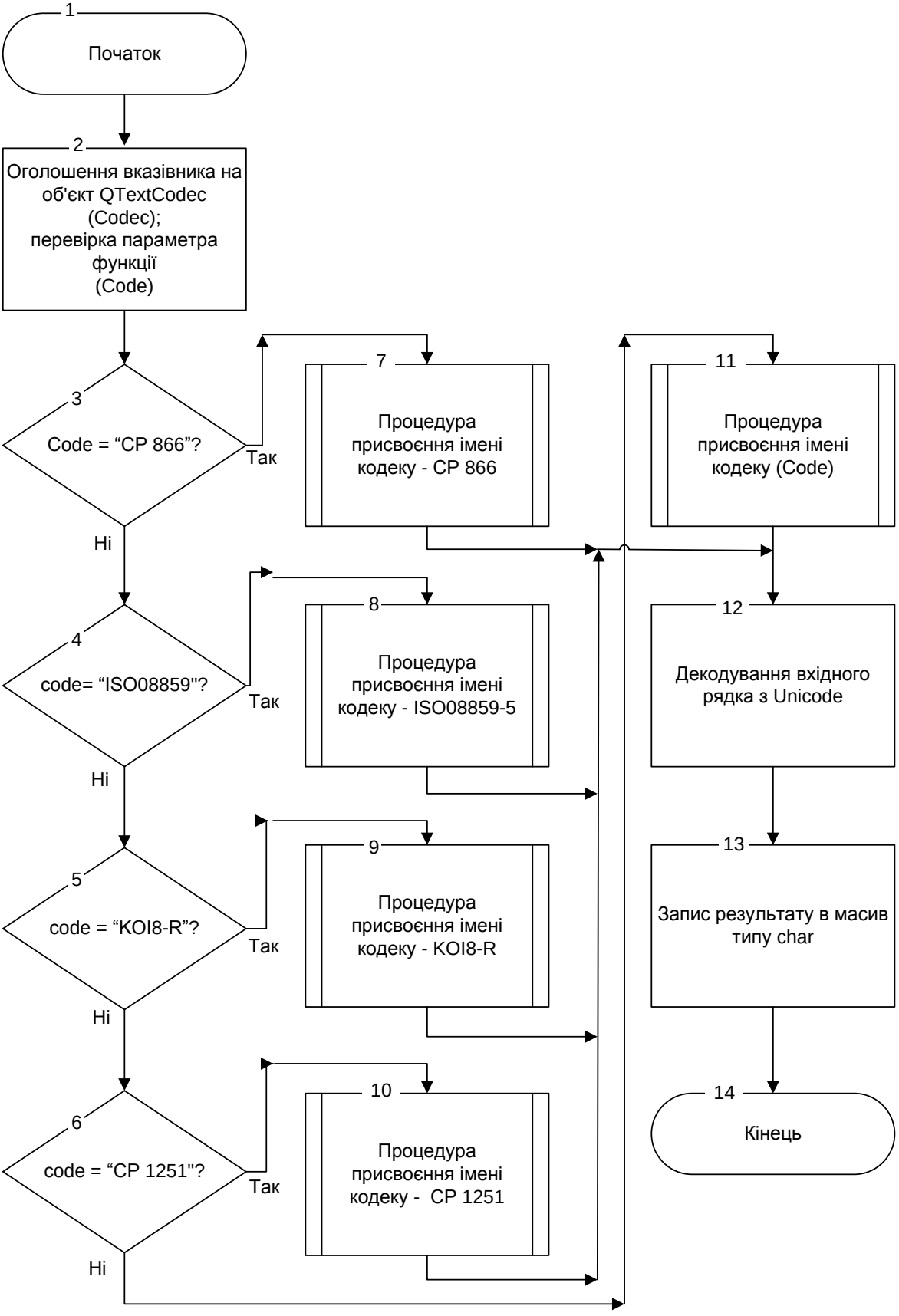


Рисунок 2.16 - Схема алгоритму функції запису даних в рядок з кодуванням

2.4.3 Головний модуль програми

Головний модуль програми, файл main.cpp, генерується системою Qt Designer автоматично.

Для цього необхідно відкрити файл проекту, що розробляється і викликати діалогове вікно New File (File \ New), де обирають опцію «C ++ Main - File (main.cpp)» і натискають кнопку ОК.

Даний модуль містить єдину функцію - main-функцію додатка.

2.4.3.1 Вміст функції main

У тілі main-функції створюється змінна типу QApplication, змінна типу створеної в Qt Designer форми, наприклад: якщо в Qt Designer створена головна форма програми має ім'я mainForm, то в тілі main-функції створюється змінна типу mainForm. Остання інструкція викликає на виконання функцію-член класу QApplication - exec(), яка запускає головний цикл подій та чекає виклику функції-члена класу QApplication - exit(...) або знищення головного об'єкта. Функція exec() повертає ціле число, встановлене в exit(...). У розробленому проекті, як зазначено в підпункті 2.3.1.6, створений слот, що містить виклик функції exit(0). Після виклику даної функції робота додатка успішно завершується (з кодом 0).

На рис. 2.17 наведена схема алгоритму функціонування функції main.

Рисунок 2.17 – Схема алгоритму main-функції програми

3 ЗАСТОСУВАННЯ РОЗРОБЛЕНОГО ПЗ

В даному розділі наведено приклад використання ПЗ МВФА (див. 3.1). Описано процес відкриття і аналізу файлів архівів AEF_Arch і HDA_Arch, наведено кілька малюнків із зображенням екранного інтерфейсу програми на різних етапах функціонування:

- відразу після запуску програми;
- під час натискання кнопки меню відкриття файлу AEF_Arch;
- під час виведення діалогового вікна відкриття файлів;
- малюнок з виведеними повідомленнями про невідповідність файлу, і рекомендаціями;
- малюнок з виведеними в таблиці даними архіву AEF_Arch;
- під час натискання кнопки меню перемикання сторінок додатку;
- малюнок з виведеними в таблиці даними архіву HDA_Arch;
- під час натискання кнопки меню виходу з програми.

У пункті 3.2 наведено короткий посібник користувача.

3.1 Приклад використання ПЗ МВФА

Відразу після запуску програми SAOZ.exe на екран виводиться сторінка архіву AEF_Ach.

На рис. 3.1 показаний вид робочого вікна програми після запуску програми.

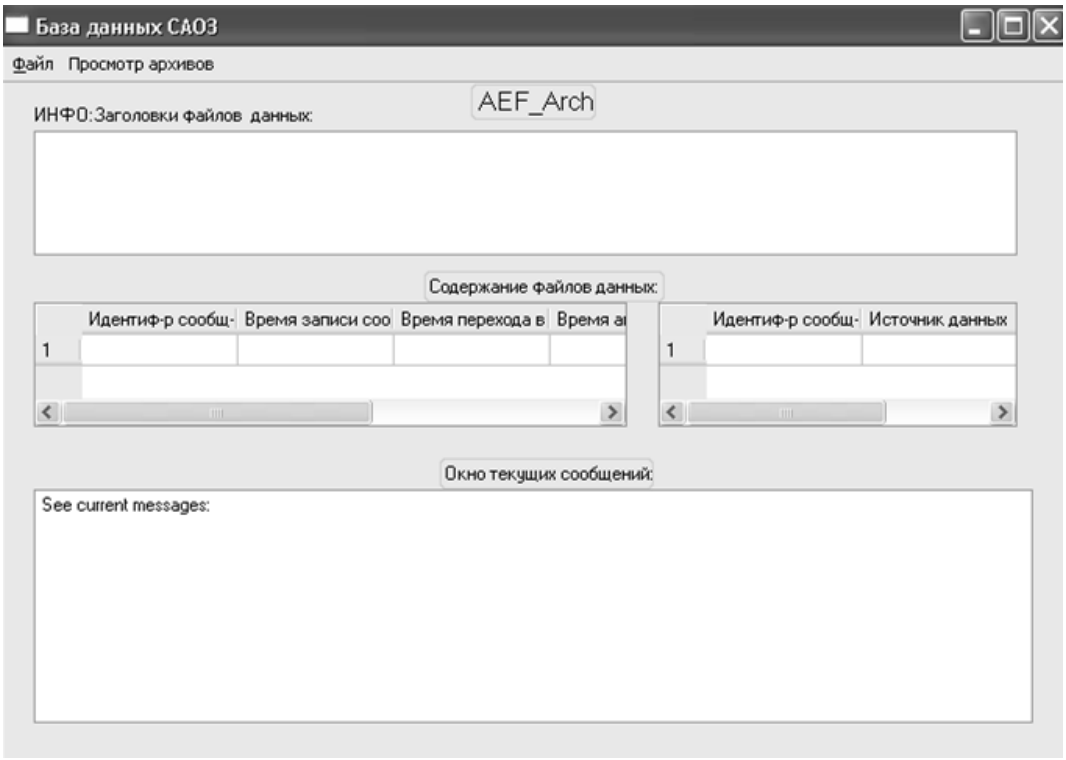


Рисунок 3.1 - Вид робочого вікна після запуску програми

Щоб переглянути вміст файлів архіву AEF_Arch, необхідно натиснути кнопку меню Файл \ Відкрити AEF_Arch, як показано на рис. 3.2.

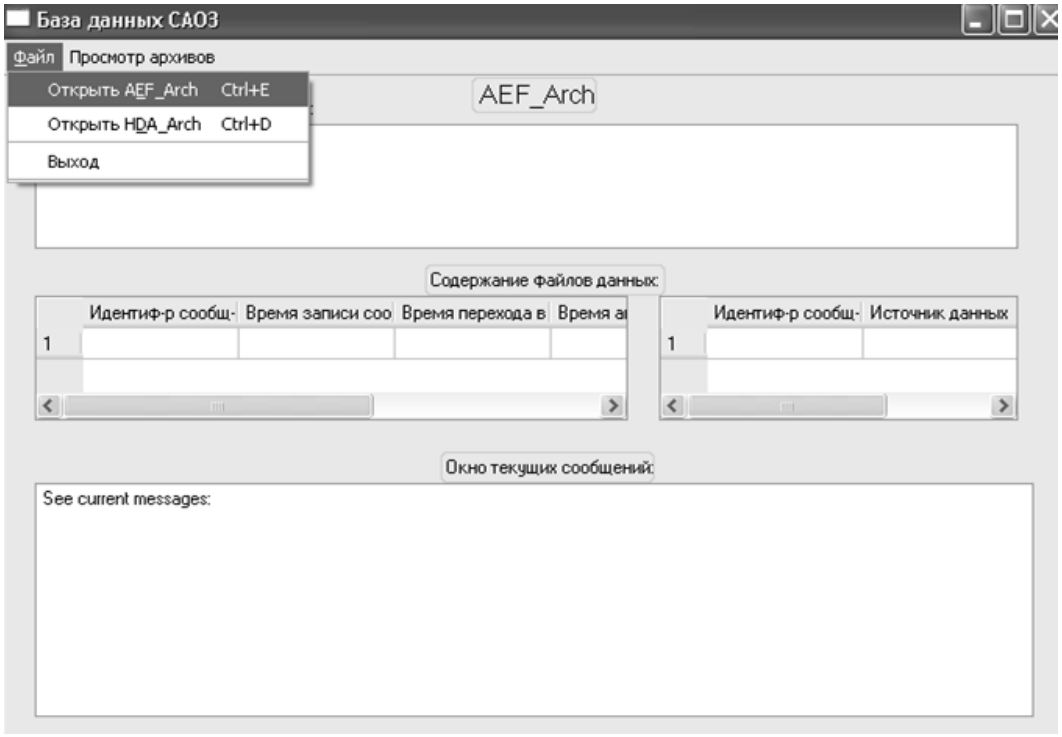


Рисунок 3.2 - Вид робочого вікна під час активації меню відкриття архіву AEF_Arch

Після натиснення на зазначену вище кнопку меню, на екрані, як показано на рис. 3.3, з'являється діалогове вікно відкриття файлів. Для перегляду обраний файл Event_Common.

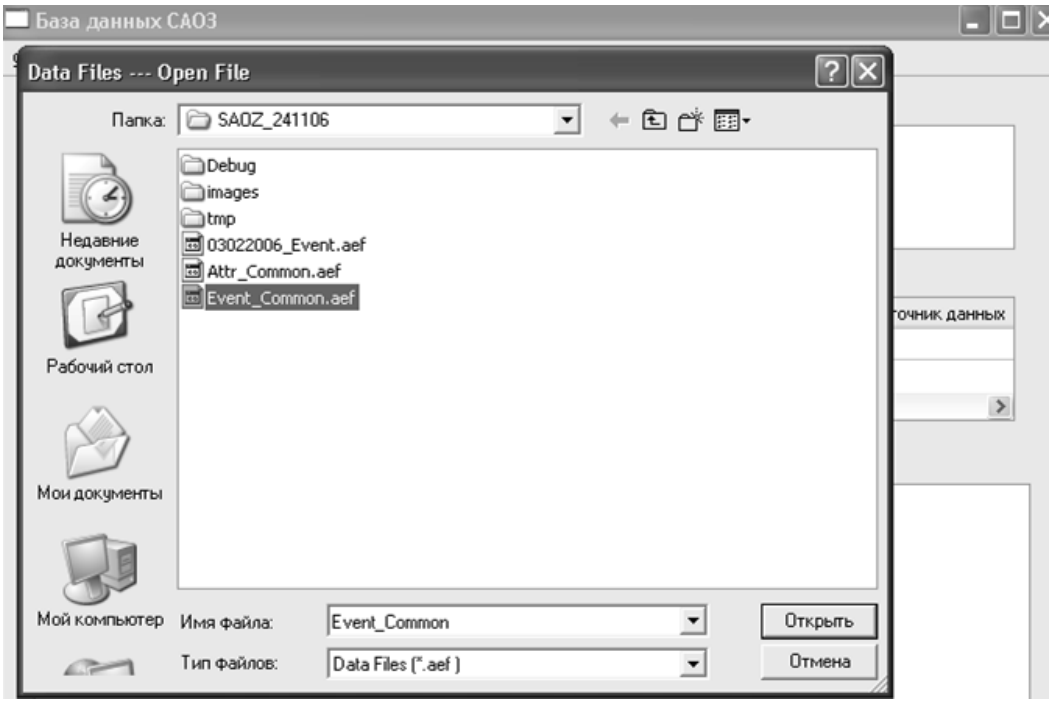


Рисунок 3.3 - Діалогове вікно відкриття файлів архіву AEF_Arch

На рис. 3.4 зображено вікно програми після відкриття файлу Event_Common.

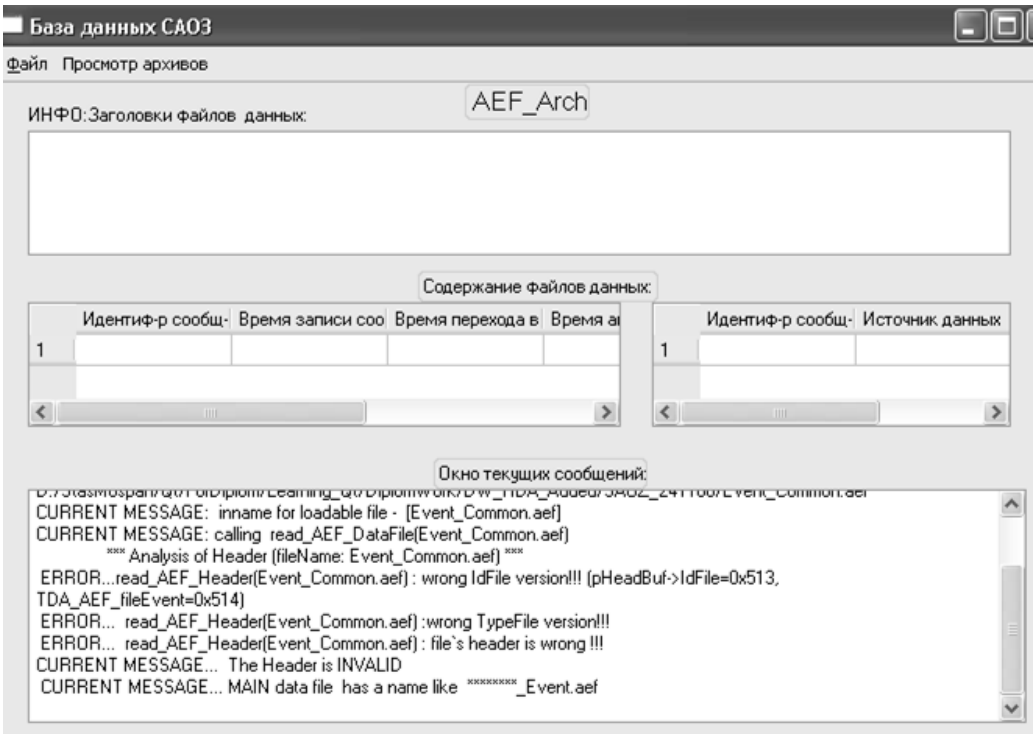


Рисунок 3.4 - Вікно програми після відкриття файлу Event_Common

У нижній частині програми, в віконці виведення поточних повідомлень, можна прочитати інформацію про вміст полів проаналізованого файлу. В останньому повідомленні дані рекомендації з приводу відкриття файлів архіву.

