

Відокремлений структурний підрозділ
«ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ імені Олеся ГОНЧАРА»

Циклова комісія програмної інженерії

КУРСОВИЙ ПРОЄКТ

з навчальної дисципліни

«Основи програмування та алгоритмічні мови»

на тему: Облік видачі кредитів банком

(вказати тему курсового проєкту)

Студента III курсу ПЗ-22-1 групи
галузь знань 12 Інформаційні технології
спеціальності 121 Інженерія

програмного забезпечення

Бєдін Т. В.

(прізвище та ініціали студента)

Керівник

викладач Т.С.Сайко

Національна шкала

Кількість балів:

Оцінка ECTS:

Члени комісії

(підпис)

О.М. Романовський

(прізвище та ініціали)

С.С.Ланська

(підпис)

(прізвище та ініціали)

Т.С.Сайко

(підпис)

(прізвище та ініціали)

м. Дніпро – 2024 рік

Відокремлений структурний підрозділ
«ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ імені Олеся ГОНЧАРА»

Циклова комісія програмної інженерії

ЗАТВЕРДЖУЮ

Голова циклової комісії ПІ

_____ С.С.Ланська
" ____ " _____ 2024 р.

ЗАВДАННЯ
на виконання курсового проекту

з дисципліни «Основи програмування та алгоритмічні мови»
студенту Бєдіну Тимуру Валерійовичу
(прізвище, ім'я та по батькові)
Відділення Програмної інженерії
Спеціальність 121 Інженерія програмного забезпечення
Курс ІІІ Група (шифр) ПЗ-22-1
1 Тема проекту Облік видачі кредитів банком

2 Початкові дані: місто проживання, ПІБ клієнта, номер телефону клієнта,
тіло кредиту, сума до сплати (вираховується за формулою тіло кредиту*1.1),
щомісячний платіж (вираховується за формулою тіло кредиту*1.1/12)., дата
видачі кредиту.

Розглянуто і ухвалено на засіданні предметної комісії
Програмної інженерії
Протокол № 1 від 30.08.2024 р.

Голова комісії ПІ _____ С.С.Ланська
Керівник КП _____ Т.С.Сайко
Завдання до виконання
одержав студент _____ В. Т. Бєдін
(підпис) (ініціали та прізвище)

Дата видачі 09 вересня 2024 р.
Термін виконання 10 листопада 2024 р.

ЗМІСТ

ВСТУП	3
1 ПОСТАНОВКА ЗАДАЧІ	4
2 ОПИС ПРОГРАМИ.....	6
2.1 Опис мови програмування	6
2.2 Опис середовища програмування.....	9
2.3 Опис полів структури запису у файлі	11
2.4 Опис використаних бібліотек	13
2.5 Опис структури програми	15
2.6 Детальний опис функцій	18
2.7 Опис функцій перевірки контролю вхідних даних	22
2.8 Схема зв'язків між функціями.....	31
3 ОПИС ПРОЦЕДУРИ НАЛАГОДЖЕННЯ ПРОГРАМИ	34
4 ІНСТРУКЦІЯ З ПІДГОТОВКИ ДАНИХ ТА КОРИСТУВАННЯ ПРОГРАМОЮ	40
ВИСНОВКИ	52
ДОДАТОК А	55
ДОДАТОК Б.....	56
ДОДАТОК В.....	86

					КП.ПЗ.221.03.ПЗ			
Змн	Лист	№ докум.	Підпис	Дат	Облік видачі кредитів банком	Лім.	Арк.	Аркушів
Розроб.		Бедін Т. В.						
Перевір.		Сайко Т.С.					2	85
Реценз.						ВСП «ФКРКМ ДНУ»		
Н. контр.								
Затверд.								

ВСТУП

Темою курсового проекту є «Ведення обліку видачі банківських кредитів». Метою його створення є автоматизація процесів обліку, аналізу та зберігання інформації про видачу кредитів. У сучасному світі, де фінансові операції відіграють важливу роль, автоматизація таких процесів є необхідною умовою для ефективної роботи банківської системи.

Комп'ютерні технології проникли в усі сфери діяльності людини, включаючи фінансову сферу. Вони дозволяють обробляти великий обсяг інформації, зберігати її в безпечному вигляді та швидко виконувати необхідні операції. Використання комп'ютерів у фінансових установах значно спрощує процеси обліку та управління фінансовими даними, підвищуючи їхню точність і швидкість.

Проект передбачає розробку системи, яка дозволяє створювати, редагувати, видаляти записи про кредити, виконувати пошук, сортування та різноманітні запити. Система може зберігати такі дані: інформацію про клієнта, суму кредиту, строк погашення, відсоткову ставку та інші деталі. Особливістю проекту є забезпечення перевірки валідності введених даних, унікальності записів та надійної обробки помилок.

Метою створення програмного продукту є спрощення роботи банківських працівників, забезпечення точності даних та підвищення ефективності обліку кредитів. Цей проект може бути використаний у банківській сфері для автоматизації внутрішніх процесів, зменшення ризику людських помилок та покращення обслуговування клієнтів.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				3
Змн.	Арк.	№ докум.	Підпис	Дат		

1 ПОСТАНОВКА ЗАДАЧІ

Темою курсового проєкту є «Облік видачі кредитів банком».

Файл повинен містити:

- 1) Місто проживання.
- 2) ПІБ клієнта.
- 3) Номер телефону клієнта.
- 4) Тіло кредиту.
- 5) Сума до сплати (вираховується за формулою тіло кредиту*1.1).
- 6) Щомісячний платіж (вираховується за формулою тіло кредиту*1.1/12).
- 7) Дата видачі кредиту.

Для роботи з файлом програма повинна мати головне меню з наступними пунктами:

- 1) Створення нового файлу.
- 2) Перегляд існуючого файлу.
- 3) Додавання даних у файл.
- 4) Редагування записів (як цілком, так і за окремими полями).
- 5) Видалення записів (за двома ознаками: за номером телефону та за ПІБ клієнта).
- 6) Сортування інформації (за двома ознаками: за містом проживання та за тілом кредиту).
- 7) Виконання запитів:
 - виведення на екран загальної кількості кредитів за ПІБ клієнта з детальною інформацією;
 - знаходження максимального та мінімального тіла кредиту;
 - пошук усіх кредитів, які видані за певний період;
 - знаходження усіх боржників;
 - розрахунок загальної суми прибутку банку.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				4
Змн.	Арк.	№ докум.	Підпис	Дат		

Кожний з пунктів головного меню оформити у вигляді окремої підпрограми (функції). Для «спілкування» функцій між собою організувати передачу параметрів в обидві сторони (не використовувати глобальні змінні).

При створенні нового файлу даних організувати підтвердження на видалення існуючого файлу.

Усі дані, що мають вводитися з клавіатури, повинні бути максимально перевірені на коректність введення.

Перегляд вмісту існуючого файлу та будь-яких запитів виконати у вигляді таблиці.

Додаткові, але не обов'язкові, вимоги:

1) При виконанні пошуку у файлі деякого рядкового значення, наприклад, ПІБ, надати користувачу можливість пошуку:

- за введенням повністю (наприклад, *Іванов К.Е.*);
- за частковим введенням початкових літер (наприклад, *Ів*);
- за частковим введенням частини рядка (наприклад, *ов К*).

2) Рядкові дані (наприклад, ПІБ, будь-яка назва тощо) зберігати у файлі так, щоб як би користувач не ввів їх з клавіатури, вони завжди повинні бути знайдені. Наприклад, ПІБ може бути уведено з пробілами на початку, наприкінці, декількома пробілами між прізвищем та по батькові, великими чи малими літерами, чи по батькові без крапок.

3) Використання масивів для зберігання постійних даних. Наприклад, перелік менеджерів фірми, групи товарів, процентів виплат, акцій тощо.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				5
Змн.	Арк.	№ докум.	Підпис	Дат		

2 ОПИС ПРОГРАМИ

2.1 Опис мови програмування

Програма написана на мові C++.

C++ – мова програмування високого рівня з підтримкою декількох парадигм програмування:

1) Процедурне програмування: код «в стилі С».

Код, написаний «в стилі С», передбачає використання процедур (функцій), які обробляють дані через змінні. Такий підхід дозволяє розбивати програму на невеликі модулі, кожен із яких виконує певну задачу. Це підвищує читабельність коду та спрощує його налагодження.

2) Об'єктно-орієнтоване програмування.

C++ значно розширює можливості об'єктно-орієнтованого програмування, включаючи такі концепції:

- класи: дозволяють об'єднувати дані та методи для їх обробки в єдині структури;

- спадкування: забезпечує повторне використання коду шляхом створення нових класів на основі існуючих;

- поліморфізм: дозволяє реалізувати різні методи з однаковим іменем у різних класах, що спрощує розширення функціоналу;

- інкапсуляція: обмежує доступ до певних даних класу, забезпечуючи їх захист від небажаних змін;

- віртуальні функції: дозволяють реалізувати динамічне визначення методу, що забезпечує виклик правильного методу для об'єкта, на який вказує базовий вказівник або посилання. Це сприяє досягненню поліморфізму та спрощує роботу з ієрархією класів.

3) Узагальнене програмування, шаблони функцій і класів.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				6
Змн.	Арк.	№ докум.	Підпис	Дат		

Використання шаблонів функцій і класів дозволяє створювати універсальний код, який можна застосовувати до різних типів даних. Це зменшує дублювання коду та забезпечує масштабованість.

4) Функціональне програмування:

- лямбда-функції: дають змогу створювати анонімні функції, які можна використовувати у вигляді аргументів для інших функцій;
- замикання: дозволяють створювати функції, які зберігають контекст свого створення;
- рекурсія: широко використовується для вирішення задач, які можна розділити на підзадачі того ж типу.

5) Функтори.

Ці об'єкти дозволяють виконувати функції через перевантажений оператор (). Вони корисні для передачі параметрів або стану в алгоритми STL, наприклад `std::for_each`.

