Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра ПЗ

Практична робота №3

з дисципліни «Операційні системи»

Виконав: ст. 2ПІ-22Б Середа Д. О.

Перевірив: доцент Рейда О. М.

Вінниця – 2025

**ПРАКТИЧНА РОБОТА №3**

**Тема**: Розробка функціонального класу і алгоритмів роботи програмного застосунку

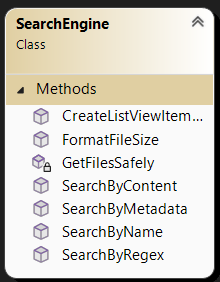


Рисунок 1 – Діаграма функціонального класу

Клас SearchEngine — це ref class (клас у C++/CLI), який містить статичні методи для пошуку файлів і допоміжні функції.

SearchByName шукає файли за назвою.

static array<String^>^ SearchByName(String^ directory, String^ pattern, OnProgressEvent^ onProgress, BackgroundWorker^ worker);

Рисунок 2 – Метод “SearchByName”

SearchByContent шукає файли за вмістом.

static array<String^>^ SearchByContent(String^ directory, String^ content, OnProgressEvent^ onProgress, BackgroundWorker^ worker);

Рисунок 3 – Метод “SearchByContent”

SearchByRegex шукає файли за регулярним виразом.

static array<String^>^ SearchByRegex(String^ directory, String^ regexPattern, String^ searchType, OnProgressEvent^ onProgress, BackgroundWorker^ worker);

Рисунок 4 – Метод “SearchByRegex”

SearchByMetadata шукає файли за метаданими.

static array<String^>^ SearchByMetadata(String^ directory, String^ metadataValue, String^ metadataType, OnProgressEvent^ onProgress, BackgroundWorker^ worker);

Рисунок 5 – Метод “SearchByMetadata”

FormatFileSize форматує розмір файлу в КБ, МБ або ГБ.

static String^ FormatFileSize(Int64 fileSize) {

double sizeInKB = fileSize / 1024.0;

if (sizeInKB < 1000) {

return String::Format(L"{0:F2} КБ", sizeInKB);

}

double sizeInMB = sizeInKB / 1024.0;

if (sizeInMB < 1000) {

return String::Format(L"{0:F2} МБ", sizeInMB);

}

double sizeInGB = sizeInMB / 1024.0;

return String::Format(L"{0:F2} ГБ", sizeInGB);}

Рисунок 6 – Метод “FormatFileSize”

CreateListViewItemForFile створює елемент ListViewItem для відображення файлу.

static ListViewItem^ CreateListViewItemForFile(String^ filePath) {

FileInfo^ fileInfo = gcnew FileInfo(filePath);

ListViewItem^ item = gcnew ListViewItem(fileInfo->Name);

item->SubItems->Add(fileInfo->FullName);

item->SubItems->Add(FormatFileSize(fileInfo->Length));

return item;

}

Рисунок 7 – Метод “CreateListViewItemForFile”

GetFilesSafely рекурсивно отримує список файлів із директорії, обробляючи винятки. Метод позначений приватним, що обмежує його використання поза класом.

private:

static void GetFilesSafely(String^ directory, String^ pattern, List<String^>^ fileList, BackgroundWorker^ worker) {

if (worker->CancellationPending) return;

try {

array<String^>^ files = Directory::GetFiles(directory, pattern);

fileList->AddRange(files);

}

catch (UnauthorizedAccessException^) {

}

catch (Exception^ ex) {

System::Diagnostics::Debug::WriteLine("GetFilesSafely Error in " + directory + ": " + ex->Message);

}

try {

array<String^>^ subDirs = Directory::GetDirectories(directory);

for each (String ^ subDir in subDirs) {

if (worker->CancellationPending) return;

GetFilesSafely(subDir, pattern, fileList, worker);

}

}

catch (UnauthorizedAccessException^) {

}

catch (Exception^ ex) {

System::Diagnostics::Debug::WriteLine("GetFilesSafely SubDir Error in " + directory + ": " + ex->Message);

}

}

Рисунок 8 – Метод “GetFilesSafely”

Усі методи статичні, тобто клас не потребує створення екземпляра.

Використовується делегат OnProgressEvent для звітування про прогрес.