Після повторного виклику діалогового вікна відкриття файлів для перегляду вибирається наступний файл архіву (малюнок 3.5).

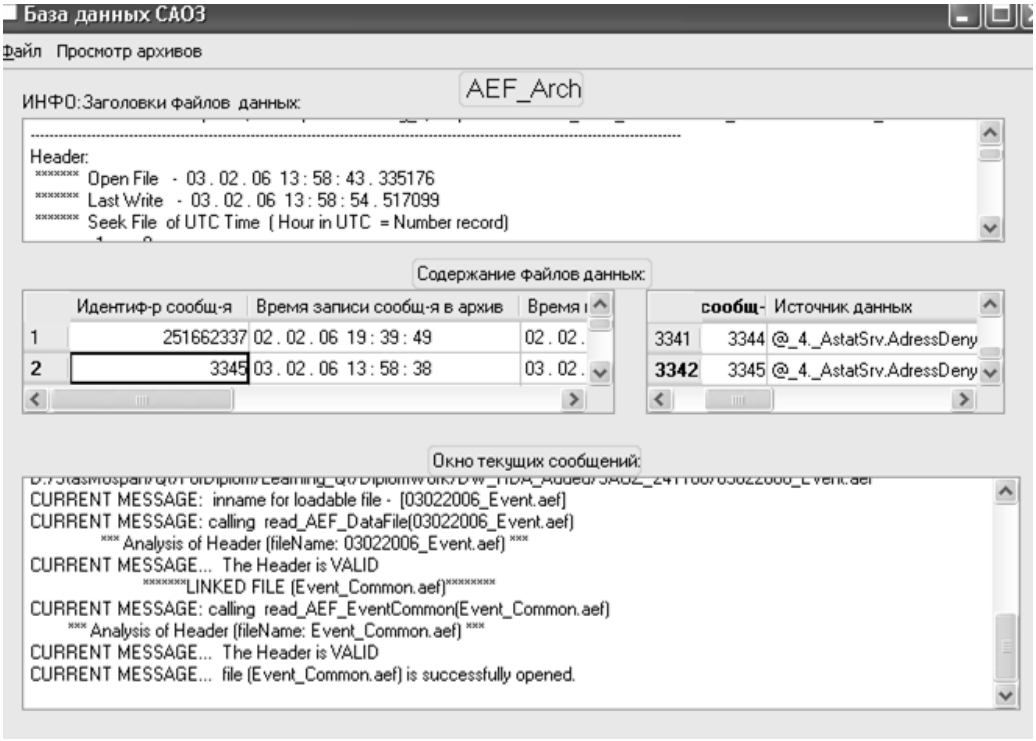


Рисунок 3.5 - Вид робочого вікна після відкриття файлу архіву AEF_Arch

Як видно з рис. 3.5, в таблиці виведено зміст пов'язаних файлів архіву. У віконцях виведення можна прочитати інформацію про вміст заголовка головного файлу і повідомлення, видані системою під час аналізу пов'язаних файлів.

На рис. 3.6 показаний вид робочого вікна в момент вибору пункту меню Перегляд архівів \ HDA_Arch.

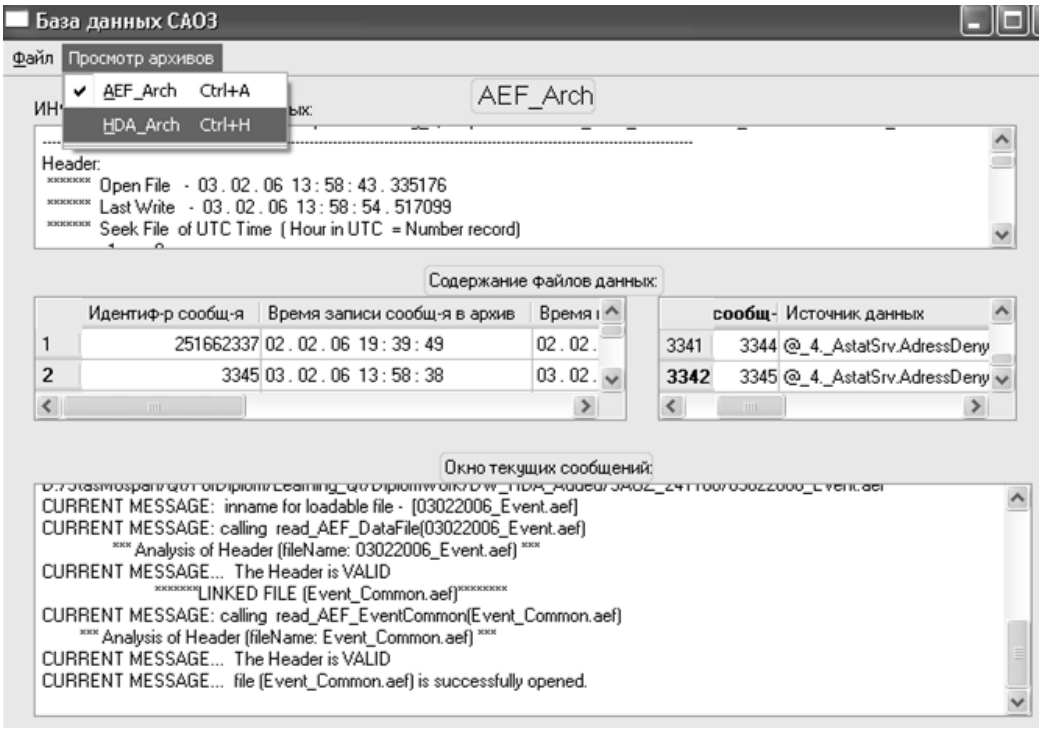


Рисунок 3.6 - Вид робочого вікна під час вибору пункту меню перемикання сторінок додатку

На рис. 3.7 зображено вікно програми з виведеними в таблиці даними із пов'язаних файлів архіву HDA_Arch і повідомленнями про хід аналізу заголовків і файлів.

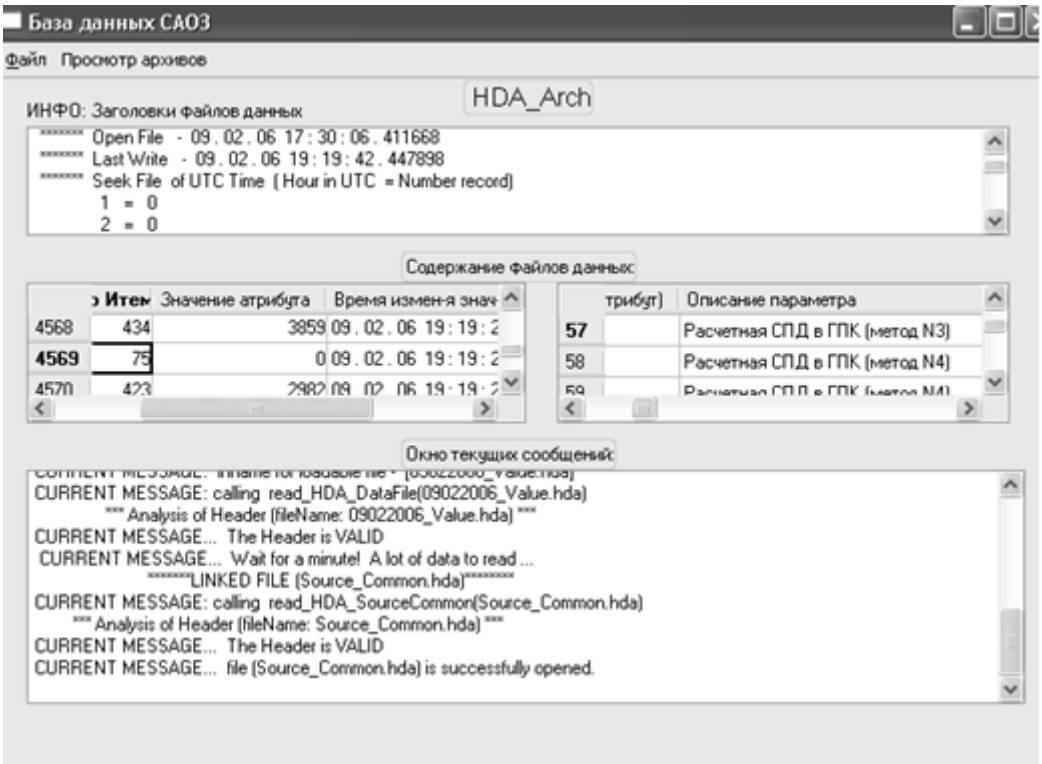


Рисунок 3.7 - Вид робочого вікна програми. Сторінка архіву HDA_Arch

На рис. 3.8 показаний вид робочого вікна програми при виборі пункту меню завершення роботи програми.

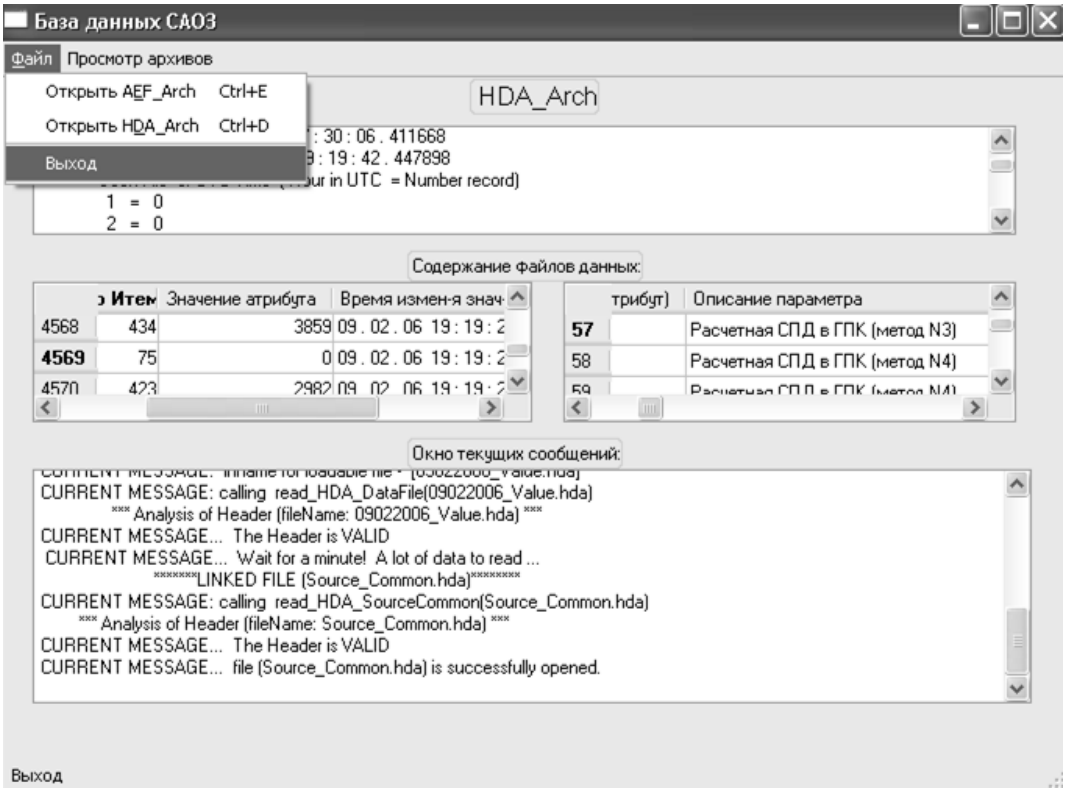


Рисунок 3.8 - Вид робочого вікна під час вибору пункту меню Вихід

3.2 Керівництво користувача

Запустіть програму SAOZ.exe.

Після запуску програми на екрані з'явиться робоче вікно програми (див. 3.1, рис. 3.1).

Перегляньте пункти меню і виберіть дії, які необхідно виконати:

– щоб відкрити файл даних архіву AEF, виберіть пункт меню Файл \ От-крити AEF_Arch;

– щоб відкрити файл даних архіву HDA, виберіть пункт меню Файл \ Відкрити HDA_Arch;

– щоб «погортати» сторінки додатка, виберіть відповідний пункт меню:

1) Перегляд архівів \ AEF_Arch;

2) Перегляд архівів \ HDA_Arch;

– щоб завершити роботу додатка, виберіть пункт меню Файл \ Вихід.

Натискання кнопки можна здійснити наступними способами:

– за допомогою маніпулятора «миша» встановіть курсор на потрібну кнопку і натисніть ліву клавішу «миші»;

– за допомогою клавіш управління курсором або клавіші «TAB» встановіть фокус на потрібну кнопку (при цьому вона змінить колір на темно-синій) і натисніть клавішу «Enter»;

– якщо на потрібній кнопці є підкреслений символ, то здійснити натискання цієї кнопки можна за допомогою послідовних дій:

1) вибору пункту меню верхнього рівня;
2) натиснення клавіші «Ctrl»;
3) натиснення відповідного символу.

Після того, як обрано пункт меню відкриття файлів одного з архівів, на екрані з'явиться діалогове вікно, в якому можна вибрати файл, що вас цікавить і натиснути кнопку «Відкрити» або клавішу «Enter».

Після відкриття файлу, в віконцях виведення інформації «ІНФО: Заголовки файлів даних» і «Вікно поточних повідомлень» можна прочитати відповідну даного файлу інформацію.

Зауваження. Якщо після відкриття файлу таблиці на відповідній сторінці додатка виявляються порожніми, то необхідно переглянути поточні повідомлення, що виводяться в віконці в нижній частині робочого вікна, де будуть дані відповідні рекомендації і аналіз файлу, що відкривається.

Для того, щоб на будь-якій з сторінок додатку витягати з таблиць пов'язану в логічне ціле інформацію, необхідно вибрати будь-яку клітинку, що цікавить записи в лівій таблиці. При цьому в правій таблиці виділеної виявиться запис, пов'язана з цікавить записом.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз потенційно небезпечних і шкідливих виробничих факторів , що впливають на персонал

Персональні ЕОМ типу IBM PC AT має наступні характеристики:

- споживана потужність 350 Вт;
- робоча напруга 220 В;
- напруга джерел живлення +12 В, -12 В, 5 В;
- робоча частота 50 Гц.

Виходячи з приведених характеристик, очевидно, що для користувача існує небезпека поразки електричним струмом у разі недбалого поводження з комп'ютером і порушення правил експлуатації (невиконання огляду відкритих частин ПЕВМ, що знаходяться під напругою або знятих для ремонту вузлів і т. д.).

Джерелами підвищеної небезпеки можуть служити наступні елементи:

- розподільний щит;
- джерела живлення;

У відповідності з [6] до легкої фізичної роботи відносяться всі види діяльності, вироблювані сидячи і не вимагаючи фізичної напруги. Робота користувача розробленого пакету програм відноситься до категорії Іа.

Згідно з [12] приміщення для ПЕОМ по ступеню небезпеки поразки людини електричним струмом відноситься до приміщень без підвищеної небезпеки (немає струмопровідної полови, вогкості, підвищеної температури, можливості одночасного дотику до корпусів устаткування з “землею” і до струмонесучих частин).

У відповідності з [7] при обслуговуванні ПЕВМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

- підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини;
- підвищений рівень статичної електрики;
- підвищений рівень електромагнітних випромінювань;
- підвищена або знижена температура повітря робочої зони;
- підвищена або знижена рухливість повітря;
- підвищена або знижена вогкість повітря;
- відсутність або недолік природного світла;

- підвищена пульсація світлового потоку;
- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- розумове перенапруження;
- емоційні навантаження;
- монотонність праці.

Щодо до впливу на довкілля, то програмний засіб, який було розроблено під час дипломного проекту на довкілля ніяк не впливає.

Діяльність за темою магістерської роботи в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: [13-18].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на відалення (знешкодження), утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- Батарейки та акумулятори (малі) - ІІІ клас небезпеки
- Макулатура - IV клас небезпеки
- Матеріали пакувальні, що не вміщують целюлозу - IV клас небезпеки
- Матеріали пакувальні, що вміщують п/ет, п/пр - IV клас небезпеки
- Змінні носії інформації - IV клас небезпеки

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм [19].

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення Зазначені для зберігання відходів місця чи об'єкти повинні використовуватися лише для заявлених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- Відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;

- Відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення;

В разі тимчасового зберігання відходів у стаціонарних складах або промислових приміщеннях повинні бути забезпечені санітарно-гігієнічними етичні вимоги до повітря робочої зони згідно з [20].

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) і інш.

Проведення заготовки, здачі, переробки та реалізації металобрухту встановлені окремо [21].

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Вимоги безпеки при поводженні з відходами:

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Наведено перелік деяких відходів, які передаються на утилізацію організаціям, які мають ліцензію на поводження з відходами як вторинної сировини:

- Лом і кускові відходи міді, бронзи, латуні, алюмінію, свинцю;
- Брухт чорних металів;
- Макулатура;
- Склобій;
- Матеріали текстильні вторинні;
- Відходи деревини кускові
- Відпрацьовані фільтрувальні засоби індивідуального захисту
- Відпрацьовані вогнегасники
- Матеріали пакувальні вторинні

Відвантаження таких відходів здійснюється відповідно до договору (контракту).