6) Безіменні функції;

Безіменні функції, або лямбда-функції, є ключовою особливістю мови C++ для функціонального програмування. Вони дозволяють визначати функцію безпосередньо у місці виклику. Це особливо корисно для передачі коротких фрагментів коду як аргументів до інших функцій, наприклад, у стандартних алгоритмах `std::sort` або `std::for_each`. Лямбда-функції підтримують захоплення змінних з оточення, що робить їх дуже зручними для створення функцій, які працюють зі станом, доступним у їхньому контексті.

7) Замикання.

Замикання у C++ реалізуються за допомогою лямбда-функцій і дають можливість захоплювати змінні з оточення (як за значенням, так і за посиланням). Це дозволяє створювати функції, які можуть утримувати доступ до змінних навіть після завершення свого контексту.

	Вик.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		7

Особливості замикань це те, що вони забезпечують простий спосіб доступу до змінних поза тілом функції. Замикання роблять код компактным і читабельним, дозволяючи уникати використання глобальних змінних.

8) Генеративное програмування: метапрограмування на шаблонах.

Генеративне програмування є потужною парадигмою, яка дозволяє виконувати обчислення на етапі компіляції. У C++ це реалізується через шаблони, які дають можливість створювати гнучкі та ефективні рішення, що генерують код динамічно залежно від вхідних параметрів.

Властивістю метапрограмування є можливість скорочувати повторюваний код, автоматизуючи його генерацію. Воно використовується у великих фреймворках, таких як Boost і STL, для створення універсальних і масштабованих рішень.

9) Мова C++ є ефективною.

Ефективність C++ полягає в його низькорівневій природі, що дозволяє програмістам оптимізувати використання ресурсів. Відсутність неявних накладних витрат робить C++ популярним вибором для розробки програм, які вимагають високої продуктивності.

Переваги мови «C++» :

- можливість управління пам'яттю через new та delete;
- низькорівневий доступ до апаратури через вказівники;
- висока швидкість виконання завдяки компіляції в машинний код.

Приклади використання C++:

- розробка операційних систем (Windows, Linux);
- ігрові рушії (Unity, Unreal Engine);
- високопродуктивні сервери та наукові симуляції.

Ці властивості роблять C++ незамінним для створення програмного забезпечення, де продуктивність і ефективність є критично важливими.

Одна з фундаментальних ідей мов C і C++ – відсутність неявних накладних витрат, які присутні в інших більш високорівневих мовах програмування:

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		8

- програміст сам вибирає рівень абстракції, на якому писати кожен окрему частину програми;
- можна реалізовувати критичні по продуктивності ділянки програми максимально ефективно;
- C++ є основною мовою для розробки додатків з комп'ютерною графікою (наприклад ігри);
- розробка системного програмного забезпечення та драйверів пристроїв, завдяки прямому доступу до пам'яті та апаратних ресурсів;
- можливість створення високопродуктивних серверів і мережових програм, де швидкість обробки запитів має вирішальне значення;
- наукові розрахунки та симуляції, завдяки підтримці точних математичних операцій і оптимізації складних алгоритмів;
- розробка вбудованих систем та IoT-пристроїв, завдяки компактності й високій продуктивності генерованого коду;
- використання C++ для написання бібліотек, які забезпечують високу продуктивність для інших мов програмування (наприклад, TensorFlow, написаний на C++ із доступом до Python API);
- можливість ефективного використання багатоядерних процесорів через паралельне програмування, що реалізується за допомогою бібліотек, таких як OpenMP або стандартних потоків C++;
- підтримка управління апаратними ресурсами в реальному часі, що важливо для розробки авіаційних, автомобільних і медичних систем.

C++ залишається незамінним інструментом для розробки продуктивного та ефективного програмного забезпечення завдяки своїй гнучкості, високій швидкості та можливості оптимізації. Це робить мову одним із найкращих виборів для критично важливих задач у програмуванні.

2.2 Опис середовища програмування

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				9
Змн.	Арк.	№ докум.	Підпис	Дат		

Для реалізації програми використовується Microsoft Visual Studio – потужне інтегроване середовище розробки (IDE), яке надає інструменти для:

- планування та створення графічного чи консольного інтерфейсу користувача;
- написання коду та його перевірки на коректність завдяки вбудованому відладчику;
- аналізу продуктивності коду для виявлення вузьких місць у програмі;
- автоматизованого тестування.

Visual Studio є універсальним інструментом для створення різних типів додатків:

- мобільні програми для Android, iOS, Windows;
- веб-додатки на основі популярних технологій (ASP.NET, Angular, React);
- ігрові проекти, що використовують DirectX для графічної візуалізації.

Можливості VS:

- робота з різними мовами: IDE підтримує C++, Python, C#, JavaScript, що дозволяє розробникам легко інтегруватися в багатомовні проекти;
- система підказок: допомагає уникати помилок на етапі написання коду завдяки вбудованому аналізу;
- інструменти для командної роботи: інтеграція з Git дозволяє організувати спільну розробку.

Visual Studio підтримує хмарну розробку, забезпечуючи легкий доступ до ресурсів Azure для створення веб-сервісів та мікросервісів. IDE також дозволяє створювати програми для IoT (інтернет речей), включаючи інтеграцію з Office, SharePoint і Hololens.

Додаткові можливості C++ у Visual Studio:

- автоматизація рутинних завдань, таких як форматування коду;
- налагодження великих проєктів завдяки розширеним можливостям моніторингу стану змінних, стеку викликів та обробки винятків;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		10

- інтеграція з інструментами аналізу коду, як-от Static Code Analysis, що дозволяє уникнути потенційних вразливостей;
- оптимізація роботи з багатоядерними процесорами завдяки підтримці багатопоточності та паралельного програмування.

Visual Studio та C++ ідеально підходять для реалізації програм, де потрібні висока швидкість виконання, ефективність, а також гнучкість у реалізації різних підходів до програмування.

2.3 Опис полів структури запису у файлі

Структура полів запису у файлі згідно постановки задачі складається з наступних полів:

- city – назва міста, з якого клієнт бере кредит. Тип даних char[20], максимальна довжина – 20 символів. Формат введення: текстовий рядок, що починається з великої літери, без цифр чи спеціальних символів;
- fullName – повне ім'я клієнта у форматі "Прізвище І.О.". Тип даних char[30], максимальна довжина – 30 символів. Формат забезпечує зручність для пошуку та ідентифікації клієнтів у базі даних;
- phoneNumber – контактний номер клієнта. Тип даних char[15], зберігається у форматі "+38XXXXXXXXXX" (де X – цифра). Введення перевіряється на відповідність шаблону регулярного виразу;
- loanBody – основна сума кредиту, яку взяв клієнт. Тип даних float, діапазон значень – від 1000 до 100000. Величина зберігається з двома знаками після коми для точності фінансових розрахунків;
- totalAmount – загальна сума до сплати, включаючи відсотки (110% від loanBody). Тип даних float. Значення обчислюється автоматично програмою та зберігається у структурі для зручного відображення та подальшого використання;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		11

– monthlyPayment – щомісячний платіж, розрахований як загальна сума, поділена на 12 місяців. Тип даних float. Дозволяє клієнту зрозуміти фінансове навантаження щомісяця;

– loanDate – дата оформлення кредиту у форматі "DD.MM.YYYY". Тип даних char[11]. Формат введення перевіряється програмою на відповідність і коректність дати;

– isPaid – статус кредиту (виплачений чи ні). Тип даних bool. Значення true відповідає виплаченому кредиту, false – наявності боргу.

Логічна структура полів:

– поля city, fullName, phoneNumber використовуються для ідентифікації клієнта;

– поля loanBody, totalAmount, monthlyPayment, loanDate забезпечують фінансову та часову характеристику кредиту;

– поле isPaid служить індикатором для визначення боргових зобов'язань клієнта.

Кожен запис у структурі є незалежним і відповідає одному клієнту. Поля працюють в унісон, забезпечуючи:

– точність розрахунків;

– зручність пошуку (за іменем, номером телефону, датою);

– можливість редагування та сортування записів.

При зчитуванні запису з файлу в оперативну пам'ять поля структури обробляються таким чином:

– ім'я клієнта з поля fullName відображається в заголовку таблиці;

– на основі loanBody розраховуються значення totalAmount та monthlyPayment;

– дата з поля loanDate використовується для перевірки актуальності кредиту;

– значення isPaid дозволяє ідентифікувати боржників у фінансовій системі.

	Вик.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		12

Завдяки такій структурі записів, програма забезпечує зручність у роботі з великим обсягом даних, а також спрощує інтеграцію нових полів за потреби.

2.4 Опис використаних бібліотек

У проєкті були використані наступні бібліотеки:

– `iostream` – стандартна бібліотека C++, необхідна для роботи з потоками введення та виведення. Вона дозволяє використовувати оператори `cin` та `cout` для читання з клавіатури та виведення даних у консоль;

Приклад використання бібліотеки для виведення та введення даних наведено в лістингу 2.1.

Лістинг 2.1 – Використання `iostream`

```
cout << "Enter the \"Sum of credit\" (from 1000 to 100,000) - ";  
getline(cin, input);
```

Тут `cin` зчитує введення користувача, а `cout` забезпечує виведення інструкцій у консоль.

– `fstream` – використовується для роботи з файлами. Завдяки цій бібліотеці програма може читати, записувати та модифікувати дані у файлі;

Приклад використання в коді проєкту наведено в лістингу 2.2.

Лістинг 2.2 – Використання `fstream`

```
ifstream file(path, ios::binary);  
if (!file) {  
    cout << "Error in file's opening" << endl;  
    return;  
}
```

У цьому фрагменті програма перевіряє, чи вдалося відкрити файл для читання.

– `iomanip` – бібліотека для маніпуляцій форматуванням виведення. У програмі вона використовується для створення відформатованих таблиць із даними клієнтів;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		13

Приклад використання в коді проекту наведено в лістингу 2.3.