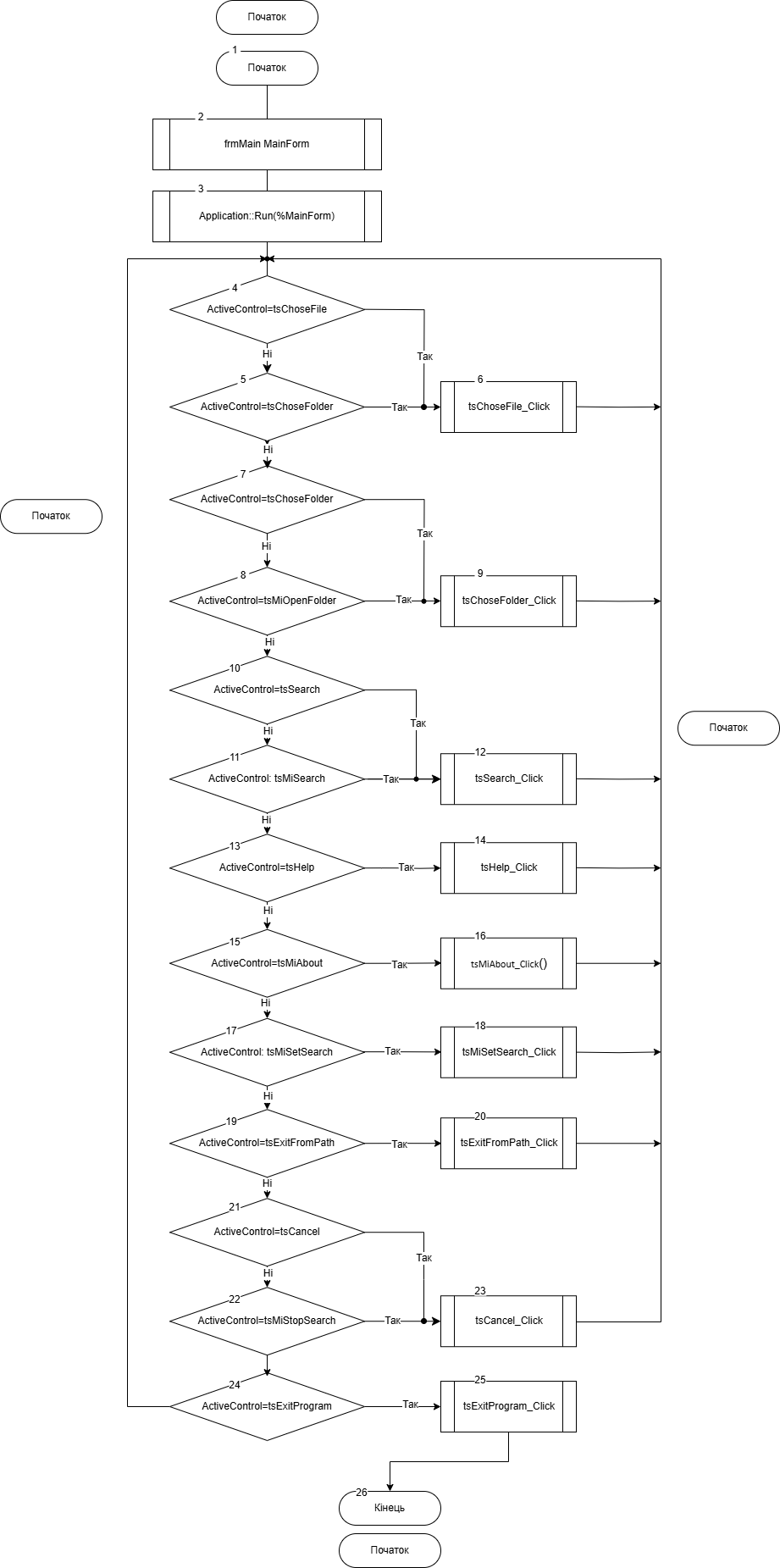


Рисунок 9 – Блок-схема роботи застосунку

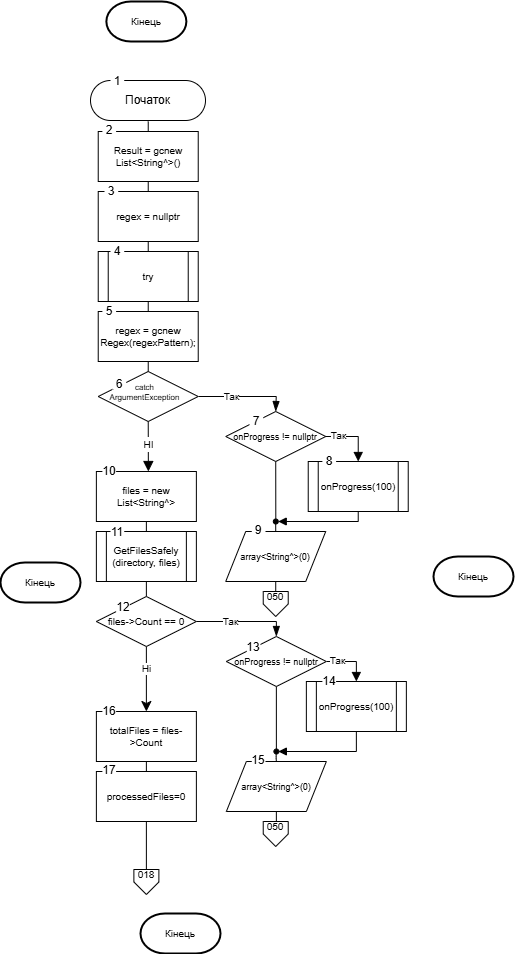


Рисунок 10 – Блок-схема алгоритму пошуку за регулярними виразами

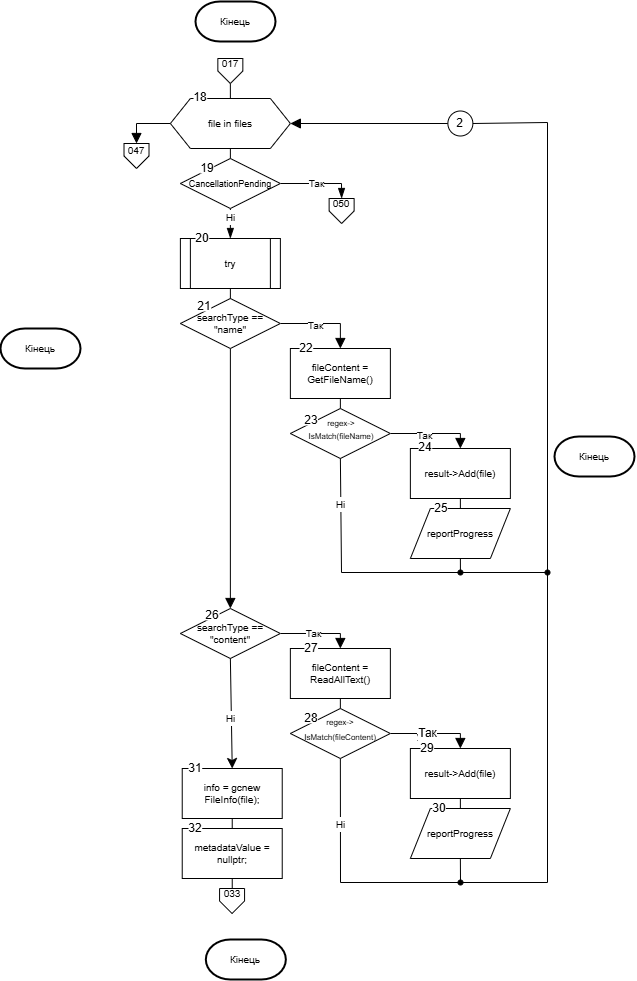