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

4.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка усугубляє тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм надає на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Ступінь ураження людини електричним струмом залежить від наступних факторів:

- значення сили струму;
- електричного опору тіла людини і тривалості протікання через нього струму;
- роду і частоти струму;
- індивідуальних властивостей людини і навколишнього середовища.

Даним проектом передбачаються наступні технічні способи і засоби, застережливі поразки людини електричним струмом:

- заземлення електроустановок;
- занулення;
- захисне відключення;
- електричне розділення сітей;
- використовування малої напруги;
- ізоляція струмоведучих частин;
- огорожа електроустановок.

Проведемо розрахунок заземлюючого пристрою.

Початкові дані для розрахунку заземлюючого пристрою:

– напруга установки, що заземляється, - 220В;
– режим нейтралу мережі - з ізольованою нейтралою;
– питомий опір ґрунту – 100 Ом·м(суглинок);
– гранично допустимий опір заземлюючого пристрою - 4 Ом;
– характеристика кліматичної зони (III):

а) середня багаторічна низька температура, ^{o}C - від –14 до -10;

б) тривалість замерзання вод, дні - 150;

в) коефіцієнт сезонності для вертикального електроду завдовжки 3м -1,5.

Визначимо розрахунковий опір ґрунту (Ом·м) по формулі (4.1).

$$\rho_{расч} = \psi \cdot \rho = 1{,}5 \cdot 100 = 150 Ом \cdot м \qquad (4.1)$$

де ρ - питомий опір ґрунту;

ψ_i – кліматичний коефіцієнт, що враховує стан ґрунту під час вимірювань (таблиця 4 [7]).

Розрахуємо опір розтіканню одиночного трубчастого заземлювача по формулі (4.2).

$$R_{з.1} = (\frac{\rho_{расч}}{2 \cdot \pi \cdot l}) \cdot \ln(4 \cdot \frac{l}{d}) \qquad (4.2)$$

де l – довжина заземлювача (l=5м);

d – діаметр труби і стрижня (d=0,05м);

$$R_{з.1} = (\frac{\rho_{расч}}{2 \cdot \pi \cdot l}) \cdot \ln(4 \cdot \frac{l}{d}) = (\frac{150}{2 \cdot 3{,}14 \cdot 5}) \cdot \ln(4 \cdot \frac{5}{0{,}05}) = 28{,}6 \text{ Ом}$$

Розрахуємо кількість паралельно сполучених одиночних заземлювачей по формулі (4.3).

$$n = \frac{R_{з.1}}{R_{доп} \cdot \eta} = \frac{28{,}6}{4 \cdot 0.47} = 15{,}2 \qquad (4.3)$$

де Rдоп=4. – самий допустимий опір заземлюючого пристрою;

η - коефіцієнт використовування ґрунтового заземлення (для шістки заземлювачей η=0,47).

Округлятимемо отримане значення у більшу сторону n=[15,2]=16.

Розрахуємо довжину горизонтальної сполучної смуги по формулі (5.4).

$$L = a \cdot (n-1) = 3 \cdot (16-1) = 45м \tag{4.4}$$

де a – відстань між вертикальними заземлювачами (a=3м);

n – кількість вертикальних заземлювачей (n=16).

Розрахуємо опір сполучної смуги по формулі (4.5).

$$R_n = \frac{\rho_{расч}}{2 \cdot \pi \cdot l} \cdot \ln(\frac{L^2}{d \cdot h}) \tag{4.5}$$

де d – еквівалентний діаметр смуги шириною l=5 (d=0,05м);

h – глибина заставляння смуги (h=0,8м).

$$R_n = \frac{\rho_{расч}}{2 \cdot \pi \cdot l} \cdot \ln(\frac{L^2}{d \cdot h}) = \frac{150}{2 \cdot 3{,}14 \cdot 5} \cdot \ln(\frac{45^2}{0{,}05 \cdot 0{,}8}) = 51{,}7 \text{ Ом}$$

Розрахуємо результуючий опір заземлюючого електроду з урахуванням сполучної смуги по формулі (5.6).

$$R_{гр} = \frac{R_{з.1} \cdot R_n}{R_{з.1} \cdot \eta_n + R_n \cdot n \cdot \eta_з} \le R_{доп} \tag{4.6}$$

де η_n – коефіцієнт використовування сполучної смуги (для 6-і заземлювачей $\eta_{n=0,27}$).

$$R_{гр} = \frac{R_{з.1} \cdot R_n}{R_{з.1} \cdot \eta_n + R_n \cdot n \cdot \eta_з} = \frac{26{,}6 \cdot 51{,}7}{26{,}6 \cdot 0{,}27 + 51{,}7 \cdot 16 \cdot 0{,}47} = 3{,}47 \text{ Ом}$$

3,47<4 $\Rightarrow$ умова забезпечення електробезпеки персоналу виконується.

Таким чином, остаточна кількість заземлювачей 15 шт.

4.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Підвищення працездатності людини і збереження його здоров'я забезпечується стабільними метеорологічними умовами.

Мікроклімат виробничих приміщень – це поєднання температури, вогкості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводить до порушення систем кровообігу, нервової і пітовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

В приміщенні для виконання робіт операторського типу, пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (див. таблицю 4.1).

Таблиця 4.1 - Оптимальні параметри мікроклімату в робочій зоні виробничого приміщення для категорії робіт 1

Період року	Температура, оС	Відносна вогкість %	Швидкість руху повітря, м/с
Холодний	22.24	40.60	0,1
Теплий	23.25	40.60	0,1

Оскільки в приміщенні немає джерел виділення шкідливих речовин, можна використовувати природну вентиляцію. Площа приміщення складає 32 м2 . Для забезпечення прийнятних параметрів мікроклімату в приміщенні з такою площею можна використовувати 1 кондиціонер типу БК-2000.

Спектр випромінювання монітора комп'ютера включає рентгенівську, ультрафіолетову, інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння нехтує мала, оскільки цей вид випромінювання поглинається речовиною екрану.

Для зниження дії електромагнітного випромінювання пропонується захист часом і відстанню. Захист часом передбачає обмеження часу перебування людини в зоні дії полів. Тривалість роботи на ПЕОМ повинна складати не більше 3.5–4.5 години.

Також необхідно забезпечити раціональне освітлення в робочому приміщенні. В проекті, що розробляється, передбачається використовувати суміщене освітлення. В

світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи володіють високою світловою віддачею до 75 Лам/Вт і більш, тривалим терміном служби до 10000 годин, спектральним складом випромінюваного світла, близьким до сонячного.

Зорова робота оператора ПЕВМ відповідно до [10] відноситься до розряду Va з світловим потоком Фл=3120 кожна. Нормована освітленість на робочому місці (Ен) при загальному освітленні складає 200 лк.

Проведемо розрахунок кількості світильників в робочому приміщенні завдовжки a=6 м, шириною b=3 м, заввишки c=4 м. Формула розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку (4.7):

$$Фл = \frac{Eн \cdot S \cdot Z \cdot K}{N \cdot U \cdot M} \qquad (4.7)$$

де *Фл* – світловий потік, Лм;

Ен – нормована освітленість;

S – площа підлоги, кв.м;

Z=1.1–1.3 - поправочний коефіцієнт світильника (для стандартних світильників);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників;

N – число світильників;

U=0.55–0.6 – коефіцієнт використовування, залежний від типу світильника, показника індексу приміщення і др.;

M – число ламп в світильнику.

З формули (5.7) виразимо N і визначимо кількість світильників для даного приміщення:

$$N = \frac{Eн \cdot S \cdot Z \cdot K}{Фл \cdot U \cdot M}$$

$$N = \frac{200 \cdot 18 \cdot 1{,}2 \cdot 1{,}5}{3120 \cdot 0{,}6 \cdot 2} = 1{,}7$$

Виходячи з цього, рекомендується використовувати 2 світильники. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. (4.1).

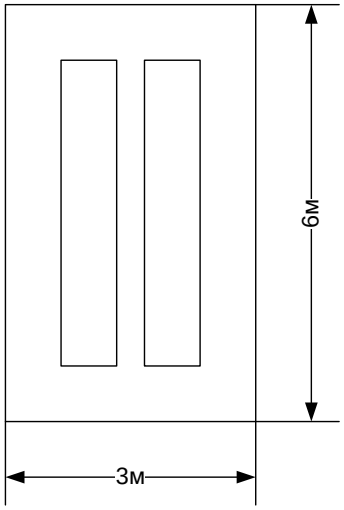


Рисунок 4.1 – Схема розташування світильників

Зниження шуму можна добитися раціонально розпланувавши приміщення, установкою устаткування на спеціальні амортизуючи прокладки. Згідно вимогам [9] рівні звуку не повинні перевищувати 50 дБ.

Для зниження стомлюваності обслуговуючого персоналу в приміщеннях, де розташовані обчислювальні засоби передбачаються використовувати спокійні колірні поєднання і покриття, що не дають відблисків. Від електромагнітного випромінювання, витікаючого від ПЕОМ, використовуються захисні екрани.

Для забезпечення чистоти повітря і відповідних мікрокліматичних умов пропонується застосувати приточування-витяжну вентиляцію. Для зменшення дії шкідливих речовин і загазованості для роботи з розплавленими матеріалами робоче місце забезпечується примусовою витяжною вентиляцією. Цей метод забезпечує притоку потрібної кількості свіжого повітря (30 мм^3 /ч на одного працюючого).

Кількість повітря, яка необхідна подавати в приміщення для забезпечення необхідних параметрів повітряного середовища, визначається на підставі кількість тепла, вологи і шкідливих речовин, що поступають в приміщення, а також враховуючи видалення повітря місцевими відсмоктуваннями від устаткування, загальнообмінною вентиляцією.

4.4 Рекомендації по пожежній профілактиці

Пожежі представляють небезпеку для життя людини і зв'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що спричиняє за собою порушення ходу технологічного процесу.

Горючими матеріалами в приміщенні, де розташовані ПЕОМ, є:

– поліамід - матеріал корпусу мікросхеми. Горюча речовина. Температура самозапалення 420 ^{o}С, енергія запалення 2мДж;

– полівінілхлорид - ізоляційний матеріал. Горюча речовина. Температура самозаймання 480 єС, енергія запалення 50мДж;

– склостоліт ДЦ - матеріал друкарської платні. Складногорючий матеріал;

– пластикат кабельний No.489 - матеріал ізоляції кабелю. Складногорючий матеріал. Температура самозаймання 1500 єС;

– плита деревостружкова - будівельний і обробний матеріал, матеріал з якого виготовлені меблі. Складнозапалений матеріал. Показник горючості 1.8;

– папір – довідкова і робоча документація, література. Горючий матеріал. Показник горючості більше 2.1.

Відповідно до [11] приміщення відноситься до категорії В (пожежовибухонебезпечної).

Джерелами запалення можуть бути:

– іскри при замиканні і розмиканні ланцюгів;

– іскри і дуги коротких замикань;

– перегріви від тривалого перевантаження і наявності перехідного опору;

– розряди статичної електрики.

Для того, щоб зупинити реакцію горіння, порушують умови її виникнення і підтримки. Звичайно для гасіння використовуються порушення двох основних умов сталого стану – пониження температури і режим руху газів. Пониження температури може бути досягнутий шляхом введення речовин, які поглинають багато тепла в результаті випаровування і дисоціації (наприклад, вода, порошки).

При повному тому, що згоряє органічних сполук утворюються С, SO, Н Про, N, а при тому, що згоряє неорганічних з'єднань – оксиди. Залежно від температури плавлення і тривалості реакції можуть знаходиться або у вигляді розплавів (Al O, Ti O), або підійматися в повітря у вигляді диму (P O, Na Про, MgO).

Склад продуктів неповного згоряє горючих речовин складений і різноманітний. Це можуть бути горючі речовини:

– H, C, CH;

– атомарний водень і кисень;

– різні радикали – OH, CH .

Продуктами неповного згоряє можуть бути також оксиди азоту, спирти, альдегіди, кетони і високотоксичні з'єднання, наприклад, синильна кислота.

Для захисту персоналу від дій небезпечних і шкідливих чинників пожежі проектом передбачено застосування промислового фільтруючого протигаза з коробкою марки В (жовтий).

До системи запобігання пожежі відносяться: запобігання утворення горючого середовища і освіти в горючому середовищі джерел запалення, забезпечення пожежебеспеки устаткування.

Щоб запобігти пожежі в обчислювальних центрах, проектом пропонується виконання наступних вимог:

– електроживлення ЕОМ має автоматичне блокування відключення електроенергії на випадок перегріву системи, що може бути результатом зупинки системи охолоджування і кондиціонування;

– система вентиляції обчислювальних центрів обладнується блокуючими пристроями, що забезпечують її відключення на випадок пожежі. Система обладнується вогнеперегороджуючими клапанами;

– застосування устаткування, що задовольняє вимогам електростатичної іскробезпеки [7];

– після закінчення роботи, перед закриттям приміщення, всі електроустановки і персональні комп'ютери відключаються від сіті електроживлення;

– в приміщеннях обчислювальних центрів забороняється:

– влаштовувати електророзетки на основах, що згоряють;

– використовувати синтетичні доріжки і килими;

– користуватися побутовими електронагрівальними приладами;

– захаращувати евакуаційні виходи і проходи;

– влаштовувати на вікнах глухі грати;

– залишати без нагляду включену в електромережу апаратуру, що використовується для вимірювань і нагляду.

Для протипожежного захисту проектом пропонується обладнати приміщення площею 18 м2, яке відноситься до категорії В, автоматичною протипожежною сигналізацією із застосуванням датчиків сповіщення РІД-1 (оповіщувач димовий іонізаційний) в кількості 1 штуки і застосовується в первинних засобах пожежегасінні. Площа контрольована оповіщувачем 150 м2.

Крім того, необхідно проводити навчання робочого персоналу правилам пожежної безпеки.

Розрахуємо вірогідність виникнення пожежі у виробничому приміщенні у разі запалювання транзистора:

$$Q = л \cdot T \cdot Ркз/отк \cdot Qвоспл \cdot Рзащ \quad (4.8)$$

де л – інтенсивність відмов пожежеопасних ЕРІ;

Т – час роботи пожежеопасного ЕРІ за оцінюваний інтервал часу;

Ркз/отк - умовна вірогідність виходу ЕРІ в стан короткого замикання при його відмові;

Qвоспл - вірогідність запалювання ЕРІ, що знаходиться в стані короткого замикання;

Рзащ – вірогідність відмови захисту пожежеопасного ЕРІ. Якщо захист відсутній, Рзащ приймається рівній 1.

Вірогідність виникнення пожежі у разі запалювання транзистора:

$$Q = 1 \cdot 10^{-6} \cdot 1 \cdot 10^{-4} \cdot 0.1 \cdot 1 \cdot 10^{-4} = 1 \cdot 10^{-15}$$

Розрахована вірогідність виникнення пожежі значно менше допустимої, яка складає $1 \cdot 10^{-6}$.

В даному розділі були проаналізовані небезпечні і шкідливі виробничі чинники, що роблять вплив на персонал, розроблені заходи щодо техніки безпеки, заходу, забезпечуючи виробничу санітарію і гігієну праці, а також заходи щодо пожежної профілактики.

ВИСНОВКИ

Відповідно до завдання, був розроблений модуль відображення експертної системи аналізу баз даних САОЗ і СУОР-І.

Проектом передбачена можливість ідентифікації файлів архівів ОБД МСКУ-М, аналіз структури файлів, декодування вмісту файлів і наочне виведення інформації на екран з можливістю «навігації» за таблицями і переглядом пов'язаних даних з двох таблиць. Реалізовано двохсторінковий інтерфейс, що дозволяє переглядати і зіставляти дані з обох архівів: подій і повідомлень про порушення (AEF_Arch) і архіву параметрів технологічного процесу (HDA_Arch).

Проектом запропоновано комплекс заходів щодо забезпечення охорони праці та техніки безпеки при роботі з ПЕОМ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Программно-технический комплекс технологических защит (Конфигурация САОЗ). Программное обеспечение инженерно-диагностической станции ИДС. Архив компонент поставки и компонент сопровождения.

2) В.И.Ященко, В.И.Дедухно, К.Е.Герасименко, О.Н.Заремская. Современные подходы к модернизации оборудования АСУ ТП АЭС на примере системы аварийного охлаждения активной зоны реактора ВВЭР-440.- ЗАО «СНПО «Импульс», Украина.

3) В.С. Харченко, М.А. Ястребенецкий, В.Н. Васильченко Нормирование и оценка безопасности информационных и управляющих систем АЭС: регулирующие требования к программному обеспечению // Ядерная и радиационная безопасность, 2002, № 1. – С. 18 – 33.

4) Qt Reference Documentation (Commercial Edition).

5) Тель Ж. Введение в распределенные алгоритмы. Пер. с англ. В. А. Захарова. / Ж. Тель − М.: МЦНМО, 2009. − 616 с.

6) 12.1.005–88. ССБТ. Общие санитарно–гигиенические требования к воздуху рабочей зоны.

7) 12.0.003–74. ССБТ. Опасные и вредные производственные факторы. Классификация.