Лістинг 2.3 – Використання iomanip

```
cout << "| " << setw(3) << left << ++i << " | "  
    << setw(17) << left << current.city << "| "  
    << setw(20) << left << current.fullName << " | +38"  
    << setw(11) << left << current.phoneNumber << " | "  
    << setw(14) << right << fixed << setprecision(2) <<  
current.loanBody << " | ";
```

У цьому прикладі `setw`, `left`, і `setprecision` формують таблицю із рівними відступами для зручності читання.

– `cstring` – бібліотека для роботи з C-рядками. У проекті вона використовується для маніпуляцій текстовими полями структури `Client`;

Приклад використання в коді проекту наведено в лістингу 2.4.

Лістинг 2.4 – Використання cstring

```
strncpy(client.phoneNumber, inputPhone.c_str(),  
sizeof(client.phoneNumber) - 1);  
client.phoneNumber[sizeof(client.phoneNumber) - 1] = '\0';
```

Тут функція `strncpy` копіює дані з введеного рядка в поле `phoneNumber`, забезпечуючи безпечне копіювання та правильне завершення рядка.

– `regex` – бібліотека для роботи з регулярними виразами. Використовується для перевірки формату даних, таких як ім'я або номер телефону;

Приклад використання в коді проекту наведено в лістингу 2.5.

Лістинг 2.5 – Використання regex

```
regex fullNamePattern("([A-Z]{1})([a-z]{1,20})([ ]{1})([A-Z]{1})([\\\\.]{1})([ ]{1})([A-Z]{1})([\\\\.]{1})");  
if (!regex_match(name, fullNamePattern)) {  
    cout << "Error: Full name must be entered in the format  
'Surname I. P.'\n";  
}
```

У цьому прикладі регулярний вираз перевіряє, чи ім'я відповідає формату "Прізвище І. О."

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		14

– ctime – бібліотека для роботи з датами і часом. Вона дозволяє перевіряти правильність дат, введених користувачем, а також виконувати операції з датами;

Приклад використання в коді проекту наведено в лістингу 2.6.

Лістинг 2.6 – Використання ctime

```
bool isDateBetween(const char targetDate[], const char
startDate[], const char endDate[]) {
    time_t target = convertToTimeT(targetDate);
    time_t start = convertToTimeT(startDate);
    time_t end = convertToTimeT(endDate);
    return (target >= start && target <= end);
}
```

Цей код перевіряє, чи належить конкретна дата певному діапазону.

– Windows.h – специфічна бібліотека для Windows, яка використовується для налаштування кодової сторінки в консолі.

Приклад використання в коді проекту наведено в лістингу 2.7.

Лістинг 2.7 – Використання Windows.h

```
SetConsoleOutputCP(CP_UTF8);
SetConsoleCP(CP_UTF8);
```

Це забезпечує правильне відображення кирилических символів у консолі.

2.5 Опис структури програми

Програма включає в себе: головну функцію main та 35 функцій-підпрограм.

Програма містить в собі наступні функції-підпрограми:

– int main() – призначена для організації головного меню програми та виклику інших функцій;

– bool FileExists(const string& path) – перевіряє наявність файлу за вказаним шляхом;

– void EnsureFileExists(const string& path) – створює файл, якщо він відсутній;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		15

- bool HasRecords(const string& path) – перевіряє, чи є записи у файлі;
- bool IsUniquePhoneNumber(const string& path, const char phoneNumber)*
– перевіряє, чи номер телефону унікальний у файлі;
- void WriteClient(const string& path, const Client& client) – записує дані клієнта у файл;
- void InputLoanBody(float& loanBody) – вводить суму кредиту з перевіркою на допустимий діапазон;
- bool IsValidDate(const char date[]) – перевіряє коректність дати у форматі "DD.MM.YYYY";
- bool isValidLength(const string& str, size_t maxLength) – перевіряє, чи рядок не перевищує максимальну довжину;
- string formatCityName() – зчитує назву міста, перевіряє її довжину, формат та коректність символів, повертає відформатований рядок;
- void SityInput(Client& client) – зчитує та записує відформатовану назву міста у поле “client.city”;
- void ValidateFullName(char fullName[30]) – зчитує та перевіряє повне ім’я за форматом "Surname I. P.", записує у “fullName”;
- void InputIsPaid(Client& client) – зчитує введення статусу штрафу ('+' або '-'), оновлює поле “client.isPaid”;
- void InputLoanDate(Client& client) – зчитує дату у форматі "DD.MM.YYYY", перевіряє валідність і записує у “client.loanDate”;
- void InputClient(Client& client, const string& path) – збирає та перевіряє дані клієнта перед їх записом;
- void ValidatePhoneNumber(Client& client) – вводить і перевіряє номер телефону клієнта;
- void ViewClients(const string& path) – виводить список клієнтів з даними;
- void EditClientRecord(Client& client) – дозволяє редагувати всі дані клієнта;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		16

- void EditClientField(Client& client) – дозволяє редагувати окремі поля запису клієнта;
- void EditClient(const string& path) – здійснює редагування записів у файлі;
- void DeleteRecordByName() – видаляє запис клієнта за повним ім'ям, перезаписуючи всі інші записи в новий файл;
- void DeleteRecordByPhone(const string& path) – видаляє запис клієнта за номером телефону аналогічно видаленню за ПІБ;
- void DeleteClient(const string& path) – пропонує вибір критерію для видалення (ПІБ або телефон) і викликає відповідну функцію;
- void SortRecordsByCity(const string& path) – сортує записи за містом клієнта в алфавітному порядку;
- void SortRecordsBySumOfCredit() – сортує записи у файлі за сумою кредиту (loan body) у порядку зростання;
- void SortRecords(const string& path) – меню для вибору сортування записів за містом або сумою кредиту;
- int OnlyOneRecord(const string& path) – рахує кількість записів у файлі за заданим шляхом;
- void FIRST_ByClientName(const string& path) – здійснює пошук кредитів за повним ім'ям клієнта;
- void Second_MAX_MIN(const string& path) – знаходить записи з мінімальною та максимальною сумою кредиту;
- time_t convertToTimeT(const char date[]) – перетворює дату у форматі "DD.MM.YYYY" на тип "time_t";
- bool compareDates(const char date1[], const char date2[]) – порівнює дві дати, повертає "true", якщо "date1" раніше за "date2";
- bool isDateBetween(const char targetDate[], const char startDate[], const char endDate[]) – перевіряє, чи знаходиться дата "targetDate" між "startDate" і "endDate";

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		17

- void Third_CreditsByDate(const string& path) – шукає кредити у заданому періоді;
- void Forth_Debetors(const string& path) – виводить список клієнтів, які ще не виплатили кредити;
- void Fifth_TotalAmount(const string& path) – обчислює загальну суму доходу банку за всі кредити;
- void Requests() – виводить меню запитів, приймає вибір користувача та викликає відповідні функції.

2.6 Детальний опис функцій

Програма складається з наступних функцій:

- int main() – дана функція призначена для організації головного меню програми та виклику інших функцій. Користувач, вибираючи пункт меню, може перейти до відповідної функції, щоб виконати потрібну дію (наприклад, додавання запису, сортування, видалення тощо);
- bool FileExists(const string& path) – функція перевіряє існування файлу за вказаним шляхом. Повертає true, якщо файл існує, і false у протилежному випадку. Передаємо у функцію шлях до файлу, результат повертається у вигляді логічного значення;
- void EnsureFileExists(const string& path) – функція створює файл, якщо він відсутній. Передаємо у функцію шлях до файлу, результат не повертається. Якщо файл вже існує, функція нічого не змінює;
- bool HasRecords(const string& path) – функція перевіряє, чи є у файлі хоча б один запис. Повертає true, якщо записи є, і false, якщо файл порожній;
- bool IsUniquePhoneNumber(const string& path, const char phoneNumber)* – функція перевіряє, чи номер телефону є унікальним у файлі. Передаємо шлях до файлу та номер телефону, функція повертає true, якщо номер унікальний, і false, якщо вже існує;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		18

– void WriteClient(const string& path, const Client& client) – функція записує дані клієнта у файл. Передаємо у функцію шлях до файлу та об'єкт клієнта. Функція не повертає значення, але записує дані у файл;

– void InputLoanBody(float& loanBody) – функція здійснює введення суми кредиту із перевіркою на допустимий діапазон (від 1000 до 100000). Передаємо змінну loanBody за посиланням, функція модифікує її значення;

– bool IsValidDate(const char date[]) – функція перевіряє коректність дати у форматі "DD.MM.YYYY". Повертає true, якщо дата валідна, і false у протилежному випадку. Передаємо у функцію дату у форматі "DD.MM.YYYY", результат повертається у вигляді логічного значення;

– bool isValidLength(const string& str, size_t maxLength) – функція перевіряє, чи довжина рядка не перевищує задану максимальну довжину. Повертає true, якщо рядок валідний, і false у протилежному випадку. Передаємо у функцію рядок і максимальну довжину, результат повертається у вигляді логічного значення;

– string formatCityName() – функція зчитує назву міста, перевіряє її довжину, формат та коректність символів. Повертає відформатований рядок. Введення здійснюється у форматі рядка, результат повертається у вигляді рядка;

– void SityInput(Client& client) – функція зчитує та записує відформатовану назву міста у поле client.city. Передаємо посилання на об'єкт Client, результат не повертається;

– void ValidateFullName(char fullName[30]) – функція зчитує та перевіряє повне ім'я за форматом "Surname I. P.". Повертає результат у вигляді масиву символів, передаємо вхідні дані через масив символів;

– void InputIsPaid(Client& client) – функція зчитує введення статусу штрафу ('+' або '-'), оновлює поле client.isPaid. Передаємо посилання на об'єкт Client, результат не повертається;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		19

– void InputLoanDate(Client& client) – функція зчитує дату у форматі "DD.MM.YYYY", перевіряє валідність і записує у client.loanDate. Передаємо посилання на об'єкт Client, результат не повертається;