Рисунок 11 – Продовження блок-схеми алгоритму пошуку за регулярними виразами

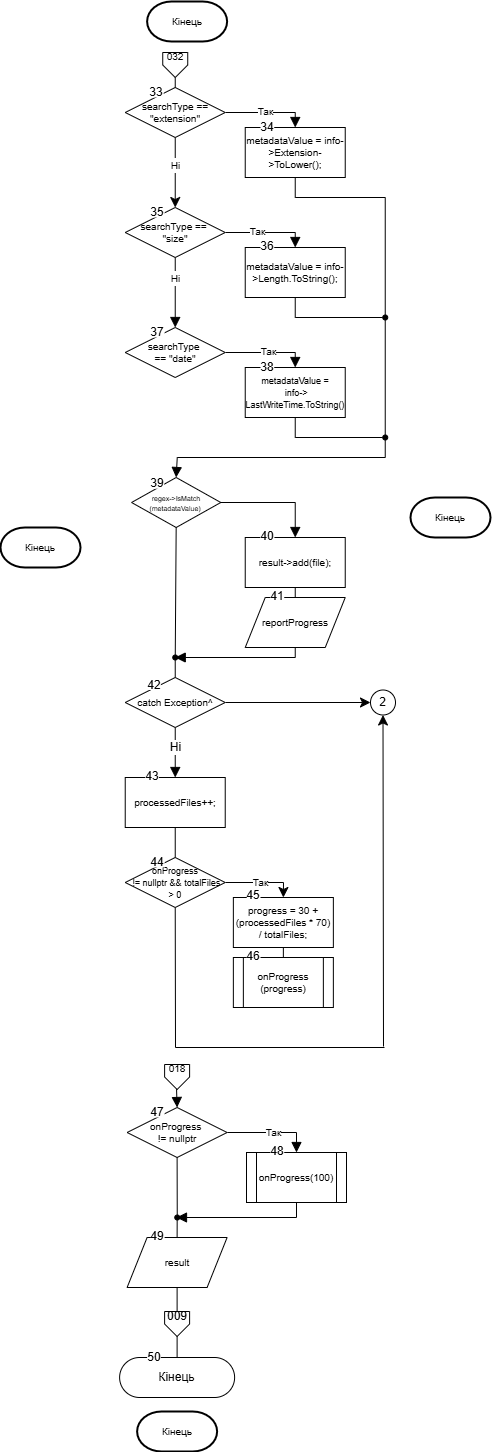


Рисунок 12 – Продовження блок-схеми алгоритму пошуку за регулярними виразами