8) 12.1.009–76. ССБТ. Электробезопасность. Термины и определения

9) 12.1.003-83. ССБТ. Шум. Общие требования безопасности

10) ДБН В.2.5-28-2006. Природне і штучне освітлення

11) НАПБ Б.03.002-2007. Нормы определения категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности

12) ДСТУ Б А.3.2-13:2011 Система стандартів безпеки праці. Будівництво. Електробезпечність. Загальні вимоги

13) Закон України «Про охорону навколишнього природного середовища»

14) Закон України «Про забезпечення санітарного та епідемічного благополуччя населення»

15) Закон України «Про відходи»

16) Закон України «Про охорону атмосферного повітря»

17) Закон України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру»

18) Водний кодекс України

19) ДСанПіН 2.2.7.029-99. Гігієнічні вимоги щодо поводження з промисловими відходами та визначення їх класу небезпеки для здоров'я населення.

20) ГОСТ 12.1.005-88 Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны.

21) Законом України «Про металобрухт»

ДОДАТОК А.
Лістинг модулю mainform.ui.h

```
/****************************************************************************
** ui.h extension file, included from the uic-generated form implementation.
**
** If you wish to add, delete or rename functions or slots use
** Qt Designer which will update this file, preserving your code. Create an
** init() function in place of a constructor, and a destroy() function in
** place of a destructor.
****************************************************************************/
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include  "aeftypes.h"
#include "aetypes.h"
#include "commontypes.h"
#include "UtilsTr.h"
#include "hdatypes.h"
#include <qstring.h>
#include <qtextcodec.h>
#include <qbuffer.h>
#define CT_NORMILIZE_DEFREADBUFFER 10000
unsigned long bufReadCount =0;
unsigned long bufMaxCount =0;
char  inname [255] =" ";  /* массив для имени открыв-го файла*/
char prtime[255];    /* массивы для размещения значений полей структур */
char prtime1[255];
char prtime2[255];
char prtime3[255];
char prtime4[255];
char DataSource [80];
char MessageStat [255];
char nameEvCom[] = "Event_Common.aef"; /* массив, содерж-й имя связанного файла */
int kod = 0;
 //******************* variables  for HDA_Arch**********
 char nameSCom[] = "Source_Common.hda";
 char descrServ[80];
 char nameServ[80];
 char nameUnit[80];
// Функция преобразования строки символов в QString
// st   - исходная строка символов
// buf - адрес переменной QString, в которую будет занесена выходная строка
// code - кодировка, в которой содержится текст входной строки
// Допустимые кодировки
//   "KOI8-R" (или "KOI8")
//   "CP 1251" (или "CP1251")
//   "CP 866" (или "CP866")
// Возвращает строку в объекте QString
// В случае возникновения ошибки - строка QString содержит пустую строку ("")
QString qstringtmp[10];
int currqstr = 0;
```

```
                                        ****************************
 QString& TM_CharToQString(const char *st, const char* code)
{
      QTextCodec* codec;
      //ISO8859-5
      if( strcmp(code, "ISO8859-5")==0 ){
                              codec = QTextCodec::codecForName("ISO8859-5");
                  }
      else
      //KOI8-R
      if( strcmp(code, "KOI8")==0 ){
            codec = QTextCodec::codecForName("KOI8-R");
                  }
      else
      //CP 1251
      if( strcmp(code, "CP1251")==0 ){
            codec = QTextCodec::codecForName("CP 1251");
                  }
      else
      //CP 866
      if( strcmp(code, "CP 866")==0 ){
            codec = QTextCodec::codecForName("IBM 866");
      }
      //Все остальные кодировки
      else{
            codec = QTextCodec::codecForName(code);
      }

      currqstr++;
      if( currqstr>=10 ) currqstr = 0;

      qstringtmp[currqstr] = codec->toUnicode(st);

      return qstringtmp[currqstr];
}
                                        ****************************
// Функция преобразования QString в строку символов
// st   - исходный объект QString
// buf - адрес памяти, в которую будет занесена выходная строка
// sz - размер памяти, выделенной под выходную строку
// code - кодировка, в которой будет записан текст выходной строки
// Допустимые кодировки
//   "KOI8-R" (или "KOI8")
//   "CP 1251" (или "CP1251")
//   "CP 866" (или "CP866")
//
// Возвращает указатель на адрес памяти, в которую записывается выходная строка
// В случае возникновения ошибки - возвращает NULL
 char* TM_QStringToChar(QString *st, char *buf, const int sz, const char* code)
{
      QTextCodec* codec;

      //ISO8859-5
```

```
                    if( strcmp(code, "ISO8859-5")==0 ){
            codec = QTextCodec::codecForName("ISO8859-5");
                    }
        else
        //KOI8-R
        if( strcmp(code, "KOI8")==0 ){
            codec = QTextCodec::codecForName("KOI8-R");
                    }
        else
        //CP 1251
        if( strcmp(code, "CP1251")==0 ){
            codec = QTextCodec::codecForName("CP 1251");
                    }
        else
        //CP 866
        if( strcmp(code, "CP 866")==0 ){
            codec = QTextCodec::codecForName("IBM 866");
        }
        //Все остальные кодировки
        else{
            codec = QTextCodec::codecForName(code);
        }

        QCString locallyEncoded = codec->fromUnicode( *st );
        strcpy(buf, locallyEncoded);

        return buf;
}
                                ****************************************
 /*  ф-я- конструктор.
 "Прорисовываем" страницу приложения AEF.
 Выделяем память под глоб.массивы.
Обнуляем глобальные указатели.
Устанавливаем в окошко вывода первичную инф-ю.*/
void mainForm::init()
{
    AEF_HDA_WidgetStack->raiseWidget(0);
      TempForChar = "Temporary string";
    memset(&inname, 0,255);
    table1->setUpdatesEnabled(true);
    table2->setUpdatesEnabled(true);
    table3->setUpdatesEnabled(true);
    table4->setUpdatesEnabled(true);
memset(&DataSource, 0, 80);
memset( &MessageStat, 0, 255);
showLinkedData = false;
    pHeadBufEvCom = 0;
    fileHeadEvCom = 0;
    pHeadBuf = 0;
    fileHead = 0;
    textBrr3->setText(QString("See current messages:  "));
    insPos3 = 1;
    setCurs3 =2;
```

```
   //********* for HDA ***********

 showLinkedData_HDA = false;
 pHeadBuf_HDA = 0;
 fileHead_HDA = 0;
 pHeadBufSCom = 0;
 fileHeadSCom = 0;
 textBrr5->setText(QString("See current messages:  "));
 insPos5 = 1;
 setCurs5 =2;
}
                                        ***************************
/* слот открытия файлов архива AEF */
void mainForm::fileOpen_AEF()
{
   clear_AEF_t1();
  clear_AEF_t2();

   fileName  = QFileDialog::getOpenFileName(
              QString::null, "Data Files (*.aef )", this,
              "Data File", "Data Files --- Open File");
     textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
             textBrr3->insertParagraph(QString("----------------------------------------------------------------------------------
----------------------------------------------------------"), insPos3);
              setCurs3++;
            textBrr3->setCursorPosition(setCurs3, insPos3);
             insPos3++;
     textBrr3->insertParagraph(QString("CURRENT MESSAGE:  fileName is %1").arg(fileName), insPos3);
              setCurs3++;
   if (!fileName.isEmpty())
           load();

  else
      statusBar()->message(" ERROR:  File`s opening is FAILURE ",2000);

}
                                     *********************************
/* слот загрузки файлов AEF */
void mainForm::load()
{
   setINNAME();
    textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
     textBrr3->insertParagraph(QString("CURRENT MESSAGE:  inname for loadable file -
[%1]").arg(inname), insPos3);
              setCurs3++;
            statusBar()->message( QString("Loading  '%1' ...").
                                  arg(fileName));
            int res_AEF_DataFile = read_AEF_DataFile();
            if(res_AEF_DataFile ==0)
            {
              statusBar()->message(QString(" Data File has successfully read.  It`s closed ('%1')").
```

```
                        arg(fileName));
            }
  else
  {
        statusBar()->message( QString("   File`s loading is FAILURE ( '%1' )").
                              arg(fileName), 3000);
  }
  if   (showLinkedData)
    {
        loadEventCommon();

     }
  showLinkedData = false;
}
                                    ***************************
/* ф-я загрузки связанного файла Event_Common.aef.
 "функция - оболочка" для создания видимой логической связи
 действий в теле слота загрузки файлов AEF (load())*/
void mainForm::loadEventCommon()
{
 int res_AEF_EvCom = read_AEF_EventCommon();
      if (res_AEF_EvCom == 0)
        {
           MESSAGE(19);
         }
         else
          {
             MESSAGE(20);
         }
}

void mainForm::fileExit()
{
  QApplication::exit(0);
}
                                  *********************************
/* ф-я чтения файла архива AEF.
 Вначале вызывается ф-я чтения заголовка
и ,получив корректный заголовок и разрешив выводить
 связанные данные -  showLinkedData = true; - чтение и вывод
информации в таблицу table1*/
int mainForm::read_AEF_DataFile()
{
  int rez = 0;
  FILE* fin;
  textBrr3->setCursorPosition(setCurs3, insPos3);
  insPos3++;
  textBrr3->insertParagraph(QString("CURRENT MESSAGE: calling
read_AEF_DataFile(%1)").arg(inname), insPos3);
   setCurs3++;
     bufReadCount = 0;
     bufMaxCount = CT_NORMILIZE_DEFREADBUFFER;
    TDA_AEF_EventDinWrite* bufReadData = new  TDA_AEF_EventDinWrite [bufMaxCount];
```

```
  if (rez= read_AEF_Header(inname) != 0)
  {
     MESSAGE(1);
     MESSAGE(23);
     fflush(0);
     delete bufReadData;
     return 1;
  }
  else
  {
     MESSAGE(0);
     showLinkedData = true;
     hourseek =0;
  insPos1 = 1;
  setCurs1 =2;
  textBrr1->setText(QString("  Data file: '%1' ").
                          arg(fileName));
  textBrr1->insertParagraph(QString("----------------------------------------------------------------------------------------------------
-------------------------------------------------"), insPos1);
  textBrr1->setCursorPosition(setCurs1, insPos1);
   insPos1++;
   textBrr1->insertParagraph(QString("Header: "), insPos1);
   setCurs1++;
   UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (fileHead->OpenFile.DateTime) , prtime, sizeof(prtime) ,
"%d . %m . %y  %H : %M : %S . %t " ) ;
   MESSAGE(11);
   UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (fileHead->LastWrite.DateTime) , prtime, sizeof(prtime) ,
"%d . %m . %y  %H : %M : %S . %t " ) ;
   MESSAGE(12);
   MESSAGE(13);
   for (int iii = 0; iii < 24; iii ++)
   {
      if (fileHead->SeekFileOfUTCTime[iii] ==0)
      {
            hourseek = 0;
      }
      else
      {
            hourseek = (fileHead->SeekFileOfUTCTime[iii] - 651)  / 36 + 1;
                  }
    textBrr1->setCursorPosition(setCurs1, insPos1);
     insPos1++;
     textBrr1->insertParagraph(QString("          %1   =   %2  ").arg(iii + 1).arg(hourseek) , insPos1);
     setCurs1++;
   }
    textBrr1->setCursorPosition(setCurs1, insPos1);
    insPos1++;
   textBrr1->insertParagraph(QString("  ----------------------------------------------------------------------------------------------------
-----------------------------"), insPos1);
    setCurs1++;

  if ( (fin = fopen ( inname, "rb") )==NULL)
```

```
{
    MESSAGE(14);
    fflush(0);
    delete bufReadData;
    return 1;
 }
int seekpos = sizeof(TDA_AEF_DataHeadFV);
fseek(fin, seekpos, SEEK_SET);
while (!feof(fin))
{
   bufReadCount = fread ( bufReadData, sizeof( TDA_AEF_EventDinWrite),  bufMaxCount, fin );
    table1->setNumRows(bufReadCount);
   for (unsigned long i = 0; i < bufReadCount; i ++)
   {
       table1->setText( i, 0, QString::number(bufReadData[i].hEvent));
       UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (bufReadData[i].ftWriteTime.DateTime) , prtime1,
sizeof(prtime1) , "%d . %m . %y  %H : %M : %S " ) ;
            TempForChar = TM_CharToQString(prtime1, "KOI8-R");
            table1->setText( i, 1, TempForChar);
       UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (bufReadData[i].ftTime.DateTime) , prtime2,
sizeof(prtime2) , "%d . %m . %y  %H : %M : %S " ) ;
            TempForChar = TM_CharToQString(prtime2, "KOI8-R");
            table1->setText( i, 2, TempForChar);
        UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (bufReadData[i].ftActivateTime.DateTime) , prtime3,
sizeof(prtime3) , "%d . %m . %y  %H : %M : %S " ) ;
            TempForChar = TM_CharToQString(prtime3, "KOI8-R");
             table1->setText( i, 3, TempForChar);
        UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (bufReadData[i].ftDeactivateTime.DateTime) ,
prtime4, sizeof(prtime4) , "%d . %m . %y  %H : %M : %S " ) ;
            TempForChar = TM_CharToQString(prtime4, "KOI8-R");
            table1->setText( i, 4, TempForChar);
            table1->setText( i, 5,  QString::number( bufReadData[i]. Value. Data. Value));
         }
       table1->adjustColumn(0);
       table1->adjustColumn(1);
       table1->adjustColumn(2);
       table1->adjustColumn(3);
       table1->adjustColumn(4);
       table1->adjustColumn(5);
          table1->setSorting(TRUE);
      }
fclose( fin );
 }
return 0;
}
                                        ****************************
/* ф-я чтения заголовка файла  ********_Event.aef  */
int mainForm::read_AEF_Header( char * namefile)
{
   bool badHead = false;
  MESSAGE(9);
 FILE* f = NULL;
 tag_TDA_AEF_DataHeadFV* Header= new tag_TDA_AEF_DataHeadFV;
```

```
  if (namefile==NULL)
  {
        MESSAGE(6);
        return 1;
  }
   if (strlen(namefile)==0)
  {
        MESSAGE(7);
        return 1;
  }

  if ( ( f = fopen (namefile, "rb")) ==NULL)
  {
     MESSAGE(4);
    return 1;
}
  fseek (f, 0, SEEK_SET);
 int couread = fread(Header, sizeof(tag_TDA_AEF_DataHeadFV), 1, f);
        if(couread==1){
              if(Header->IdFile!=TDA_AEF_fileEvent)
          {
              badHead = true;
              ERROR_Message( 1, Header);
              }
              if(Header->TypeFile!=TDA_AEF_fileEvent)
              {
              badHead = true;
             ERROR_Message( 2, Header);
            }
              if(Header->VersionFile!=TDA_AEF_VersionFile)
              {
              badHead = true;
              ERROR_Message(3, Header);
              }
              if (badHead)
              {
              ERROR_Message(4, Header);
              Header = NULL;
              delete Header;
              fclose(f);
              return 1;
              }
              else
              {
              fileHead = Header;
              }
        }
        else
        {
            MESSAGE(5);
              Header = NULL;
              delete Header;
              fclose(f);
```

```
                return 1;
            }
           fclose(f);
           return 0;
}
                                    **********************************
/* ф-я выдачи тек-го сообщения о
 ходе анализа файла AEF*/
void mainForm::MESSAGE( int  kodd)
{
  kod = kodd;
 switch (kod)
 {
    case 0:
            textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
            textBrr3->insertParagraph(QString("CURRENT MESSAGE...  The Header is VALID"), insPos3);
             setCurs3++;
           break;
    case 1:
            textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
                textBrr3->insertParagraph(QString("CURRENT MESSAGE...  The Header is INVALID"),
insPos3);
             setCurs3++;
           break;
    case 2:
           textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
            textBrr3->insertParagraph(QString("ERROR:  Reading of file Header"), insPos3);
             setCurs3++;
           break;
    case 3:

     case 4:
           textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
            textBrr3->insertParagraph(QString("ERROR:  file is not opened. Mistake"), insPos3);
             setCurs3++;
           break;
      case 5:
           textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
            textBrr3->insertParagraph(QString("ERROR: dont readed to pHeadBuffer "), insPos3);
             setCurs3++;
           break;
  case 6:
           textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
                textBrr3->insertParagraph(QString("ERROR:  namefile==NULL.
(inname['%1'])").arg(inname), insPos3);
             setCurs3++;
                 break;
```

```
    case 7:
            textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
             textBrr3->insertParagraph(QString("ERROR:  strlen(namefile)==0.
(inname['%1'])").arg(inname), insPos3);
              setCurs3++;
                    break;
   case 8:
            textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
             textBrr3->insertParagraph(QString("ERROR:  pHeadBuf==NULL"), insPos3);
              setCurs3++;
                    break;
  case 9:
            textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
           textBrr3->insertParagraph(QString("            *** Analysis of Header (fileName: %1)
***").arg(inname), insPos3);
              setCurs3++;
                    break;
    case 10:
           textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
          textBrr3->insertParagraph(QString(" CURRENT MESSAGE...  Some mistake caught in program.  IN-
file is: %1)").arg(inname), insPos3);
              setCurs3++;
           break;
    case 11:
           textBrr1->setCursorPosition(setCurs1, insPos1);
           insPos1++;
           textBrr1->insertParagraph(QString(" *******  Open File   -  %1 ").arg(prtime) , insPos1);
          setCurs1++;
              break;
     case 12:
            textBrr1->setCursorPosition(setCurs1, insPos1);
            insPos1++;
             textBrr1->insertParagraph(QString(" *******  Last Write   -  %1 ").arg(prtime) , insPos1);
          setCurs1++;
              break;
     case 13:
            textBrr1->setCursorPosition(setCurs1, insPos1);
            insPos1++;
             textBrr1->insertParagraph(QString(" *******  Seek File  of UTC Time  ( Hour in UTC  = Number
record)"), insPos1);
           setCurs1++;
              break;
     case 14:
             textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
             textBrr3->insertParagraph(QString("     ERROR...open  IN-file ( %1)").arg(inname), insPos3);
            setCurs3++;
           break;
       case 15:
```

```
            textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
             textBrr3->insertParagraph(QString("       *** Analysis of Header (fileName: %1)
***").arg(nameEvCom), insPos3);
            setCurs3++;
            break;
    case 16:
             textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
             textBrr3->insertParagraph(QString("ERROR:  namefile==NULL.
(inname['%1'])").arg(nameEvCom), insPos3);
            setCurs3++;
            break;
     case  17:
           textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
           textBrr3->insertParagraph(QString("ERROR:  strlen(namefile)==0.
(inname['%1'])").arg(nameEvCom), insPos3);
           setCurs3++;
           break;
      case 18:
             textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
            textBrr3->insertParagraph(QString("     ERROR...open  IN-file ( %1)").arg(inname), insPos3);
            setCurs3++;
          break;
     case 19:
             textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
            textBrr3->insertParagraph(QString("CURRENT MESSAGE...  file (%1) is successfully opened.
").arg(nameEvCom), insPos3);
            setCurs3++;
            break;
      case 20:
             textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
            textBrr3->insertParagraph(QString("ERROR:  file (%1) is not opened.
Mistake").arg(nameEvCom), insPos3);
            setCurs3++;
            break;
    case 21:
             textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
            textBrr3->insertParagraph(QString(" ERROR...open  IN-file ( %1)").arg(nameEvCom), insPos3);
            setCurs3++;
          break;
     case 22:
           textBrr3->setCursorPosition(setCurs3, insPos3);
            insPos3++;
            textBrr3->insertParagraph(QString("ERROR:  file (%1) is not opened.  The Header is not
known").arg(nameEvCom),   insPos3);
            setCurs3++;
            break;
```

```
    case 23:
            textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
            textBrr3->insertParagraph(QString(" CURRENT MESSAGE... MAIN data file  has a name like
********_Event.aef "), insPos3);
          setCurs3++;
              break;

    }
}
                                        ***************************
/* ф-я выдачи сообщ-я об ошибке при анализе
 файла архива AEF.*/
void mainForm::ERROR_Message( int k , tag_TDA_AEF_DataHeadFV * pHeadBuf)
{
 switch(k)
 {
 case 1:
   textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
          textBrr3->insertParagraph(QString(" ERROR...read_AEF_Header(%1) : wrong IdFile version!!!
(pHeadBuf->IdFile=0x%2,  TDA_AEF_fileEvent=0x%3)").arg(inname).arg(pHeadBuf->IdFile). arg(
TDA_AEF_fileEvent), insPos3);
            setCurs3++;
           break;
    case 2:
          textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
          textBrr3->insertParagraph(QString(" ERROR...  read_AEF_Header(%1) :wrong TypeFile version!!!
").arg(inname), insPos3);
          setCurs3++;
           break;
    case 3:
           textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
          textBrr3->insertParagraph(QString(" ERROR...  read_AEF_Header(%1) :wrong VersionFile
version!!! ").arg(inname), insPos3);
          setCurs3++;
           break;
    case 4:
            textBrr3->setCursorPosition(setCurs3, insPos3);
           insPos3++;
          textBrr3->insertParagraph (QString(" ERROR...  read_AEF_Header(%1) : file`s header is wrong !!!
").arg(inname), insPos3);
          setCurs3++;
           break;
    }
}
                                        ***************************
/* ф-я копирования имени открыв-го файла
 в глоб.строку TempForChar*/
void mainForm::setINNAME()
{
```

```
 int i = fileName.length() - 1;
 int ii = 0;
 char buf;
 QChar qBuf= ' ';
 while (qBuf !='/')
 {
    qBuf=fileName.at(i);
    i --;
}
 i ++;
 i ++;
 while (i < fileName.length())
 {
    qBuf = fileName.at(i);
    buf = (char) qBuf;
    inname[ii] = buf;
    i ++;
    ii ++;
 }
 inname[ii] = '\0';
}
                                        ***************************
/* ф-я копирования содержимого массива,
  полученного в качестве параметра
 в глоб.строку TempForChar*/
void mainForm::CharToQString( char  tempAr [] )
{
  QString temp = "";
char buf;
QChar qBuf;
   int i = 0;
while ( buf !='\0')
{
   buf = tempAr[i];
   if (buf!='\0')
   {
   qBuf = (QChar) buf;
   temp.at(i) = qBuf;
    }
   i ++;
 }
TempForChar = temp;
}
                                          ************************
/*  функция чтения файла Event_Common.aef.
 textBrr1/3 - компоненты вывода инф-и.