– void InputClient(Client& client, const string& path) – функція збирає дані клієнта, виконує перевірки коректності введених даних та ініціалізує структуру Client. Передаємо змінну client за посиланням і шлях до файлу;

– void ValidatePhoneNumber(Client& client) – функція перевіряє введення номера телефону клієнта (10 цифр без пробілів та інших символів). Передаємо у функцію змінну client за посиланням, функція змінює поле phoneNumber;

– void ViewClients(const string& path) – функція виводить на екран список усіх клієнтів з їх даними. Передаємо шлях до файлу, функція не повертає значення;

– void EditClientRecord(Client& client) – функція дозволяє редагувати всі поля структури Client. Передаємо змінну client за посиланням, функція змінює всі поля клієнта;

– void EditClientField(Client& client) – функція редагує окреме поле клієнта за вибором користувача. Передаємо змінну client за посиланням, функція модифікує вибране поле;

– void EditClient(const string& path) – функція дозволяє редагувати записи у файлі. Передаємо у функцію шлях до файлу, результат не повертається;

– void DeleteRecordByName() – функція видаляє запис клієнта за повним ім'ям, перезаписуючи всі інші записи в новий файл. Передаємо у функцію нічого, але модифікується файл із записами;

– void DeleteRecordByPhone(const string& path) – функція видаляє запис клієнта за номером телефону аналогічно видаленню за ПІБ. Передаємо у функцію шлях до файлу;

– void DeleteClient(const string& path) – функція пропонує користувачеві вибрати критерій для видалення запису (ПІБ або номер телефону) та викликає відповідну функцію. Передаємо шлях до файлу;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				20
Змн.	Арк.	№ докум.	Підпис	Дат		

– void SortRecordsByCity(const string& path) – функція сортує записи у файлі за містом клієнта в алфавітному порядку. Передаємо шлях до файлу, результат записується у файл;

– void SortRecordsBySumOfCredit() – функція сортує записи у файлі за сумою кредиту (loan body) у порядку зростання. Файл відкривається за шляхом PATH, результат не повертається;

– void SortRecords(const string& path) – функція є меню для вибору користувачем одного з двох варіантів сортування (за містом або за сумою кредиту). Передаємо шлях до файлу;

– int OnlyOneRecord(const string& path) – функція рахує кількість записів у файлі за заданим шляхом. Передаємо у функцію шлях до файлу, результат повертається у вигляді цілого числа;

– void FIRST_ByClientName(const string& path) – функція здійснює пошук кредитів за повним ім'ям клієнта. Передаємо шлях до файлу, результат не повертається;

– void Second_MAX_MIN(const string& path) – функція знаходить записи з мінімальною та максимальною сумою кредиту. Передаємо шлях до файлу, результат виводиться на екран;

– time_t convertToTimeT(const char date[]) – функція перетворює дату у форматі рядка ("dd.mm.yyyy") у тип "time_t" для зручності виконання порівнянь і операцій з датами;

– bool compareDates(const char date1[], const char date2[]) – функція порівнює дві дати у форматі рядків і визначає, чи є перша дата раніше другої. Використовує допоміжну функцію "convertToTimeT" для перетворення дат у формат "time_t". Повертає "true", якщо перша дата менша за другу, інакше повертає "false";

– bool isDateBetween(const char targetDate[], const char startDate[], const char endDate[]) – функція перевіряє, чи входить цільова дата у заданий діапазон дат. Дати передаються у вигляді рядків. Використовує функцію "convertToTimeT"

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		21

для перетворення дат у формат “time_t”. Повертає “true”, якщо цільова дата знаходиться між початковою та кінцевою, включно, інакше повертає “false”;

– void Third_CreditsByDate(const string& path) – функція шукає кредити у заданому періоді. Передаємо шлях до файлу, результат виводиться на екран;

– void Forth_Debetors(const string& path) – функція виводить список клієнтів, які ще не виплатили кредити. Передаємо шлях до файлу;

– void Fifth_TotalAmount(const string& path) – функція обчислює загальну суму доходу банку за всі кредити. Передаємо шлях до файлу;

– void Requests() – функція виводить меню запитів, приймає вибір користувача та викликає відповідні функції. Вхідні дані передаються через вибір користувача у меню, результат не повертається.

2.7 Опис функцій перевірки контролю вхідних даних

Проект використовує наступні функції, які контролюють перевірку на коректне введення даних користувачем.

void WriteClient(const string& path, const Client& client) – це функція, яка слугує для запису об'єкта типу Client у файл у двійковому форматі. Вона продемонстрована у лістингу Б.2. Функція перевіряє, чи файл було успішно відкрито для запису, і у випадку помилки повідомляє про це через стандартний потік помилок (cerr). Якщо файл відкрито, функція записує дані у файл, після чого закриває його.

В даній функції контроль вхідних даних здійснюється за рахунок перевірки успішного відкриття файлу.

На рисунку 2.1 наведений приклад помилки запису даних до файлу.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				22
Змн.	Арк.	№ докум.	Підпис	Дат		

```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Tttt
Enter Full Name (format: 'Surname I. P.'): Ff F. F.
Enter the "Phone number" (10 digits): +38089521463
Error: Phone number must consist of exactly 10 digits.
Enter the "Phone number" (10 digits): +380895214635
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 15685
Please enter the date in DD.MM.YYYY format: 04.05.2003
Enter, whether the fine "Is paid?" (+/-): +
Error: Could not open file for writing!
Press any key to continue . . .

```

Рисунок 2.1 – Помилка запису даних до файлу.

`void InputLoanBody(float& loanBody)` – це функція, яка слугує для введення суми кредиту від користувача, вона продемонстрована у лістингу Б.3. Функція виконує дві основні перевірки введених даних:

Перевірка формату введених даних – за допомогою регулярного виразу визначається, чи є введений текст коректним числовим значенням.

Перевірка діапазону – перевіряється, чи входить число у допустимі межі (від 1,000 до 1,000,000).

Якщо введені дані не відповідають вимогам, функція виводить повідомлення про помилку та запрошує повторне введення, доки не буде отримано коректне значення.

На рисунку 2.2 наведений приклад помилок введення суми кредиту.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		23


```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Ffff
Enter Full Name (format: 'Surname I. P.'): Fff D. F.
Enter the "Phone number" (10 digits): +380456123485
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 1000k
Error: You must enter a valid number!
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 10
Error: Amount is not in the acceptable range!
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 1000
Please enter the date in DD.MM.YYYY format: |

```

Рисунок 2.2 – Помилки введення суми кредиту

Проект використовує функцію для контролю введення та форматування назви міста.

string formatCityName() – це функція, яка отримує від користувача назву міста, перевіряє її на відповідність вимогам та форматує у правильному вигляді, вона продемонстрована у лістингу Б.4.

Функція виконує такі перевірки та дії:

- перевірка довжини: Назва міста не може перевищувати встановлену максимальну довжину (16 символів);
- перевірка символів: Назва міста може містити тільки букви, пробіли або дефіси. Якщо введено цифри чи спеціальні символи, користувачеві пропонується повторити введення;
- форматування: Після перевірки кожне слово у назві міста починається з великої літери, а решта символів стають малими.

Якщо введені дані не відповідають вимогам, функція виводить відповідне повідомлення про помилку та запрошує повторне введення.

На рисунку 2.3 наведений приклад помилок введення назви міста.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		24

```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Llanfairpwllgwyngyll
Error: The city name cannot exceed 16 characters. Try again: Dnipro4
Error: The city name cannot have numbers or special symbols. Try again: dnipro0

```

Рисунок 2.3 – Помилки введення назви міста.

Функція `ValidateFullName(char fullName[30])` забезпечує введення повного імені користувача у визначеному форматі, продемонстрована у лістингу Б.5. Вона гарантує, що введена інформація відповідає формату "Прізвище І. П.", де:

Прізвище починається з великої літери та може мати до 20 символів.

Ім'я та по-батькові представлені ініціалами, розділеними пробілами і з крапкою після кожної літери.

Функція виконує дві основні перевірки:

- Видалення зайвих пробілів на початку, в кінці та між словами;
- Перевірка формату введення за допомогою регулярного виразу.

Функція працює у циклі, доки користувач не введе дані у правильному форматі. У разі некоректного введення виводиться відповідне повідомлення про помилку.

На рисунку 2.4 наведений приклад помилки введення ПІБ.

```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Dnipro
Enter Full Name (format: 'Surname I. P.'): Marek D.D.
Error: Full name must be entered in the format 'Surname I. P.' (Surname max: 20)
Enter Full Name (format: 'Surname I. P.'): Marek d. D.
Error: Full name must be entered in the format 'Surname I. P.' (Surname max: 20)
Enter Full Name (format: 'Surname I. P.'): Marek D. D.

```

Рисунок 2.4 – Помилка введення ПІБ

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		25

`void InputIsPaid(Client& client)` – це функція, яка забезпечує введення від користувача інформації про сплату штрафу, вона продемонстрована у лістингу Б.6. Введення має бути лише одним із символів:

- «+» якщо штраф сплачено;
- «-» якщо штраф не сплачено.

Функція виконує одну перевірку:

Перевірка введеного символу: використовується регулярний вираз для перевірки, чи є введений символ допустимим (+ або -).

Якщо введення некоректне, користувач отримує відповідне повідомлення про помилку та запрошується повторно ввести значення.

На рисунку 2.5 наведений приклад помилки введення даних про сплату штрафу.

```

Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Dnipro
Enter Full Name (format: 'Surname I. P.'): Biedin T. V.
Enter the "Phone number" (10 digits): +380852369541
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 10000
Please enter the date in DD.MM.YYYY format: 04.05.2008
Enter, whether the fine "Is paid?" (+/-): 77dddd
Error: Input must be '+' or '-' only. Try again.
Enter, whether the fine "Is paid?" (+/-): 1
Error: Input must be '+' or '-' only. Try again.
Enter, whether the fine "Is paid?" (+/-): +-
Error: Input must be '+' or '-' only. Try again.
Enter, whether the fine "Is paid?" (+/-): +
Press any key to continue . . . |
  
```

Рисунок 2.5 – Помилка введення даних про сплату штрафу

`void InputLoanDate(Client& client)` – це функція, яка забезпечує введення дати кредиту у форматі DD.MM.YYYY, вона продемонстрована у лістингу Б.7. Функція виконує перевірку введених даних на правильність формату та довжини.