insPos1/3,  setCurs1/3  -  глоб. пер-е для установки
позиции курсора и добавления сообщения в соответ-е
"окошко" вывода инф-и.*/
int mainForm::read_AEF_EventCommon()
{
 textBrr3->setCursorPosition(setCurs3, insPos3);
 insPos3++;
```

```
 textBrr3->insertParagraph(QString("                    *******LINKED FILE
(%1)********").arg(nameEvCom), insPos3);
            setCurs3++;
  int rez =0;
  FILE* finEvCom;
   textBrr3->setCursorPosition(setCurs3, insPos3);
   insPos3++;
   textBrr3->insertParagraph(QString("CURRENT MESSAGE: calling
read_AEF_EventCommon(%1)").arg(nameEvCom), insPos3);
    setCurs3++;
     bufReadCount = 0;
     bufMaxCount = CT_NORMILIZE_DEFREADBUFFER;
     TDA_AEF_EventDescrWrite* bufReadDataEvCom = new  TDA_AEF_EventDescrWrite [bufMaxCount];

  if (rez= read_AEF_EvCom_Header(nameEvCom) != 0)
  {
     MESSAGE(1);
     fflush(0);
     delete bufReadDataEvCom;
     return 1;
  }
  else
  {
     MESSAGE(0);

  textBrr1->setCursorPosition(setCurs1, insPos1);
  insPos1++;
  textBrr1->insertParagraph(QString(" *** Linked Data file:  %1 ***").arg(nameEvCom),  insPos1);
  setCurs1++;
  textBrr1->setCursorPosition(setCurs1, insPos1);
  insPos1++;
  textBrr1->insertParagraph(QString("----------------------------------------------------------------------------------------
"), insPos1);
  setCurs1++;
 textBrr1->setCursorPosition(setCurs1, insPos1);
  insPos1++;
  textBrr1->insertParagraph(QString("          *** Header: "), insPos1);
  setCurs1++;

    memset(&descrServ, 0, 80);
  memset(&nameServ, 0, 80);
  memset(&nameUnit, 0, 80);
  //*************************************
  strcpy(descrServ, (const char*) fileHeadEvCom->DescriptionAEF);
  TempForChar = TM_CharToQString(descrServ,"KOI8-R");
   textBrr1->setCursorPosition(setCurs1, insPos1);
  insPos1++;
  textBrr1->insertParagraph(QString(" *** Server description: %1").arg(qstringtmp[currqstr]),  insPos1);
  setCurs1++;
  strcpy(nameServ, ( const char*) fileHeadEvCom->NameAEF);
  TempForChar = TM_CharToQString(nameServ, "KOI8-R");
   textBrr1->setCursorPosition(setCurs1, insPos1);
  insPos1++;
```

```
  textBrr1->insertParagraph(QString(" *** Server name:   %1").arg(TempForChar),  insPos1);
  setCurs1++;
  strcpy(nameUnit,  (const char*) fileHeadEvCom->NameUnit);
  TempForChar = TM_CharToQString(nameUnit, "KOI8-R");
   textBrr1->setCursorPosition(setCurs1, insPos1);
  insPos1++;
  textBrr1->insertParagraph(QString(" *** Unit name where Server is situated: %1").arg(TempForChar),
insPos1);
  setCurs1++;
   textBrr1->setCursorPosition(setCurs1, insPos1);
   insPos1++;
   textBrr1->setCursorPosition(setCurs1, insPos1);
   insPos1++;
  textBrr1->insertParagraph(QString("  ----------------------------------------------------------------------------------------
"), insPos1);
   setCurs1++;

 if ( (finEvCom = fopen (nameEvCom, "rb"))==NULL)
{
    MESSAGE(21);
    fflush(0);
    delete bufReadDataEvCom;
    return 1;
 }
int seekpos = sizeof(TDA_AEF_HeadFV);
fseek(finEvCom, seekpos, SEEK_SET);
while (!feof(finEvCom))
{
   bufReadCount = fread ( bufReadDataEvCom, sizeof( TDA_AEF_EventDescrWrite),  bufMaxCount,
finEvCom );
   table2->setNumRows(bufReadCount);
   for (unsigned long i = 0; i < bufReadCount; i ++)
   {
       table2->setText( i, 0, QString::number(bufReadDataEvCom[i].hEvent));
      strcpy(DataSource, bufReadDataEvCom[i].szSource);
           TempForChar = TM_CharToQString(DataSource, "KOI8-R");
           if (TempForChar =="")
           {
              TempForChar = QString("No Message (bufReadDataEvCom[i].szSource)");
           }
           else
           table2->setText( i, 1, TempForChar);
            strcpy(MessageStat, bufReadDataEvCom[i].szMessageStat);
           TempForChar = TM_CharToQString(MessageStat, "KOI8-R");
            if (TempForChar =="")
           {
              TempForChar = QString("No Message (bufReadDataEvCom[i].szMessageStat)");
           }
            else
           table2->setText( i, 2, TempForChar);
         }
       table2->adjustColumn(0);
       table2->adjustColumn(1);
```

```
        table2->adjustColumn(2);
            table2->setSorting(TRUE);

       }
fclose( finEvCom );
 }
return 0;
}
                                        ********************************
/* ф-я чтения заголовка файла Event_Common.aef.
 badHead - бул.перем-я утанавливается в "заголовок корректен"(false).
 в случае обнаружения ошибок  - "плохой заголовок" присваиваем true.*/
int mainForm::read_AEF_EvCom_Header( char * namefile)
{
 bool badHead = false;
  MESSAGE(15);
 FILE* f2 = NULL;
 tag_TDA_AEF_HeadFV* HeaderEvCom= new tag_TDA_AEF_HeadFV;
  if (namefile==NULL)
  {
          MESSAGE(16);
          return 1;
  }
   if (strlen(namefile)==0)
  {
          MESSAGE(17);
          return 1;
  }

  if ( ( f2 = fopen (namefile, "rb")) ==NULL)
  {
     MESSAGE(22);
    return 1;
}
  fseek (f2, 0, SEEK_SET);
 int couread = fread(HeaderEvCom, sizeof(tag_TDA_AEF_HeadFV), 1, f2);
          if(couread==1){
                  if(HeaderEvCom->IdFile!=TDA_AEF_fileEventCommon)
             {
                    badHead = true;
                    ERROR_EvCom_Message( 1, HeaderEvCom);
                  }
                  if(HeaderEvCom->TypeFile!=TDA_AEF_fileEventCommon)
                  {
                    badHead = true;
                   ERROR_EvCom_Message( 2, HeaderEvCom);
               }
                  if(HeaderEvCom->VersionFile!=TDA_AEF_VersionFile)
                  {
                    badHead = true;
                    ERROR_EvCom_Message(3, HeaderEvCom);
                  }
                  if (badHead)
```

```
                {
                   ERROR_EvCom_Message(4, HeaderEvCom);
                   HeaderEvCom = NULL;
                   delete HeaderEvCom;
                   fclose(f2);
                   return 1;
                }
                else
                {
                   fileHeadEvCom = HeaderEvCom;
                }
          }
          else
          {
                MESSAGE(5);
                    HeaderEvCom = NULL;
                    delete HeaderEvCom;
                    fclose(f2);
                    return 1;
           }
          namefile = 0;
          fseek(f2, 0, SEEK_END);
          fclose(f2);
          return 0;
}
                                        *********************************
/* ф-я выдачи сооб-я об ошибке при анализе файла
  Event_Common.aef*/
void mainForm::ERROR_EvCom_Message( int k, tag_TDA_AEF_HeadFV * pHeadBufEvCom )
{
  switch(k)
 {
 case 1:
    textBrr3->setCursorPosition(setCurs3, insPos3);
    insPos3++;
    textBrr3->insertParagraph(QString(" ERROR...read_AEF_EvCom_Header(%1) : wrong IdFile version!!!
(pHeadBufEvCom->IdFile=0x%2,
TDA_AEF_fileEventCommon=0x%3)").arg(nameEvCom).arg(pHeadBufEvCom->IdFile). arg(
TDA_AEF_fileEventCommon), insPos3);
             setCurs3++;
            break;
  case 2:
     textBrr3->setCursorPosition(setCurs3, insPos3);
     insPos3++;
     textBrr3->insertParagraph(QString(" ERROR...  read_AEF_EvCom_Header(%1) : wrong TypeFile
version!!! ").arg(nameEvCom), insPos3);
          setCurs3++;
            break;
  case 3:
    textBrr3->setCursorPosition(setCurs3, insPos3);
     insPos3++;
     textBrr3->insertParagraph(QString(" ERROR...  read_AEF_EvCom_Header(%1) :  wrong VersionFile
version!!! ").arg(nameEvCom), insPos3);
```

```
      setCurs3++;
    break;
  case 4:
      textBrr3->setCursorPosition(setCurs3, insPos3);
      insPos3++;
      textBrr3->insertParagraph (QString(" ERROR...  read_AEF_EvCom_Header(%1) : file`s header is
wrong !!! ").arg(nameEvCom), insPos3);
      setCurs3++;
     break;
  }
}
                                        ************************
/*  слот получения связанных данных
таблиц AEF. По значению текущей ячейки
"главной" таблицы, просматривается содержимое
"связанной" таблицы и устанавливается текущей
ячейка связ-й таблицы с найденным значением. В
прот-м случае -выдача соотв-го сообщения*/
void mainForm::getLinkedData()
{
   bool found = false;
  int row = table1->currentRow();
  int val = table1->text( row, 0 ). toInt();
  QString qstrVal = table1->text ( row, 0);
  for (int i = 0; i < table2->numRows(); ++i)
  {
     if(table2->text( i, 0 ) == qstrVal)
     {
            table2->setCurrentCell( i, 0 );
            found = true;
            break;
     }
  }
 if (! found)
  {
    AEF_Mes_NoLinkedData( val, inname);
 }
}
                                      ****************************
 /*  функция выдачи сообщения об отсутствии связанных данных*/
void mainForm::AEF_Mes_NoLinkedData( int val, char * namefile)
{
   textBrr3->setCursorPosition(setCurs3, insPos3);
     insPos3++;
     textBrr3->insertParagraph(QString("         >>>>>  NO LINKED  DATA  for  %1.  A message identifier
(hEvent)  is  %2   <<<<< ").arg(namefile).arg(val),   insPos3);
    setCurs3++;

}
                                          **********************
/* функция изменения просмотра страниц приложения.
 Параметр - имя объекта QAction (нажимаемая кнопка
  меню)*/
```

```
void mainForm::changeView( QAction * action )
{
   if (action == AEF_Arch)
   {
          AEF_HDA_WidgetStack->raiseWidget(0);
            }
   if (action == HDA_Arch)
   {
          AEF_HDA_WidgetStack->raiseWidget(1);
   }

}

//****************** for HDA_Arch ********************
/* функция открытия файлов архива HDA*/
void mainForm::fileOpen_HDA()
{
   clear_HDA_t1();
   clear_HDA_t2();
  fileName  = QFileDialog::getOpenFileName(
              QString::null, "Data Files (*.hda )", this,
              "Data File", "Data Files --- Open File");
     textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
             textBrr5->insertParagraph(QString("----------------------------------------------------------------------------------------
----------------------------------------------------------"), insPos5);
              setCurs5++;
            textBrr5->setCursorPosition(setCurs5, insPos5);
             insPos5++;
     textBrr5->insertParagraph(QString("CURRENT MESSAGE:  fileName is %1").arg(fileName), insPos5);
              setCurs5++;
   if (!fileName.isEmpty())
          load_HDA();

  else
     statusBar()->message(" ERROR:  File`s opening is FAILURE ",2000);

}
                                          ************************
/*  "функция-оболочка" чтения данных из "главного"
   файла архива HDA  и, в удачном случае, загрузка
   данных из "связанного" файла( Source_Common.hda)*/
void mainForm::load_HDA()
{
  setINNAME();
    textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
     textBrr5->insertParagraph(QString("CURRENT MESSAGE:  inname for loadable file -
[%1]").arg(inname), insPos5);
             setCurs5++;
           statusBar()->message( QString("Loading  '%1' ...").
                                arg(fileName));
           int res_HDA_DataFile = read_HDA_DataFile();
```

```
          if(res_HDA_DataFile ==0)
          {
             MESSAGE_HDA(3);
            statusBar()->message(QString(" Data File has successfully read.  It`s closed ('%1')").
                        arg(fileName));
             }
   else
   {
          statusBar()->message( QString("   File`s loading is FAILURE ( '%1' )").
                                arg(fileName), 3000);
   }
   if   (showLinkedData_HDA)
     {
            loadSourceCommon();
      }
   showLinkedData_HDA = false;
}
                                          **********************
/*  функция чтения данных из файла ********_Value.hda.
    UT_Tr_GetPrintTime (...) - ф-я получения времени в формате Unix
   (библиотекаUtilsTr)*/
int mainForm::read_HDA_DataFile()
{
  int rez = 0;
   FILE* fin_HDA;
   textBrr5->setCursorPosition(setCurs5, insPos5);
   insPos5++;
   textBrr5->insertParagraph(QString("CURRENT MESSAGE: calling
read_HDA_DataFile(%1)").arg(inname), insPos5);
    setCurs5++;
     bufReadCount = 0;
      bufMaxCount = CT_NORMILIZE_DEFREADBUFFER;
     TDA_HDA_DataFV* bufReadData_HDA = new  TDA_HDA_DataFV [bufMaxCount];

  if (rez= read_HDA_Header(inname) != 0)
  {
     MESSAGE_HDA(1);
     MESSAGE_HDA(23);
     fflush(0);
     delete bufReadData_HDA;
     return 1;
  }
  else
  {
     MESSAGE_HDA(0);
     showLinkedData_HDA = true;
    hourseek =0;
  insPos4 = 1;
  setCurs4 =2;
 textBrr4->setText(QString("  Data file: '%1' ").
                       arg(fileName));
 textBrr4->insertParagraph(QString("----------------------------------------------------------------------------------------
-------------------------------------------------"), insPos4);
```

```
  textBrr4->setCursorPosition(setCurs4, insPos4);
   insPos4++;
   textBrr4->insertParagraph(QString("Header: "), insPos4);
   setCurs4++;
   UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (fileHead_HDA->OpenFile.DateTime) , prtime,
sizeof(prtime) , "%d . %m . %y  %H : %M : %S . %t " ) ;
   MESSAGE_HDA(11);
   UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (fileHead_HDA->LastWrite.DateTime) , prtime,
sizeof(prtime) , "%d . %m . %y  %H : %M : %S . %t " ) ;
   MESSAGE_HDA(12);
   MESSAGE_HDA(13);
   for (int iii = 0; iii < 24; iii ++)
   {
      if (fileHead_HDA->SeekFileOfUTCTime[iii] ==0)
      {
            hourseek = 0;
      }
      else
      {
            hourseek = (fileHead_HDA->SeekFileOfUTCTime[iii] - 651)  / 36 + 1;
                  }
    textBrr4->setCursorPosition(setCurs4, insPos4);
     insPos4++;
     textBrr4->insertParagraph(QString("          %1   =   %2  ").arg(iii + 1).arg(hourseek) , insPos4);
     setCurs4++;
   }
    textBrr4->setCursorPosition(setCurs4, insPos4);
    insPos4++;
   textBrr4->insertParagraph(QString("  ----------------------------------------------------------------------------------------
------------------------------"), insPos4);
    setCurs4++;

  if ( (fin_HDA = fopen ( inname, "rb") )==NULL)
{
     MESSAGE_HDA(14);
     fflush(0);
     delete bufReadData_HDA;
     return 1;
  }
int seekpos = sizeof(TDA_HDA_DataHeadFV);
fseek(fin_HDA, seekpos, SEEK_SET);
while (!feof(fin_HDA))
{
    bufReadCount = fread ( bufReadData_HDA, sizeof( TDA_HDA_DataFV),  bufMaxCount, fin_HDA );
    table3->setNumRows(bufReadCount);
    for (unsigned long i = 0; i < bufReadCount; i ++) ////change bufReadCount!!!
    {
        table3->setText( i, 0, QString::number(bufReadData_HDA[i].hItem));
        table3->setText( i, 1, QString::number(bufReadData_HDA[i].ValueAttr. Data. Value));
        UT_Tr_GetPrintTime ( (TD_Tr_TimeUnix*) & (bufReadData_HDA[i].ValueAttr.Time.DateTime) ,
prtime1, sizeof(prtime1) , "%d . %m . %y  %H : %M : %S " ) ;
            TempForChar = TM_CharToQString(prtime1, "KOI8-R");
            table3->setText( i, 2, TempForChar);
```

```
        }
       table3->adjustColumn(0);
       table3->adjustColumn(1);
       table3->adjustColumn(2);
           table3->setSorting(TRUE);
      }
fclose( fin_HDA );
 }
return 0;
}
                                        **********************
/*  "функция-оболочка" вызова функции чтения
  файла Source_Common.hda и, в зависимости  от
результатов, выдачи соотв-х сообщений.
Служит для наглядного представлени логики действий в теле
слота load_HDA("загрузка файлов HDA")*/
void mainForm::loadSourceCommon()
{
  int res_HDA_SCom = read_HDA_SourceCommon();
      if (res_HDA_SCom == 0)
        {
           MESSAGE_HDA(19);
         }
         else
          {
             MESSAGE_HDA(20);
         }
}
                                      *************************
/*  функциячтения заголовка файла
********_Value.hda*/
int mainForm::read_HDA_Header( char * namefile)
{
    bool badHead_HDA = false;
  MESSAGE_HDA(9);
 FILE* f_HDA = NULL;
 tag_TDA_HDA_DataHeadFV* Header_HDA= new tag_TDA_HDA_DataHeadFV;
   if (namefile==NULL)
   {
           MESSAGE_HDA(6);
           return 1;
   }
    if (strlen(namefile)==0)
   {
           MESSAGE_HDA(7);
           return 1;
   }

  if ( ( f_HDA = fopen (namefile, "rb")) ==NULL)
  {
     MESSAGE_HDA(4);
    return 1;
}
```

```
 fseek (f_HDA, 0, SEEK_SET);
 int n_read = fread(Header_HDA, sizeof(tag_TDA_HDA_DataHeadFV), 1, f_HDA);
        if(n_read==1){
                if(Header_HDA->IdFile!=TDA_HDA_fileValue)
          {
                   badHead_HDA = true;
                   HDA_ERROR_Message( 1, Header_HDA);
                }
                if(Header_HDA->TypeFile!=TDA_HDA_fileValue)
                {
                   badHead_HDA = true;
                  HDA_ERROR_Message( 2, Header_HDA);
            }
                if(Header_HDA->VersionFile!=TDA_HDA_VersionFile)
                {
                   badHead_HDA = true;
                   HDA_ERROR_Message(3, Header_HDA);
                }
                if (badHead_HDA)
                {
                   HDA_ERROR_Message(4, Header_HDA);
                   Header_HDA = NULL;
                   delete Header_HDA;
                   fclose(f_HDA);
                   return 1;
                }
                else
                {
                   fileHead_HDA = Header_HDA;
                }
        }
        else
        {
             MESSAGE_HDA(5);
                Header_HDA = NULL;
                delete Header_HDA;
                fclose(f_HDA);
                return 1;
         }
        fclose(f_HDA);
        return 0;

}
                                        **********************
/* функция выдачи информации  в "окошки" -
 textBrr4/5 -  текущих сообщений */
void mainForm::MESSAGE_HDA( int mk)
{
 kod = mk;
 switch (kod)
 {
    case 0:
          textBrr5->setCursorPosition(setCurs5, insPos5);
```

```
            insPos5++;
             textBrr5->insertParagraph(QString("CURRENT MESSAGE...  The Header is VALID"), insPos5);
              setCurs5++;
            break;
    case 1:
             textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
            textBrr5->insertParagraph(QString("CURRENT MESSAGE...  The Header is INVALID"), insPos5);
             setCurs5++;
            break;
    case 2:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
             textBrr5->insertParagraph(QString("ERROR:  Reading of file Header"), insPos5);
              setCurs5++;
            break;
    case 3:
             textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
          textBrr5->insertParagraph(QString(" CURRENT MESSAGE...  Wait for a minute!  A lot of data to
read ..."), insPos5);
             setCurs5 ++;
            break;
     case 4:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
             textBrr5->insertParagraph(QString("ERROR:  file is not opened. Mistake"), insPos5);
              setCurs5++;
            break;
      case 5:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
             textBrr5->insertParagraph(QString("ERROR: dont readed to pHeadBuffer "), insPos5);
              setCurs5++;
            break;
  case 6:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
            textBrr5->insertParagraph(QString("ERROR:  namefile==NULL. (inname['%1'])").arg(inname),
insPos5);
             setCurs5++;
                    break;
    case 7:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
             textBrr5->insertParagraph(QString("ERROR:  strlen(namefile)==0.
(inname['%1'])").arg(inname), insPos5);
             setCurs5++;
                    break;
   case 8:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
             textBrr5->insertParagraph(QString("ERROR:  pHeadBuf==NULL"), insPos5);
```

```
            setCurs5++;
                    break;
   case 9:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
           textBrr5->insertParagraph(QString("            *** Analysis of Header (fileName: %1)
***").arg(inname), insPos5);
            setCurs5++;
                    break;
    case 10:
           textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
          textBrr5->insertParagraph(QString(" CURRENT MESSAGE...  Some mistake caught in program.  IN-
file is: %1)").arg(inname), insPos5);
            setCurs5++;
          break;
    case 11:
           textBrr4->setCursorPosition(setCurs4, insPos4);
           insPos4++;
           textBrr4->insertParagraph(QString(" *******  Open File   -  %1 ").arg(prtime) , insPos4);
          setCurs4++;
              break;
     case 12:
           textBrr4->setCursorPosition(setCurs4, insPos4);
           insPos4++;
            textBrr4->insertParagraph(QString(" *******  Last Write   -  %1 ").arg(prtime) , insPos4);
          setCurs4++;
             break;
     case 13:
           textBrr4->setCursorPosition(setCurs4, insPos4);
           insPos4++;
            textBrr4->insertParagraph(QString(" *******  Seek File  of UTC Time  ( Hour in UTC  = Number
record)"), insPos4);
          setCurs4++;
             break;
     case 14:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
            textBrr5->insertParagraph(QString("     ERROR...open  IN-file ( %1)").arg(inname), insPos5);
           setCurs5++;
          break;
       case 15:
           textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
            textBrr5->insertParagraph(QString("       *** Analysis of Header (fileName: %1)
***").arg(nameSCom), insPos5);
           setCurs5++;
           break;
    case 16:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
            textBrr5->insertParagraph(QString("ERROR:  namefile==NULL.
(inname['%1'])").arg(nameSCom), insPos5);
```

```
            setCurs5++;
            break;
     case  17:
           textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
           textBrr5->insertParagraph(QString("ERROR:  strlen(namefile)==0.
(inname['%1'])").arg(nameSCom), insPos5);
           setCurs5++;
           break;
      case 18:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
           textBrr5->insertParagraph(QString("     ERROR...open  IN-file ( %1)").arg(inname), insPos5);
           setCurs5++;
          break;
     case 19:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
           textBrr5->insertParagraph(QString("CURRENT MESSAGE...  file (%1) is successfully opened.
").arg(nameSCom), insPos5);
           setCurs5++;
           break;
      case 20:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
           textBrr5->insertParagraph(QString("ERROR:  file (%1) is not opened. Mistake").arg(nameSCom),
insPos5);
           setCurs5++;
           break;
    case 21:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
           textBrr5->insertParagraph(QString(" ERROR...open  IN-file ( %1)").arg(nameSCom), insPos5);
           setCurs5++;
          break;
     case 22:
           textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
           textBrr5->insertParagraph(QString("ERROR:  file (%1) is not opened.  The Header is not
known").arg(nameSCom),   insPos5);
           setCurs5++;
           break;
    case 23:
            textBrr5->setCursorPosition(setCurs5, insPos5);
           insPos5++;
            textBrr5->insertParagraph(QString(" CURRENT MESSAGE... MAIN data file  has a name like
********_Value.hda "), insPos5);
          setCurs5++;
              break;

    }
}
```

****

```
/*  функция выдачи сообщений об ошибках при анализе
 файла *********_Value.hda*/
void mainForm::HDA_ERROR_Message( int er, tag_TDA_HDA_DataHeadFV * pHeadBuf_HDA )
{
 switch(er)
 {
 case 1:
    textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
           textBrr5->insertParagraph(QString(" ERROR...read_HDA_Header(%1) : wrong IdFile version!!!
(pHeadBuf_HDA->IdFile=0x%2,  TDA_HDA_fileValue=0x%3)").arg(inname).arg(pHeadBuf_HDA->IdFile).
arg( TDA_HDA_fileValue), insPos5);
             setCurs5++;
            break;
     case 2:
           textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
           textBrr5->insertParagraph(QString(" ERROR...  read_HDA_Header(%1) :wrong TypeFile version!!!
").arg(inname), insPos5);
           setCurs5++;
            break;
     case 3:
            textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
           textBrr5->insertParagraph(QString(" ERROR...  read_HDA_Header(%1) :wrong VersionFile
version!!! ").arg(inname), insPos5);
           setCurs5++;
            break;
     case 4:
             textBrr5->setCursorPosition(setCurs5, insPos5);
            insPos5++;
           textBrr5->insertParagraph (QString(" ERROR...  read_HDA_Header(%1) : file`s header is wrong !!!
").arg(inname), insPos5);
           setCurs5++;
            break;
    }
}
                                        **************************
/*функция чтения файла Source_Common.hda.
 textBrr4/5 - объекты типа QTextBrowser - "окошки"
для вывода сообщений*/
int mainForm::read_HDA_SourceCommon()
{
   /*      arrays for fields Sourse_Common */
   char nameObj[80];
   memset(&nameObj, 0, 80);
   char nameAttr[80];
   memset(&nameAttr, 0, 80);
   char descr[80];
   memset(&descr, 0, 80);
   unsigned char allMod = ' ';
   char unitMeasure[80];
   memset(&unitMeasure, 0, 80);
```

```
    unsigned char stepped = ' ';
    char printFormat[80];
    memset(&printFormat, 0, 80);
    char nodeName[80];
    memset(&nodeName, 0, 80);
    /* ************* */
   textBrr5->setCursorPosition(setCurs5, insPos5);
  insPos5++;
  textBrr5->insertParagraph(QString("                    *******LINKED FILE
(%1)********").arg(nameSCom), insPos5);
  setCurs5++;
   int rez =0;
   FILE* finSCom;
    textBrr5->setCursorPosition(setCurs5, insPos5);
    insPos5++;
    textBrr5->insertParagraph(QString("CURRENT MESSAGE: calling
read_HDA_SourceCommon(%1)").arg(nameSCom), insPos5);
     setCurs5++;
       bufReadCount = 0;
       bufMaxCount = CT_NORMILIZE_DEFREADBUFFER;
      TDA_HDA_ATTRIBUTEFV* bufReadDataSCom = new  TDA_HDA_ATTRIBUTEFV [bufMaxCount];

   if (rez= read_HDA_SCom_Header(nameSCom) != 0)
   {
      MESSAGE_HDA(1);
      fflush(0);
      delete bufReadDataSCom;
      return 1;
   }
   else
   {
      MESSAGE_HDA(0);

   textBrr4->setCursorPosition(setCurs4, insPos4);
   insPos4++;
   textBrr4->insertParagraph(QString(" *** Linked Data file:  %1 ***").arg(nameSCom),  insPos4);
   setCurs4++;
   textBrr4->setCursorPosition(setCurs4, insPos4);
   insPos4++;
   textBrr4->insertParagraph(QString("----------------------------------------------------------------------------------
"), insPos4);
   setCurs4++;
  textBrr4->setCursorPosition(setCurs4, insPos4);
   insPos4++;
   textBrr4->insertParagraph(QString("          *** Header: "), insPos4);
   setCurs4++;
   memset(&descrServ, 0, 80);
   memset(&nameServ, 0, 80);
   memset(&nameUnit, 0, 80);
   strcpy(descrServ, (const char* )  fileHeadSCom->DescriptionHDA);
   strcpy(nameServ, (const char* ) fileHeadSCom->NameHDA);
   strcpy(nameUnit, (const char*) fileHeadSCom->NameUnit);
```

```
  textBrr4->setCursorPosition(setCurs4, insPos4);
  insPos4++;
  textBrr4->insertParagraph(QString(" *** Server description: %1").arg(descrServ),  insPos4);
  setCurs4++;
  textBrr4->setCursorPosition(setCurs4, insPos4);
  insPos4++;
  textBrr4->insertParagraph(QString(" *** Server name: %1").arg(nameServ),  insPos4);
  setCurs4++;
  textBrr4->setCursorPosition(setCurs4, insPos4);
  insPos4++;
  textBrr4->insertParagraph(QString(" *** Unit name where Server is situated: %1").arg(nameUnit),
insPos4);
  setCurs4++;
   textBrr4->setCursorPosition(setCurs4, insPos4);
   insPos4++;
  textBrr4->insertParagraph(QString("  ----------------------------------------------------------------------------------------
"), insPos4);
   setCurs4++;

 if ( (finSCom = fopen (nameSCom, "rb"))==NULL)
{
    MESSAGE_HDA(21);
    fflush(0);
    delete bufReadDataSCom;
    return 1;
 }
int seekpos = sizeof(TDA_HDA_HeadFV);
fseek(finSCom, seekpos, SEEK_SET);
while (!feof(finSCom))
{
   bufReadCount = fread ( bufReadDataSCom, sizeof( TDA_HDA_ATTRIBUTEFV),  bufMaxCount, finSCom
);
   table4->setNumRows(bufReadCount);
   for (unsigned long i = 0; i < bufReadCount; i ++)
   {
       table4->setText( i, 0, QString::number(bufReadDataSCom[i].hItem));//
      strcpy(nameObj, bufReadDataSCom[i].NameObj);
           TempForChar = TM_CharToQString(nameObj, "KOI8-R");
           table4->setText( i, 1, TempForChar);
      strcpy(nameAttr, bufReadDataSCom[i].NameAtr);
           TempForChar = TM_CharToQString(nameAttr, "KOI8-R");
           table4->setText( i, 2, TempForChar);
       strcpy(descr, bufReadDataSCom[i].Description);
           TempForChar = TM_CharToQString(descr, "KOI8-R");
           table4->setText( i, 3, TempForChar);
           allMod = bufReadDataSCom[i].AllModifications;
           ucharToQString(allMod);
           table4->setText( i, 4, TempForChar);
          table4->setText( i, 5, QString::number(bufReadDataSCom[i].RangePercent));
        strcpy(unitMeasure, bufReadDataSCom[i].UnitEng);
           TempForChar = TM_CharToQString(unitMeasure, "KOI8-R");
           table4->setText( i, 6, TempForChar);
           stepped = bufReadDataSCom[i].Stepped;
```

```
            ucharToQString(stepped);
            table4->setText( i, 7, TempForChar);
            table4->setText( i, 8, QString::number(bufReadDataSCom[i].NormalMin));
           table4->setText( i, 9, QString::number(bufReadDataSCom[i].NormalMax));
            table4->setText( i, 10, QString::number(bufReadDataSCom[i].EntryLimitLow));
            table4->setText( i, 11, QString::number(bufReadDataSCom[i].EntryLimitHigh));
            table4->setText( i, 12, QString::number(bufReadDataSCom[i].TypeAttr));
         strcpy(printFormat, bufReadDataSCom[i].PrintFormat);
            TempForChar = TM_CharToQString(printFormat, "KOI8-R");
            table4->setText( i, 13, TempForChar);
           table4->setText( i, 14, QString::number((int) bufReadDataSCom[i].DataType));
          table4->setText( i, 15, QString::number(bufReadDataSCom[i].RefTimePeriod));
          table4->setText( i, 16, QString::number(bufReadDataSCom[i].CallbackTimeout));
          table4->setText( i, 17, QString::number(bufReadDataSCom[i].IPAdressNODE));
          table4->setText( i, 18, QString::number(bufReadDataSCom[i].IPPortNODE));
        strcpy(nodeName, bufReadDataSCom[i].NODE_NAME);
            TempForChar = TM_CharToQString(nodeName, "KOI8-R");
            table4->setText( i, 19, TempForChar);
          }
        table4->adjustColumn(0);
        table4->adjustColumn(1);
        table4->adjustColumn(2);
        table4->adjustColumn(3);
        table4->adjustColumn(4);
        table4->adjustColumn(5);
        table4->adjustColumn(6);
        table4->adjustColumn(7);
        table4->adjustColumn(8);
        table4->adjustColumn(9);
        table4->adjustColumn(10);
        table4->adjustColumn(11);
        table4->adjustColumn(12);
        table4->adjustColumn(13);
        table4->adjustColumn(14);
        table4->adjustColumn(15);
        table4->adjustColumn(16);
        table4->adjustColumn(17);
        table4->adjustColumn(18);
        table4->adjustColumn(19);
    table4->setSorting(TRUE);
       }
fclose( finSCom );
  }
return 0;
}
                                        **********************
/*функция чтения заголовка
   файла Source_Common.hda.
  Параметр - указатель на имя
  данного файла*/
int mainForm::read_HDA_SCom_Header( char * namefile)
{
  bool badHead = false;
```

```
 MESSAGE_HDA(15);
FILE* f2 = NULL;
tag_TDA_HDA_HeadFV* HeaderSCom= new tag_TDA_HDA_HeadFV;
 if (namefile==NULL)
 {
        MESSAGE_HDA(16);
        return 1;
 }
  if (strlen(namefile)==0)
 {
        MESSAGE_HDA(17);
        return 1;
 }

 if ( ( f2 = fopen (namefile, "rb")) ==NULL)
 {
    MESSAGE_HDA(22);
   return 1;
}
 fseek (f2, 0, SEEK_SET);
int couread = fread(HeaderSCom, sizeof(tag_TDA_HDA_HeadFV), 1, f2);
        if(couread==1){
                if(HeaderSCom->IdFile!=TDA_HDA_fileSourceCommon)//////////////////////
           {
                   badHead = true;
                   ERROR_SCom_Message( 1, HeaderSCom);
                }
                if(HeaderSCom->TypeFile!=TDA_HDA_fileSourceCommon)
                {
                   badHead = true;
                  ERROR_SCom_Message( 2, HeaderSCom);
             }
                if(HeaderSCom->VersionFile!=TDA_HDA_VersionFile)
                {
                   badHead = true;
                   ERROR_SCom_Message(3, HeaderSCom);
                }
                if (badHead)
                {
                   ERROR_SCom_Message(4, HeaderSCom);
                   HeaderSCom = NULL;
                   delete HeaderSCom;
                   fclose(f2);
                   return 1;
                }
                else
                {
                   fileHeadSCom = HeaderSCom;
                }
        }
        else
        {
             MESSAGE_HDA(5);
```

```
                HeaderSCom = NULL;
                delete HeaderSCom;
                fclose(f2);
                return 1;
          }
        namefile = 0;
        fseek(f2, 0, SEEK_END);
        fclose(f2);
        return 0;
}
                                        **********************
/*функция выдачи сообщения об ошибке при
 анализе файла Source_Common.hda.
 Второй параметр - указатель на структуру заголовка
 данного файла*/
void mainForm::ERROR_SCom_Message( int er, tag_TDA_HDA_HeadFV * pHeadBufSCom)
{
  switch(er)
 {
 case 1:
    textBrr5->setCursorPosition(setCurs5, insPos5);
    insPos5++;
    textBrr5->insertParagraph(QString(" ERROR...read_HDA_SCom_Header(%1) : wrong IdFile version!!!
(pHeadBufSCom->IdFile=0x%2,
TDA_HDA_fileSourceCommon=0x%3)").arg(nameSCom).arg(pHeadBufSCom->IdFile). arg(
TDA_HDA_fileSourceCommon), insPos5);
            setCurs5++;
           break;
 case 2:
     textBrr5->setCursorPosition(setCurs5, insPos5);
     insPos5++;
     textBrr5->insertParagraph(QString(" ERROR...  read_HDA_SCom_Header(%1) : wrong TypeFile
version!!! ").arg(nameSCom), insPos5);
          setCurs5++;
           break;
  case 3:
    textBrr5->setCursorPosition(setCurs5, insPos5);
     insPos5++;
     textBrr5->insertParagraph(QString(" ERROR...  read_HDA_SCom_Header(%1) :  wrong VersionFile
version!!! ").arg(nameSCom), insPos5);
     setCurs5++;
   break;
  case 4:
      textBrr5->setCursorPosition(setCurs5, insPos5);
      insPos5++;
      textBrr5->insertParagraph (QString(" ERROR...  read_HDA_SCom_Header(%1) : file`s header is
wrong !!! ").arg(nameSCom), insPos5);
      setCurs5++;
     break;
  }
                                                }
                                        ************************
/*копирование объекта uchar в глоб.строку типа
```

```
 QString*/
void mainForm::ucharToQString( unsigned char buf)
{
QString temp = "";
QChar qBuf = (QChar) buf;
temp.at(0) = qBuf;
TempForChar = temp;
}
                                        **************************
/*слоты "прорисовки" определенной
 страницы приложения*/
void mainForm::raise_AEF()
{
    AEF_HDA_WidgetStack->raiseWidget(0);
}


void mainForm::raise_HDA()
{
    AEF_HDA_WidgetStack->raiseWidget(1);
}
                                          ***********************
/*слоты  "очистки" содержимого талблиц путем
 поочередной очистки ячеек таблицы*/
void mainForm::clear_AEF_t1()
{
 for ( int row = 0; row < table1->numRows(); row ++)
 {
   for (int col = 0;  col <6;  col ++)
   table1->clearCell( row, col );
 }
}


void mainForm::clear_AEF_t2()
{
 for ( int row = 0; row < table2->numRows(); row ++)
 {
   for (int col = 0;  col <3;  col ++)
   table2->clearCell( row, col );
 }
}


void mainForm::clear_HDA_t1()
{
 for ( int row = 0; row < table3->numRows(); row ++)
 {
   for (int col = 0;  col <3;  col ++)
   table3->clearCell( row, col );
 }
}
```

```
void mainForm::clear_HDA_t2()
{
 for ( int row = 0; row < table4->numRows(); row ++)
 {
    for (int col = 0;  col <20;  col ++)
    table4->clearCell( row, col );
 }
}
                                        **********************
/* слот получения связанных данных таблиц архива AEF и HDA*/
/*фиксируем текущее начение главной таблицы и ищем в подчиненной
таблице ячейку с этим значением.Найдя  - устанавливаем  ее текущей,
 нет - выдаем соответствующее сообщение*/
void mainForm::getLinkedData_HDA()
{
   bool found = false;
  int row = table3->currentRow();
  int val = table3->text( row, 0 ). toInt();
  QString qstrVal = table3->text ( row, 0);
  for (int i = 0; i < table4->numRows(); ++i)
  {
      if(table4->text( i, 0 ) == qstrVal)
      {
              table4->setCurrentCell( i, 0 );
              found = true;
              break;
      }
  }
 if (! found)
  {
     AEF_Mes_NoLinkedData( val, inname);
 }
 }
```