Функція виконує такі перевірки:

- перевірка довжини введеного рядка: Дата повинна містити не більше 10 символів. Якщо довжина перевищує 10 символів, користувач отримує відповідне повідомлення про помилку;

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		26

- перевірка формату дати: Для перевірки коректності формату та існування дати використовується окрема функція IsValidDate;
- після успішного введення дані копіюються у масив loanDate структури Client.

На рисунку 2.6 наведений приклад помилки введення дати кредиту.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Dnipro
Enter Full Name (format: 'Surname I. P.'): Biedin T. V.
Enter the "Phone number" (10 digits): +380444444444
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 20000
Please enter the date in DD.MM.YYYY format: 20.04.24
Please enter the date in DD.MM.YYYY format: fgd
Please enter the date in DD.MM.YYYY format: 02.04.2007
Enter, whether the fine "Is paid?" (+/-): |
```

Рисунок 2.6 – Помилка введення дати кредиту

void ValidatePhoneNumber(Client& client) – це функція, яка забезпечує введення та перевірку номера телефону у форматі +38XXXXXXXXXXXX, де XXXXXXXXXXXX – рівно 10 цифр. Вона продемонстрована у лістингу Б.8.

Функція виконує такі перевірки:

Перевірка формату введення: Використовується регулярний вираз для перевірки, що введений номер телефону складається рівно з 10 цифр.

Запис даних у структуру: Після успішної перевірки номер телефону зберігається у полі phoneNumber структури Client.

Якщо введений номер не відповідає вимогам, функція виводить відповідне повідомлення про помилку та запрошує повторно ввести значення.

На рисунку 2.7 наведений приклад помилки введення номера телефону.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		27

```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Dnipro
Enter Full Name (format: 'Surname I. P.'): Biedin T. V.
Enter the "Phone number" (10 digits): +38041202143f
Error: Phone number must consist of exactly 10 digits.
Enter the "Phone number" (10 digits): +38041202143
Error: Phone number must consist of exactly 10 digits.
Enter the "Phone number" (10 digits): +380412021434
Enter the "Sum of credit" (from 1,000 to 1,000,000) - |

```

Рисунок 2.7 – Помилка введення номера телефону

`void ViewClients(const string& path)` – це функція, яка виводить на екран список клієнтів, збережених у файлі у форматі таблиці. Вона продемонстрована у лістингу Б.9. Функція забезпечує читання даних із файлу та форматоване відображення інформації про клієнтів.

Функція виконує такі дії:

Перевірка наявності записів: За допомогою функції `HasRecords` перевіряється, чи файл містить записи. Якщо файл порожній, функція виводить відповідне повідомлення.

Перевірка доступності файлу: Якщо файл не вдалося відкрити, виводиться повідомлення про помилку.

На рисунку 2.8 наведений приклад порожнього файлу.

```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 1
File is empty.
Press any key to continue . . . |

```

Рисунок 2.8 – Повідомлення про порожній файл

`void EditClientField(Client& client)` – це функція, яка забезпечує редагування вибраного поля для конкретного клієнта. Вона продемонстрована у

лістингу Б.10. Функція пропонує користувачеві вибрати одне з шести полів для редагування, перевіряючи коректність вибору.

Функція виконує такі дії:

1) Вибір поля:

- місто;
- повне ім'я;
- номер телефону;
- сума кредиту;
- дата кредиту;
- статус першого платежу.

2) Перевірка вибору: Використовується регулярний вираз для перевірки, що вибір користувача є числом від 1 до 6.

3) Редагування вибраного поля: Виклик відповідної функції для обробки введених даних залежно від вибраного поля.

На рисунку 2.9 наведений приклад помилки при виборі неправильного номера поля.

```
Are you sure? (Y/any other input): y
Enter the record number to edit: 8
Error: Invalid record number.
Enter the record number to edit: 1

Edit:
1. Full record
2. Particular field
Your choice: 2
Select a field to edit:
1. City
2. Full Name
3. Phone Number
4. Sum of credit
5. Loan Date
6. First pay
Select an option: 8
Invalid choice. Try again (1-6): a
Invalid choice. Try again (1-6): 1

Enter the "City" (max 16 characters): New-York
Press any key to continue . . . |
```

Рисунок 2.9 – Помилка вибору номера поля

`void EditClient(const string& path)` – це функція, яка дозволяє редагувати інформацію про клієнтів, збережену у файлі. Вона забезпечує вибір запису для

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				29
Змн.	Арк.	№ докум.	Підпис	Дат		

редагування, перевіряє введені дані та надає можливість редагувати повний запис або окремі поля. Вона продемонстрована у лістингу Б.11.

Функція виконує такі дії:

- 1) Перевірка наявності записів: Якщо файл порожній, функція повідомляє про це та завершує виконання.
 - 2) Візуалізація списку клієнтів: Виводить список клієнтів для вибору запису.
 - 3) Підтвердження редагування: Користувач має підтвердити, чи хоче редагувати запис.
 - 4) Вибір запису: Перевіряє коректність введеного номера запису, використовуючи перевірку діапазону та обробку виключень (`invalid_argument` та `out_of_range`).
 - 5) Редагування запису: Користувач може вибрати один із двох варіантів:
 - 6) Редагувати повний запис (`EditClientRecord`).
 - 7) Редагувати окреме поле (`EditClientField`).
 - 8) Збереження змін: Модифікований запис перезаписується у файл.
- На рисунку 2.10 наведений приклад помилки введення номера запису.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 3
x-----
| № | City | FullName | Phone Number |
x-----
| 1 | New-York | Biedin T. V. | +380589647132 |
x-----
Are you sure? (Y/any other input): y
Enter the record number to edit: 8
Error: Invalid record number.
Enter the record number to edit: 1

Edit:
1. Full record
2. Particular field
Your choice: |
```

Рисунок 2.10 – Помилка введення номера запису

`void DeleteRecordByName()` – це функція, яка видаляє запис про клієнта за вказаним іменем у форматі Surname I. P. Вона продемонстрована у лістингу Б.12. Основна увага приділяється перевірці правильності введення.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				30
Змн.	Арк.	№ докум.	Підпис	Дат		

Функція виконує такі перевірки:

1) Перевірка формату введення. Використовується регулярний вираз, який гарантує, що ім'я відповідає формату:

- Прізвище починається з великої літери та містить до 20 символів;
- Ім'я та по-батькові записані у вигляді ініціалів з крапками;
- Приклад коректного введення: Smith J. D;
- Видалення зайвих пробілів - видаляються пробіли на початку, в кінці та між словами за допомогою `regex_replace`.

2) Перевірка існування файлу та створення тимчасового файлу. Якщо файл не відкривається або тимчасовий файл не створюється, виводиться відповідне повідомлення.

На рисунку 2.11 наведений приклад помилок введення імені клієнта.

```
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 4
X-----X
| № | City | FullName | Phone Number | Sum of credit |
X-----X
| 1 | New-York | Biedin T. V. | +380589647132 | 15475.00 |
X-----X
Are you sure? (Y/ any other input): y
Choose deletion method:
1. Delete by Full Name
2. Delete by Phone Number
Enter your choice: 1
Enter the full name to delete (format: 'Surname I. P.'): biedin T. V.
Error: Full name must be entered in the format 'Surname I. P.' (Surname max: 20)
Enter the full name to delete (format: 'Surname I. P.'): Biedin T.V.
Error: Full name must be entered in the format 'Surname I. P.' (Surname max: 20)
Enter the full name to delete (format: 'Surname I. P.'): Biedin T. V.
Client with name "Biedin T. V." deleted successfully.
Press any key to continue . . . |
```

Рисунок 2.11 – Помилка введення імен клієнта

2.8 Схема зв'язків між функціями

Схема зв'язку між функціями наведена на рисунках 2.12 – 2.13.