ДОДАТОК Б.
Структури заголовків та записів файлів архівів

ЗАГОЛОВОК

struct tag_TDA_AEF_HeadFV

Поле	Описание
unsigned char IsAccess	Признак наличия информации в записи
unsigned long IdFile	Код идентификации типа файла
unsigned long TypeFile	Тип файла
double VersionFile	Версия структуры файла
TDA_UnixZoneTimeFV OpenFile	Время начала ведения файла
unsigned long Duration	Длительность ведения файла
unsigned long IdentCode	Код идентификации
unsigned charDescriptionAEF[80]	Описание сервера
unsigned char NameAEF[80]	Имя сервера
double VersionAEF	Версия сервера
unsigned char NameUnit[80]	Имя узла где расположен сервер
unsigned char NameFile[80]	Имя файла без пути доступа
unsigned long NumberAttribute	Количество атрибутов в архиве
TDA_UnixZoneTimeFV WriteFileToRemote	Время копирования архива на внешний носитель
TDA_UnixZoneTimeFV WriteFileFromRemote	Время копирования архива с внешнего носителя

ЗАПИСЬ

struct tag_TDA_AEF_EventDescrWrite

Поле	Описание
unsigned long hEvent	Идентификатор сообщения
char szSource[TDA_MAX_STRING80]	Источник данных
char szMessageStat[TDA_MAX_STRING255]	Сообщение (статическая часть)
char szMessageDin[TDA_MAX_STRING255]	Сообщение (динамическая часть)
char dwEventType	тип сообщения
unsigned long dwEventCategory	Код категории состояния
unsigned long dwSeverity	Код важности состояния
char szStateName[TDA_MAX_STRING80]	Имя состояния
char szSubStateName[TDA_MAX_STRING80]	Имя подсостояния
char szCategoryName[TDA_MAX_STRING80]	Имя категории
char szAreaName[TDA_MAX_STRING80]	Имя области
char szAreaDescription[TDA_MAX_STRING80]	Идентификатор области
char szExpression[TDA_MAX_STRING255]	Строка выражения которое определяет активное подсостояние или пустая строка
unsigned char bAckRequired	Истина - означает что сообщение требует квитирования
char NormState	Признак, что состояние относится к нормальным (=1)

Рисунок Б.1 – Структури заголовка та записів файлу Event_Common.aef

ЗАГОЛОВОК

struct tag_TDA_AEF_DataHeadFV

Поле	Описание
unsigned char IsAccess	Признак наличия информации в записи (!= 0 информация доступна)
unsigned long IdFile	Код идентификации типа файла
unsigned long TypeFile	Тип файла
double VersionFile	Версия структуры файла
TDA_UnixZoneTimeFV OpenFile	Время начала ведения файла
TDA_UnixZoneTimeFV LastWrite	Время последней записи в файл
unsigned long Duration	Длительность ведения файла
unsigned long IdentCode	Код идентификации
unsigned char DescriptionAEF[80]	Описание сервера
unsigned char NameAEF[80]	Имя сервера
double VersionAEF	Версия сервера
unsigned char NameUnit[80]	Имя узла где расположен сервер
unsigned long Port	Номер порта работы сервера
unsigned char NameFile[80]	Имя файла без пути доступа
unsigned unsigned short ChkSumField	Контрольная сумма записи (0 - не используется)

ЗАПИСЬ

struct tag_TDA_AEF_EventDinWrite

Поле	Описание
unsigned long hEvent	Идентификатор сообщения
unsigned long dwState	Состояние Сообщения
TDA_UnixZoneTimeFV ftWriteTime	Время записи сообщения в архив
TDA_UnixZoneTimeFV ftTime	Время перехода в новое состояние
TDA_ArchValueAtrributeFV Value	Значение источника данных
TDA_UnixZoneTimeFV ftActivateTime	Время активизации подсостояния (или состояния)
TDA_UnixZoneTimeFV ftDeactivateTime	Время последней деактивизации этого состояния
unsigned long NumberAttribute	Количество атрибутов, сопровождающих сообщение
double Version	Версия структуры
unsigned short ChkSumField	Контрольная сумма записи (0 - не используется)

Рисунок Б.2 - Структури заголовка та записів файлу DDMMYYYY_Event.aef

ЗАГОЛОВОК

struct tag_TDA_HDA_HeadFV

Поле	Описание
unsigned char IsAccess	Признак наличия информации в записи (!= 0 информация доступна)
unsigned long IdFile	Код идентификации типа файла
unsigned long TypeFile	Тип файла
double VersionFile	Версия структуры файла
TDA_UnixZoneTimeFV OpenFile	Время начала ведения файла
unsigned long Duration	Длительность ведения файла
unsigned long IdentCode	Код идентификации
unsigned char DescriptionHDA[80]	Описание сервера
unsigned char NameHDA[80]	Имя сервера
double VersionHDA	Версия сервера
unsigned char NameUnit[80]	Имя узла где расположен сервер
unsigned long Port	Номер порта работы сервера
unsigned char NameFile[80]	Имя файла без пути доступа
unsigned char Description[80]	Комментарий, записываемый при сохранении для переноса на внешний носитель
unsigned short ChkSumField	Контрольная сумма записи (0 - не используется)

ЗАПИСЬ

struct tag_TDA_HDA_ATTRIBUTEFV

Поле	Описание
unsigned long hItem	Идентификатор Итема
char NameObj[TDA_MAX_STRING80]	Имя источника данных (объект)
char NameAtr[TDA_MAX_STRING80]	Имя источника данных (атрибут)
char Description[TDA_MAX_STRING80]	Описание параметра
unsigned char AllModifications	Признак: фиксировать все изменения атрибута (=1) или только с периодом (=0)
double RangePercent	Процент изменения величины требуемый для фиксации в архиве (от диапазона)
char UnitEng[TDA_MAX_STRING80]	Единицы измерения
unsigned char Stepped	Способ отображения данных (1 - ступенчато (по умолчанию), 0 - интерполированно)
double NormalMin	Минимально допустимое значение (по умолчанию совпадает со шкалой)
double NormalMax	Максимально допустимое значение (по умолчанию совпадает со шкалой)
double EntryLimitLow	Начало диапазона
double EntryLimitHigh	Конец диапазона
unsigned long TypeAttr	Тип параметра (0 - аналоговый(по умолчанию), 1 - дискретный)
unsigned short DataType	Тип данных
double Version	Версия структуры (для контроля целостности файла)

Рисунок Б.3 - Структури заголовка та записів файлу Source_Common.hda

ЗАГОЛОВОК

struct tag_TDA_HDA_DataHeadFV

Поле	Описание
unsigned char IsAccess	Признак наличия информации в записи (!= 0 информация доступна)
unsigned long IdFile	Код идентификации типа файла
unsigned long TypeFile	Тип файла
double VersionFile	Версия структуры файла
TDA_UnixZoneTimeFV OpenFile	Время начала ведения файла
TDA_UnixZoneTimeFV LastWrite	Время последней записи в файл
unsigned long Duration	Длительность ведения файла
unsigned long IdentCode	Код идентификации
unsigned char DescriptionHDA[80	Описание сервера
unsigned char NameHDA[80]	Имя сервера
double VersionHDA	Версия сервера
unsigned char NameUnit[80	Имя узла где расположен сервер
unsigned long Port	Номер порта работы сервера
unsigned long NumberAttribute	Количество записей данных в файле
unsigned short ChkSumField	Контрольная сумма записи (0 - не используется)