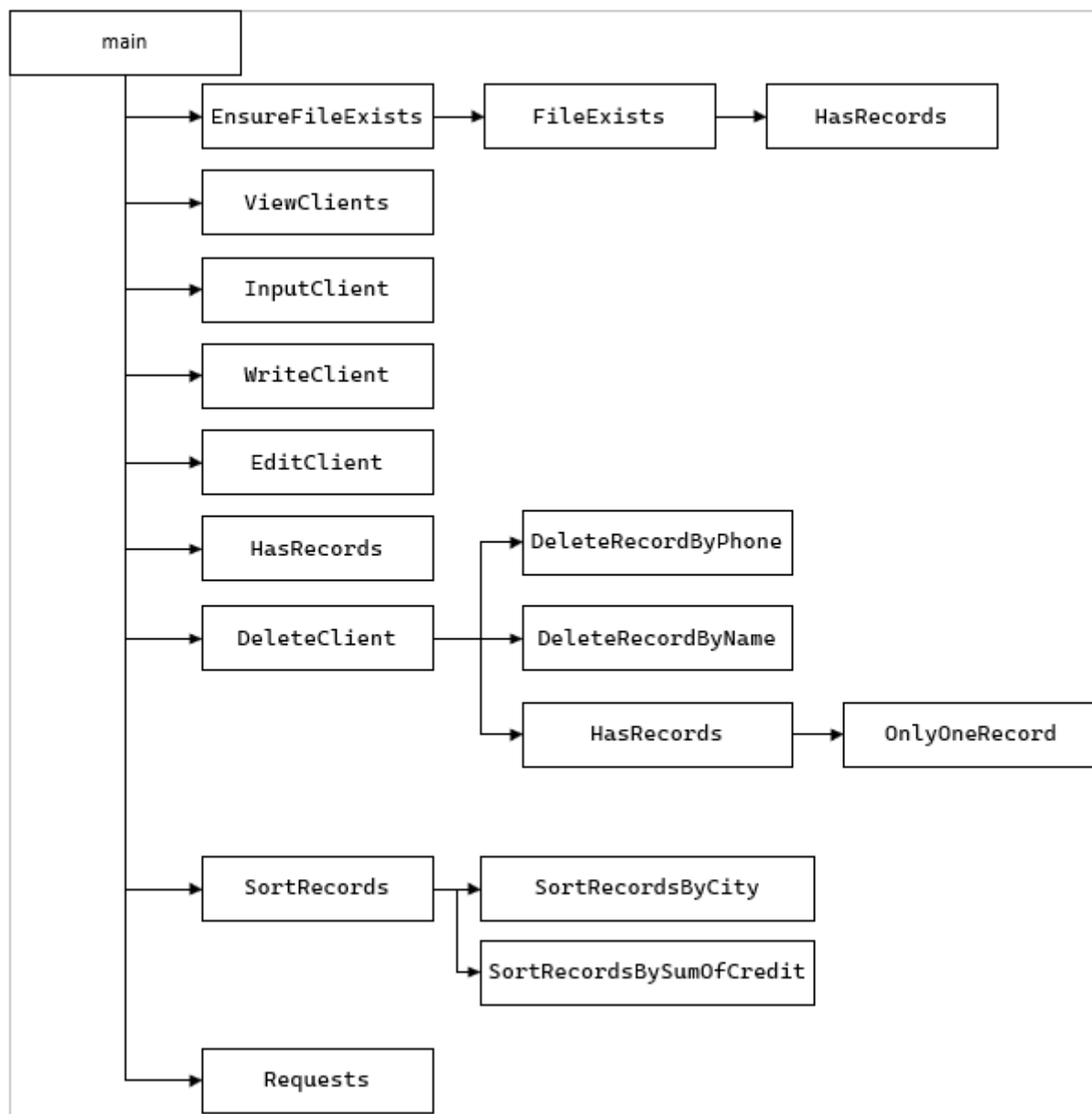


Рисунок 2.12 – Схема зв'язків між функціями

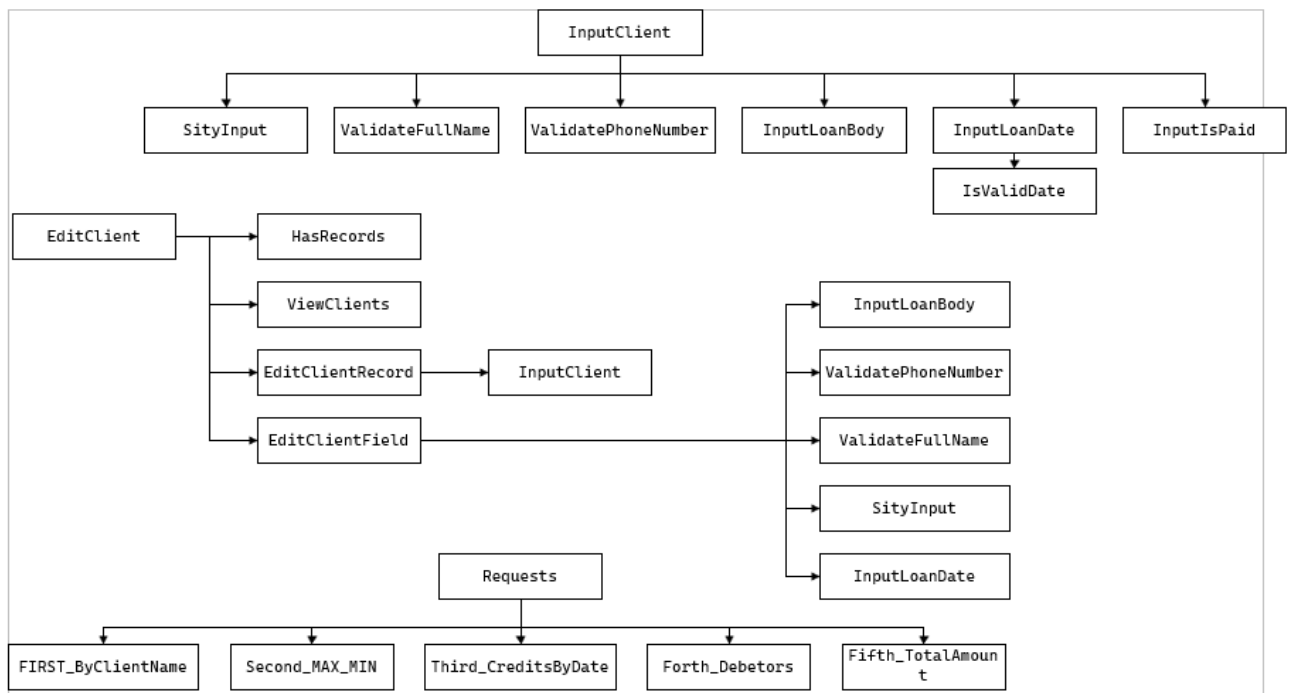


Рисунок 2.13 – Схема зв'язків між функціями (продовження)

Рисунок 3.3 – Відсутність виводу порожності файлу

4) Помилка представлена на рисунку 3.4. Функція `strcpy` вважається небезпечною в сучасних компіляторах, оскільки не перевіряє розмір буфера, куди копіюються дані. Для виправлення помилки було використано функцію `strcpy_s`, яка більш безпечна та перевіряє розмір буфера.

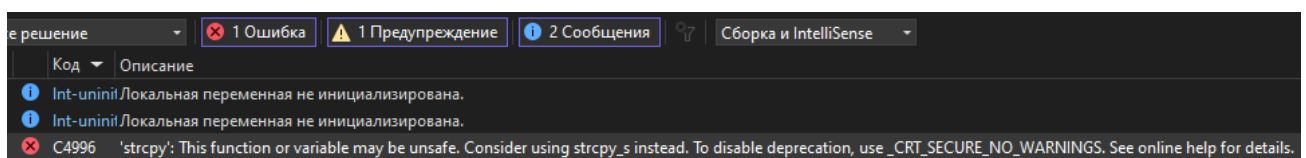


Рисунок 3.4 – Використана ненадійна функція

5) Помилка представлена на рисунку 3.5. Значення кредитної суми (`loanBody`) можна ввести поза межами діапазону. Для виправлення помилки було додано перевірку на діапазон від 10,000 до 1,000,000.

Рисунок 3.5 – Тіло кредиту = 1

6) Помилка представлена на рисунку 3.6. Перша літера назв міст зникає бо не використовується `cin.ignore()`. Для виправлення помилки було спеціально додано `cin.ignore()`.

```

F:\3 курс\КП\Project\64\Debug\Project.exe
Облік кредитів банком
1. Вихід з програми
2. Перегляд файлу
3. Додати дані у файл
Введіть номер пункту: 2
x-----x
| № | Місто | ПІБ клієнта | Номер телефону | Тіло кредиту | Сума до сплати | Щомісячний платіж | Перший платіж |
x-----x
| 1 | ніпро | Котенко В.С. | +380669989971 | 1e+06 | 1.1e+06 | 91666.7 | + |
| 2 | иїв | Пупкин В.А. | +380671589972 | 100000 | 110000 | 9166.67 | + |
| 3 | ернігів | Васечкин В.В. | +380679989973 | 123000 | 135300 | 11275 | - |
x-----x
Press any key to continue . . .

```

Рисунок 3.6 – Відсутність першої літери у назвах міст

7) Помилка представлена на рисунку 3.7. Потреба повторного натискання «Enter» для вводу міста. Для виправлення помилки було спеціально змінено місце для `cin.ignore()`.

```

D:\3 курс\дз\КП\код\1\Proje
Меню:
0. Вийти
1. Переглянути записи
2. Додати запис
3. Редагувати запис
Оберіть пункт: 3
x-----x
| № | Місто | ПІБ клієнта | Номер телефону | Сума кредиту |
x-----x
| 1 | Dnipro | NoLay H. H. | +380669989971 | 1000.00 |
| 2 | Kyiv | Fallen B. E. | +380671589972 | 100.00 |
x-----x
Введіть номер запису для редагування: 1

Редагувати:
1. Запис повністю
2. окреме поле
Оберіть: 1
Редагування запису клієнта.
Введіть нові дані для клієнта:

Введіть місто: |

```

Рисунок 3.7 – Подвійний «Enter»

8) Помилка представлена на рисунку 3.8. Проблема із кодуванням терміналу за замовчуванням та кодуванням сторінки – не пропорційна таблиця. Вхідні дані були переведені на англійську мову.

```

F:\3 курс\КП\Project\64\Debug\Project.exe
Облік кредитів банком
1. Вихід з програми
2. Перегляд файлу
3. Додати дані у файл
Введіть номер пункту: 2
x-----x
| № | Місто | ПІБ клієнта | Номер телефону | Тіло кредиту | Сума до сплати | Щомісячний платіж | Перший платіж |
x-----x
| 1 | Lass Angels | Бурулько | +380321654987 | 1 | 1.1 | 0.0916667 | + |
x-----x
Press any key to continue . . .