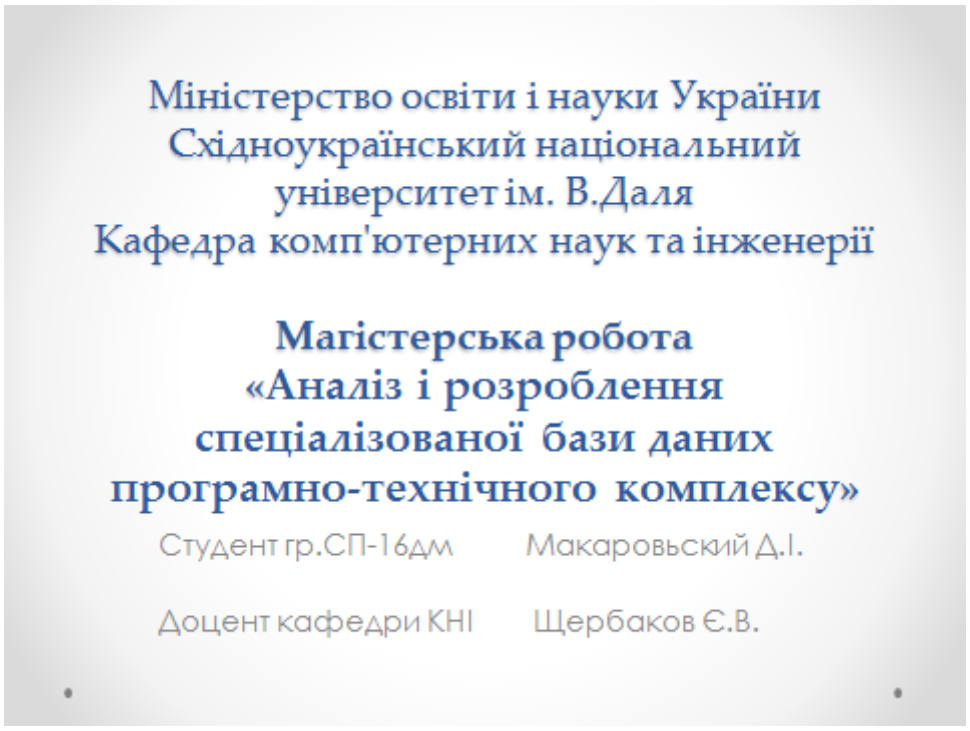
ЗАПИСЬ

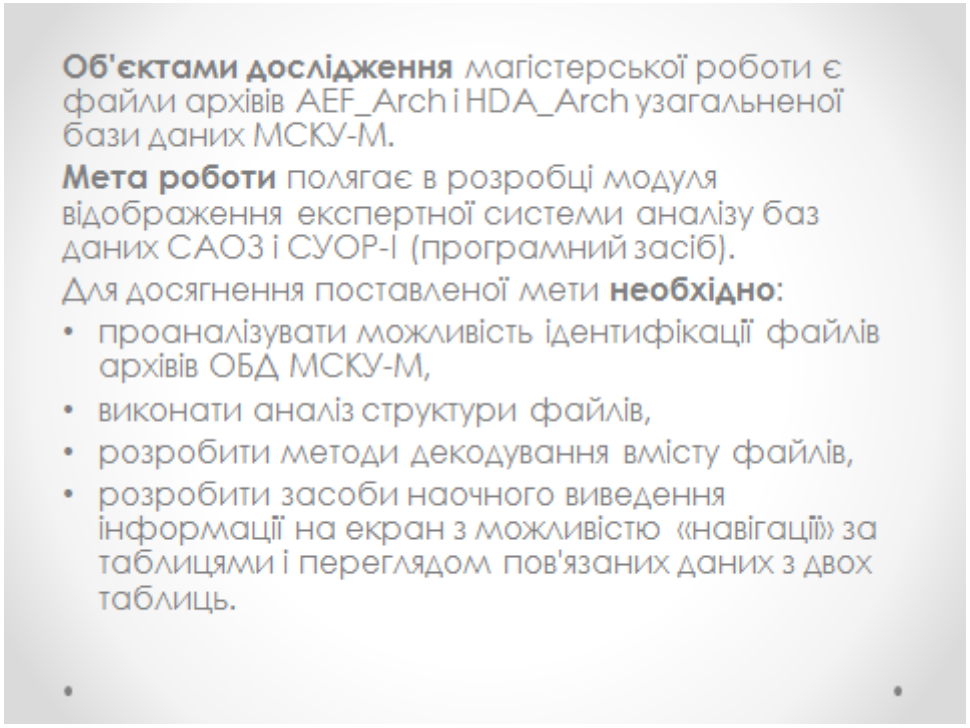
struct tag_TDA_HDA_DataFV

Поле	Описание
unsigned char IsAccess	Признак наличия информации в записи (!= 0 информация доступна)
unsigned long hItem	Идентификатор Итема
TDA_ArchValueAtrributeFV ValueAttr	Данные о параметре (значение состояние время изменения)
unsigned short ChkSumField	Контрольная сумма записи (0 - не используется)

Рисунок Б.4 - Структури заголовка та записів файлу DDMMYYYY_Value.hda

ДОДАТОК В.
Електронні плакати



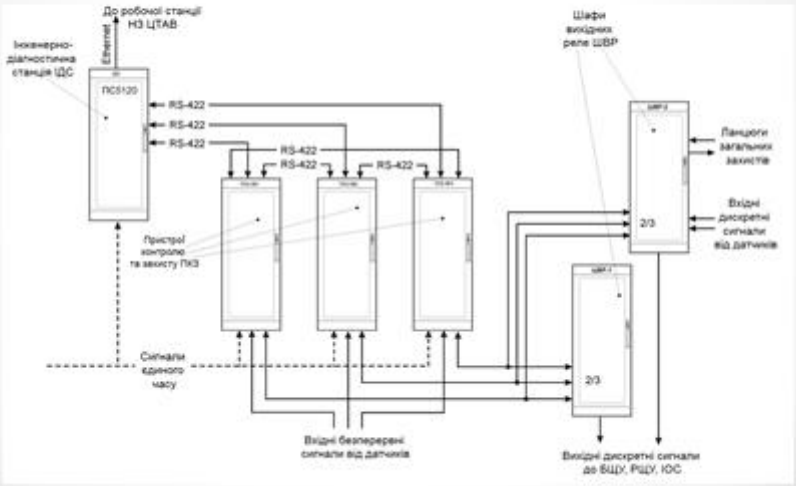


В даний час на ЗАТ «СНВО «Імпульс» (далі - підприємство) ведуться розробки з модернізації програмно-технічного комплексу технологічних захистів (далі - КТЗ-І).

КТЗ-І орієнтований на застосування в якості обладнання АСКТП при реконструкції діючих і створенні нових управляючих систем безпеки і систем нормальної експлуатації.

Одна з областей практичного застосування КТЗ-І - система аварійного охолодження активної зони

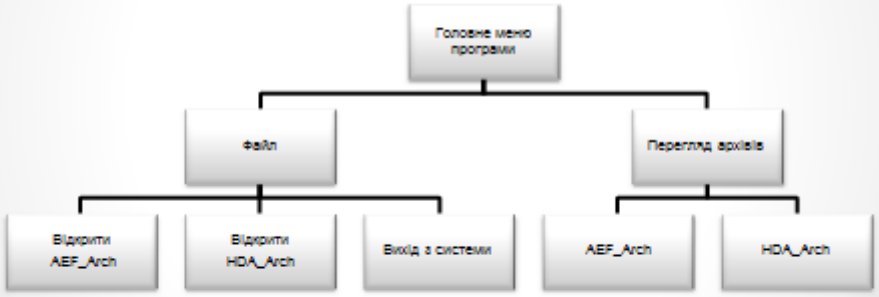
Структура трьох каналів САОЗ



Основні функції КТЗ-І:

- прийом і первинну обробку вхідних безперервних і дискретних сигналів, визначення поточних значень технологічних параметрів;
- порівняння поточних значень технологічних параметрів із заданими граничними значеннями (уставками), виконання обчислювальних і логічних операцій відповідно до алгоритмів спрацьовування захистів;
- логічну обробку в кожному каналі (ПКЗ) сигналів, які формуються в ньому самому і в двох інших каналах (ПКЗ) при ідентифікації вихідної події спрацьовування захисту;
- формування і видачу команд захисту при виході контрольованих технологічних параметрів за межі встановлених граничних значень (уставок) і / або за іншими умовами спрацьовування захисту;
- формування і видачу на БЩК і РЩК сигналів при виході поточних значень технологічних параметрів за межі встановлених граничних значень і при виявленні несправностей технічних засобів, які контролюють вихід параметрів за межі уставок;
- формування і видачу в ІОС сигналів про спрацювання САОЗ за окремими видами захистів;
- прийом сигналів від мережі єдиного всесвітнього координованого часу, відлік часу за допомогою вбудованих засобів, синхронізацію результатів незалежного рахунку з мережею єдиного часу.

Відповідно до завдання на розробку дипломного проекту, схема проектованого модуля повинна являти собою ієрархічну деревоподібну структуру, що описує процедури введення, обробки і виведення даних.



При проектуванні даної системи весь процес був розбитий на кілька частин:

- проектування меню програми;
- створення функцій і слотів екранного інтерфейсу;
- розробка функцій і слотів, за допомогою яких вирішуються всі завдання, які стоять перед проектом;
- підключення до проекту необхідних бібліотек;
- комплексне налагодження та тестування програми в процесі дослідної експлуатації.

Схема алгоритму функціонування слоту, виклику відповідної сторінки додатку

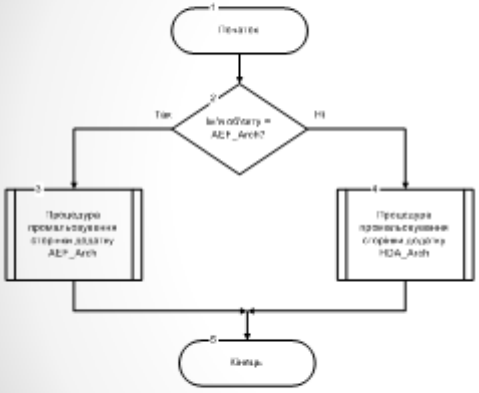


Схема алгоритму функціонування слоту відкриття файлів

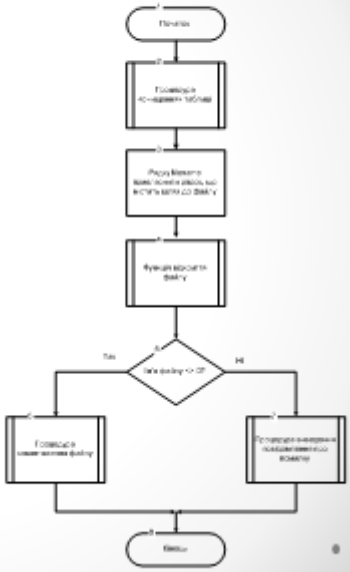
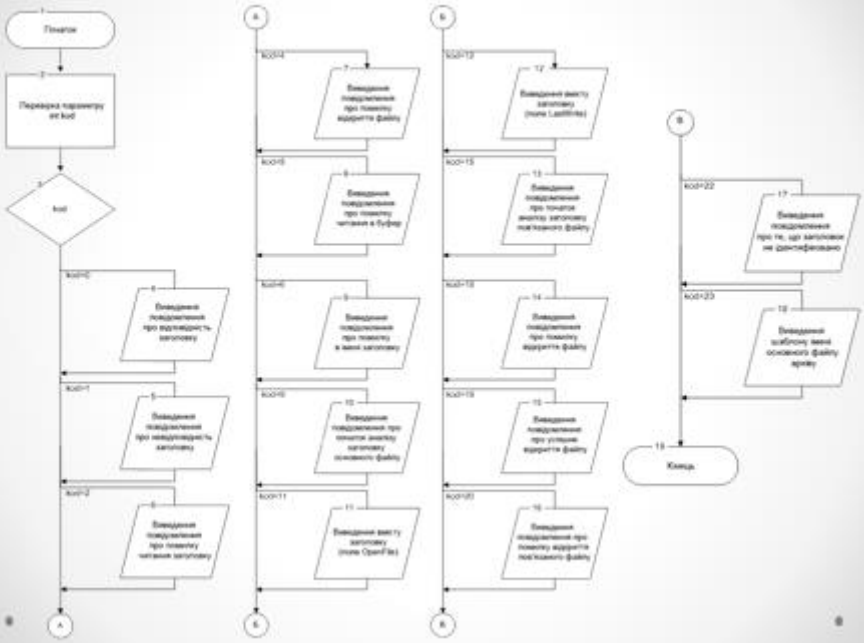


Схема алгоритму функції виведення поточних повідомлень



Відповідно до завдання, для виведення на екран вмісту файлів в програмі використовувалися компоненти типу QTable (таблиці).

Також розроблені дві функції завантаження файлів для обох архівів. Функції розроблені для надання програмі читабельності, тобто за допомогою даних функцій в програмі проглядається послідовність і логіка дій. Функції корисні при модифікації програми.

Схема алгоритму приведення таблиць у початковий стан

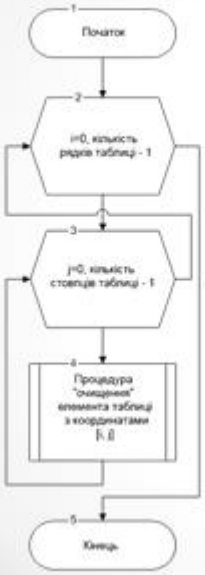
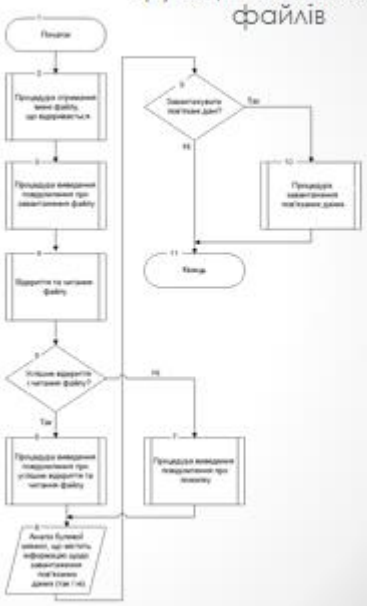


Схема алгоритму функції завантаження файлів



Відповідно до завдання, проектом розроблено дві функції, призначені для ідентифікації заголовків основних файлів.

Вміст полів заголовку перевіряється на відповідність певним значенням. У разі невідповідності - на екран виводиться повідомлення про помилку, яке вказує неправильне значення поля заголовку файлу. Так само в віконце виведення видається еталонне значення для даного поля.

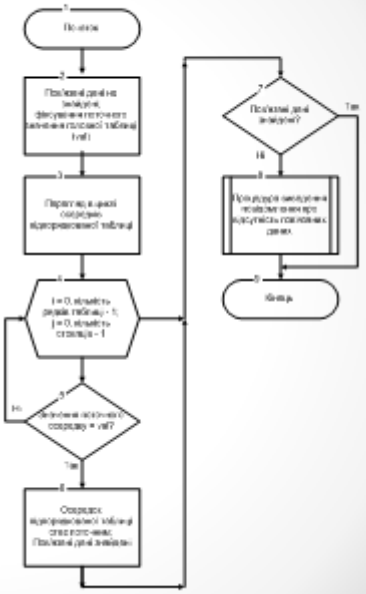
Схема алгоритму роботи з архівами

Відповідно до завдання, проектом розроблено два слота для відображення пов'язаних даних.

Зазначені слоти викликаються на виконання, коли відбувається виділення якого-небудь осередку в основній таблиці або переміщення по рядках таблиці. Слід нагадати, що відповідно до завдання, на кожній сторінці додатка розміщено по дві таблиці: одна для відображення даних з основного файлу, а інша - для даних зі зв'язаного файлу.

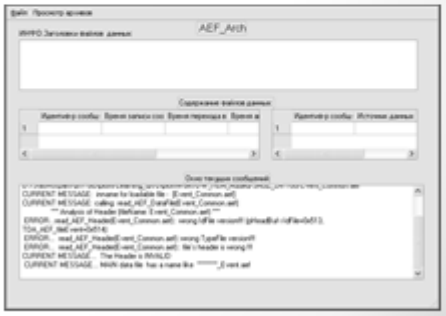
Також слід зазначити, що для кожного з зазначених слотів в середовищі проектування створено по два з'єднання, що зв'язують сигнал головної таблиці з даними слотом. Зазначені сигнали визначені в класі QTable.

Схема алгоритму слота відображення на екрані пов'язаних даних



Вид робочого вікна після запуску програми

Вікно програми після відкриття файлу Event_Common



Вікно програми з виведеними в таблиці даними із пов'язаних файлів архіву HDA_Arch та повідомленнями про хід аналізу заголовків і файлів

Висновок

Відповідно до завдання, був розроблений модуль відображення експертної системи аналізу баз даних САОЗ.

Проектом передбачена можливість ідентифікації файлів архівів УБД МСКУ-М, аналіз структури файлів, декодування вмісту файлів і наочний вивід інформації на екран з можливістю «навігації» за таблицями і переглядом пов'язаних даних з двох таблиць. Реалізовано двохсторінковий інтерфейс, що дозволяє переглядати і зіставляти дані з обох архівів: подій і повідомлень про порушення (AEF_Arch) і архіву параметрів технологічного процесу (HDA_Arch).