```

Рисунок 3.8 – Не пропорційна таблиця

9) Помилка представлена на рисунку 3.9. Проблема із виводом помилки при введенні неправильного числа. Було змінено перевірку вводу.

```

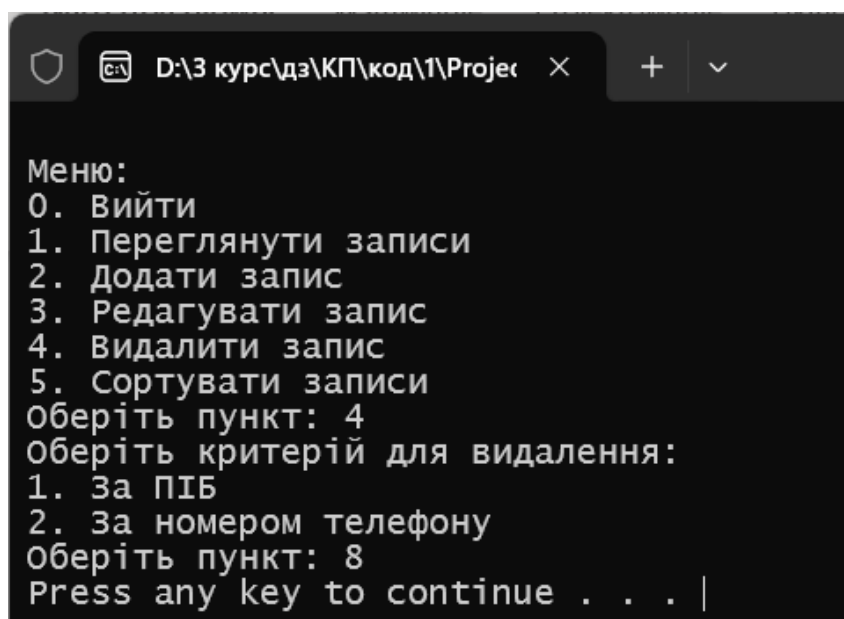
Оберіть пункт: 3
X-----X
| № | Місто | ПІБ клієнта | Номер телефону | Сума кредиту |
X-----X
| 1 | Dnipro | No1ay H. H. | +380669989971 | 1000.00 |
| 2 | Kyiv | Fallen B. E. | +380671589972 | 100.00 |
X-----X
Введіть номер запису для редагування: 2

Редагувати:
1. Запис повністю
2. Окреме поле
Оберіть: 2
Оберіть поле для редагування:
1. Місто
2. ПІБ
3. Номер телефону
4. Сума кредиту
5. Дата видачі кредиту
6. Статус платежу
Оберіть пункт: 15323
Невірний вибір.
Запис відредаговано успішно.
Press any key to continue . . . |

```

Рисунок 3.9 – Відсутність виводу помилки

10) Помилка представлена на рисунку 3.10. Відсутній вивід невірно набраного числа. Був доданий default.



```

D:\3 курс\дз\КП\код\1\Projес
Меню:
0. Вийти
1. Переглянути записи
2. Додати запис
3. Редагувати запис
4. Видалити запис
5. Сортувати записи
Оберіть пункт: 4
Оберіть критерій для видалення:
1. За ПІБ
2. За номером телефону
Оберіть пункт: 8
Press any key to continue . . . |

```

Рисунок 3.10 – Відсутність виводу помилки

11) Помилка представлена на рисунку 3.11. Відсутній вивід невірно набраного номера. Була додана перевірка.

```

D:\3 курс\дз\КП\код\1\Project
+
v
Меню:
0. Вийти
1. Переглянути записи
2. Додати запис
3. Редагувати запис
4. Видалити запис
5. Сортувати записи
Оберіть пункт: 4
Оберіть критерій для видалення:
1. За ПІБ
2. За номером телефону
Оберіть пункт: 2
Введіть номер телефону для видалення: 48
Запис успішно видалено.
Press any key to continue . . . |

```

Рисунок 3.11 – Неіснуючий в таблиці номер

12) Помилка представлена на рисунку 3.12. Відсутній вивід невірно набраного числа. Був доданий default.

```

D:\3 курс\дз\КП\код\1\Project
+
v
Меню:
0. Вийти
1. Переглянути записи
2. Додати запис
3. Редагувати запис
4. Видалити запис
5. Сортувати записи
Оберіть пункт: 5
Оберіть параметр для сортування:
1. Місто (city)
2. Сума кредиту (loanBody)
Ваш вибір: 8
Press any key to continue . . . |

```

Рисунок 3.12 – Відсутність виводу помилки

13) Помилка представлена на рисунку 3.13. Відсутнє повернення до головного меню. Була перероблена функція, додана перевірка вводу ПІБ, при відсутності клієнта відповідний вивід та повернення до головного меню.

```

D:\3 курс\дз\КП\код\1\Proje
Меню:
0. Вийти
1. Переглянути записи
2. Додати запис
3. Редагувати запис
4. Видалити запис
5. Сортувати записи
6. Виконати запити
Оберіть пункт: 6
Введіть ПІБ клієнта: Загальна кількість кредитів для : 0
Кредити відсутні.
Введіть початкову дату (DD.MM.YYYY): |

```

Рисунок 3.13 – Відсутність повернення до головного меню

Всі помилки були успішно усунені.

	Вик.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				39
Змн.	Арк.	№ докум.	Підпис	Дат		

4 ІНСТРУКЦІЯ З ПІДГОТОВКИ ДАНИХ ТА КОРИСТУВАННЯ ПРОГРАМОЮ

Для того, щоб розпочати роботу із програмою необхідно запустити її ехе-файл BankSystem.exe. Після цього на екрані з'явиться головне користувацьке меню, яке зображене на рисунку 4.1.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: |
```

Рисунок 4.1 – Головне користувацьке меню

Для перегляду всіх записів виберіть пункт 1 у головному меню (див. рис. 4.1). На екрані з'явиться таблиця з інформацією про всіх клієнтів, яка містить наступні поля, як зображено на рисунку 4.2.

Menu: 0. Exit 1. View Records 2. Add Record 3. Edit Record 4. Delete Record 5. Sort Records 6. Make a request Choose an option: 1										
№	City	FullName	Phone Number	Sum of credit	Total payable	Monthly Payment	First pay	LoanDate		
1	Chernigiv	Dorean D. D.	+380671589104	1000.00	1100.00	91.67	-	15.09.2023		
2	Dnipro	Nolay H. H.	+380669989971	1000.00	1100.00	91.67	+	15.09.2023		
3	Chernigiv	Mildon P. N.	+380661234566	15000.00	16500.00	1375.00	+	16.09.2023		
4	Odessa	Telon M. S.	+380661234103	24500.00	26950.00	2245.83	-	18.09.2023		
5	Odessa	Korner B. T.	+380679989979	34000.00	37400.00	3116.67	+	15.09.2023		
6	Lviv	Nolay H. H.	+380999876547	40000.00	44000.00	3666.67	+	17.09.2023		
7	Odessa	Korner B. T.	+380669989109	50000.00	55000.00	4583.33	+	16.09.2023		
8	Lviv	Dorean D. D.	+380999876106	75000.00	82500.00	6875.00	-	17.09.2023		
9	Dnipro	Mildon P. N.	+380669989111	99999.00	109998.90	9166.58	+	18.09.2023		
10	Kyiv	Fallen B. E.	+380671589972	100000.00	110000.00	9166.67	+	16.09.2023		
11	Dnipro	Telon M. S.	+380679989107	105000.00	115500.00	9625.00	+	18.09.2023		
12	Chernigiv	Nolay H. H.	+380661234102	110000.00	121000.00	10083.33	+	17.09.2023		
13	Kyiv	Nolay H. H.	+380935589110	110500.00	121550.00	10129.17	-	17.09.2023		
14	Odessa	Dorean D. D.	+380669989974	120500.00	132550.00	11045.83	+	18.09.2023		
15	Chernigiv	Mildon P. N.	+380679989973	200000.00	220000.00	18333.33	-	17.09.2023		
16	Dnipro	Fallen B. E.	+380661234568	200000.00	220000.00	18333.33	+	18.09.2023		
17	Kyiv	Telon M. S.	+380671589101	550000.00	605000.00	50416.67	-	16.09.2023		
18	Lviv	Korner B. T.	+380631234565	560000.00	616000.00	51333.33	-	15.09.2023		
19	Lviv	Fallen B. E.	+380671589108	800000.00	880000.00	73333.34	-	15.09.2023		
20	Kyiv	Mildon P. N.	+380669989105	990000.00	1089000.00	90750.00	+	16.09.2023		

Рисунок 4.2 – Вміст файлу data

Для виходу у головне меню програми необхідно натиснути будь-яку клавішу.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т. С.				
Змн.	Арк.	№ докум.	Підпис	Дат		40

Для додавання нового запису виберіть пункт 2 у головному меню програми, (ввівши цифру 2 та натиснувши клавішу «Enter»). Далі на екрані з'явиться запит підтвердження. Для підтвердження операції натисніть „у”, а для скасування натисніть бідь-яку іншу клавішу. На екрані послідовно з'являться запити для введення інформації про клієнта. Введіть дані у відповідному форматі, як наведено на рисунку 4.3.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 2
Are you sure? (Y/ any other input): y

Enter the "City" (max 16 characters): Dnipro
Enter Full Name (format: 'Surname I. P.'): Mildon P. N.
Enter the "Phone number" (10 digits): +380669989111
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 99999
Please enter the date in DD.MM.YYYY format: 18.09.2023
Enter, whether the fine "Is paid?" (+/-): +
Press any key to continue . . . |
```

Рисунок 4.3 – Робота функції додавання нового запису в файл Data

Для редагування запису виберіть пункт 3 у головному меню (див. рис. 4.1). Програма виведе список клієнтів із номерами записів та запит підтвердження. Введіть чи впевнені ви, що хочете редагувати рядок та введіть номер запису, який потрібно відредагувати.

Після цього в вас з'явиться можливість вибрати один із двох варіантів редагування:

- 1) Повний запис – можливість ввести усі поля заново.
- 2) Окреме поле – можливість вибору, яке поле запису потрібно змінити.

Залежно від вибору викликається та чи інша функція редагування. Ця можливість представлена на рисунку 4.4.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		41

```

6. Make a Request
Choose an option: 3
X-----X
| No | City | FullName | Ph |
X-----X
1 | Dnipro | Nolay H. H. | +380
2 | Kyiv | Fallen B. E. | +380
3 | Chernigiv | Mildon P. N. | +380
4 | Odessa | Dorean D. D. | +380
5 | Lviv | Korner B. T. | +380
6 | Chernigiv | Mildon P. N. | +380
7 | Lviv | Nolay H. H. | +380
8 | Dnipro | Fallen B. E. | +380
9 | Odessa | Korner B. T. | +380
10 | Kyiv | Telon M. S. | +380
11 | Chernigiv | Nolay H. H. | +380
12 | Odessa | Telon M. S. | +380
13 | Chernigiv | Dorean D. D. | +380
14 | Kyiv | Mildon P. N. | +380
15 | Lviv | Dorean D. D. | +380
16 | Dnipro | Telon M. S. | +380
17 | Lviv | Fallen B. E. | +380
18 | Odessa | Korner B. T. | +380
19 | Kyiv | Nolay H. H. | +380
20 | Odessa | Snode K. R. | +380
X-----X
Are you sure? (Y/any other input): y
Enter the record number to edit: 20

Edit:
1. Full record
2. Particular field
Your choice: |

```

Рисунок 4.4 – Вибір пункту редагування запису

Щоб відредагувати запис цілком, виберіть пункт меню №1(Full record), (ввівши цифру 1 у під-меню програми та натиснувши клавішу «Enter»). Далі вводите значення кожного поля вибраного запису. Ця можливість представлена на рисунку 4.5.

```

X
Are you sure? (Y/any other input): y
Enter the record number to edit: 20

Edit:
1. Full record
2. Particular field
Your choice: 1
Edit a customer record.
Enter new data for the client:

Enter the "City" (max 16 characters): Dnipro
Enter Full Name (format: 'Surname I. P.'): Mildon P. N.
Enter the "Phone number" (10 digits): +380669989111
Enter the "Sum of credit" (from 1,000 to 1,000,000) - 99999
Please enter the date in DD.MM.YYYY format: 18.09.2023
Enter, whether the fine "Is paid?" (+/-): +
Press any key to continue . . .

```

Рисунок 4.5 – Редагування запису цілком

Щоб відредагувати запис за окреми полями, виберіть пункт меню №2(Particular field), (ввівши цифру 2 у під-меню програми та натиснувши клавішу „Enter”). Функція виведе поля запису з котрих треба вибрати одне для редагування. Виберіть поле для редагування ввівши цифру від 1 до 6 та натиснувши «Enter». Введіть нове значення поля та натисніть «Enter». Ця можливість представлена на рисунку 4.6).

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 3
x-----
| № | City | FullName | Phone Number | Sum of credit |
x-----
1 | Dnipro | Nolay H. H. | +380669989971 | 1000.00 |
2 | Kyiv | Fallen B. E. | +380671589972 | 100000.00 |
3 | Chernigiv | Mildon P. N. | +380679989973 | 200000.00 |
4 | Odessa | Dorean D. D. | +380669989974 | 120500.00 |
5 | Lviv | Korner B. T. | +380631234565 | 560000.00 |
6 | Chernigiv | Mildon P. N. | +380661234566 | 15000.00 |
7 | Lviv | Nolay H. H. | +380999876547 | 40000.00 |
8 | Dnipro | Fallen B. E. | +380661234568 | 200000.00 |
9 | Odessa | Korner B. T. | +380679989979 | 34000.00 |
10 | Kyiv | Telon M. S. | +380671589101 | 550000.00 |
11 | Chernigiv | Nolay H. H. | +380661234102 | 110000.00 |
12 | Odessa | Telon M. S. | +380661234103 | 24500.00 |
13 | Chernigiv | Dorean D. D. | +380671589104 | 1000.00 |
14 | Kyiv | Mildon P. N. | +380669989105 | 990000.00 |
15 | Lviv | Dorean D. D. | +380999876106 | 75000.00 |
16 | Dnipro | Telon M. S. | +380679989107 | 105000.00 |
17 | Lviv | Fallen B. E. | +380671589108 | 800000.00 |
18 | Odessa | Korner B. T. | +380669989109 | 50000.00 |
19 | Kyiv | Nolay H. H. | +380935589110 | 110500.00 |
20 | Dnipro | Mildon P. N. | +380669989111 | 99999.00 |
x-----
Are you sure? (Y/any other input): y
Enter the record number to edit: 20

Edit:
1. Full record
2. Particular field
Your choice: 2
Select a field to edit:
1. City
2. Full Name
3. Phone Number
4. Sum of credit
5. Loan Date
6. First pay
Select an option: 6
Enter, whether the fine "Is paid?" (+/-): -
Press any key to continue . . . |
```

Рисунок 4. 6 – Редагування запису за окремими полями

Щоб видалити запис, виберіть відповідний пункт меню ввівши цифру 4 у головному меню програми та натиснувши клавішу „Enter”. (див. рис. 4.1). Функція виведе таблицю з записами у файлі, після цього на екрані з’явиться запит підтвердження. Для підтвердження операції натисніть „у”, а для скасування натисніть будь-яку іншу клавішу. Після підтвердження операції

виведеться під-меню, в якому необхідно вибрати, за яким критерієм необхідно видалити запис із файлу. Залежно від вибору викликається та чи інша функція видалення. Ця можливість представлена на рисунку 4.7.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 4
X-----
| No | City | FullName | Phone Number |
X-----
1 | Dnipro | NoIay H. H. | +380669989971 |
2 | Kyiv | Fallen B. E. | +380671589972 |
3 | Chernigiv | Mildon P. N. | +380679989973 |
4 | Odessa | Dorean D. D. | +380669989974 |
5 | Lviv | Korner B. T. | +380631234565 |
6 | Chernigiv | Mildon P. N. | +380661234566 |
7 | Lviv | NoIay H. H. | +380999876547 |
8 | Dnipro | Fallen B. E. | +380661234568 |
9 | Odessa | Korner B. T. | +380679989979 |
10 | Kyiv | Telon M. S. | +380671589101 |
11 | Chernigiv | NoIay H. H. | +380661234102 |
12 | Odessa | Telon M. S. | +380661234103 |
13 | Chernigiv | Dorean D. D. | +380671589104 |
14 | Kyiv | Mildon P. N. | +380669989105 |
15 | Lviv | Dorean D. D. | +380999876106 |
16 | Dnipro | Telon M. S. | +380679989107 |
17 | Lviv | Fallen B. E. | +380671589108 |
18 | Odessa | Korner B. T. | +380669989109 |
19 | Kyiv | NoIay H. H. | +380935589110 |
20 | Dnipro | Mildon P. N. | +380669989111 |
X-----
Are you sure? (Y/ any other input): y
Choose deletion method:
1. Delete by Full Name
2. Delete by Phone Number
Enter your choice: |
```

Рисунок 4.7 – Видалення запису із файлу

Щоб видалити запис за номером ПІБ, виберіть відповідний пункт під-меню ввівши цифру 1 у під-меню програми та натиснувши клавішу „Enter” (див рис. 4.7). Після цього треба ввести ПІБ, за яким необхідно видалити записи із файлу. Ця можливість представлена на рисунку 4.8. Після видалення вас автоматично поверне до головного меню.

```

Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 4
X-----X
| № | City | FullName | Phone Number | Sum of cr |
X-----X
1 | Dnipro | NoIay H. H. | +380669989971 | 100000 |
2 | Kyiv | Fallen B. E. | +380671589972 | 100000 |
3 | Chernigiv | Mildon P. N. | +380679989973 | 200000 |
4 | Odessa | Dorean D. D. | +380669989974 | 120500 |
5 | Lviv | Korner B. T. | +380631234565 | 560000 |
6 | Chernigiv | Mildon P. N. | +380661234566 | 150000 |
7 | Lviv | NoIay H. H. | +380999876547 | 400000 |
8 | Dnipro | Fallen B. E. | +380661234568 | 200000 |
9 | Odessa | Korner B. T. | +380679989979 | 340000 |
10 | Kyiv | Telon M. S. | +380671589101 | 550000 |
11 | Chernigiv | NoIay H. H. | +380661234102 | 110000 |
12 | Odessa | Telon M. S. | +380661234103 | 245000 |
13 | Chernigiv | Dorean D. D. | +380671589104 | 100000 |
14 | Kyiv | Mildon P. N. | +380669989105 | 990000 |
15 | Lviv | Dorean D. D. | +380999876106 | 750000 |
16 | Dnipro | Telon M. S. | +380679989107 | 105000 |
17 | Lviv | Fallen B. E. | +380671589108 | 800000 |
18 | Odessa | Korner B. T. | +380669989109 | 500000 |
19 | Kyiv | NoIay H. H. | +380935589110 | 110500 |
20 | Dnipro | Mildon P. N. | +380669989111 | 999900 |
21 | New-York | Mole D. F. | +388459652368 | 745500 |
X-----X
Are you sure? (Y/ any other input): y
Choose deletion method:
1. Delete by Full Name
2. Delete by Phone Number
Enter your choice: 1
Enter the full name to delete (format: 'Surname I. P.'): Mole D. F.
Client with name "Mole D. F." deleted successfully.
Press any key to continue . . .

```

Рисунок 4.8 – Видалення запису із файлу за ПІБ

Щоб видалити запис за номером телефону, виберіть відповідний пункт під-меню ввівши цифру 2 у під-меню програми та натиснувши клавішу „Enter” (див рис. 4.8). Після цього виведеться запитномеру телефону, за яким необхідно видалити запис із файлу. Ця можливість представлена на рисунку 4.9. Після видалення, натиснувши будь-яку клавішу, вас поверне до головного меню.

```

X-----X
Are you sure? (Y/ any other input): y
Choose deletion method:
1. Delete by Full Name
2. Delete by Phone Number
Enter your choice: 2
Enter the phone number of the client to delete (+38): +380669989111
Client with phone number "+380669989111" deleted successfully.
Press any key to continue . . .

```

Рисунок 4.9 – Видалення запису із файлу за номером телефону

Щоб відсортувати записи, виберіть відповідний пункт меню ввівши цифру 5 у головному меню програми та вписати «у» в запит підтвердження (див. рис.

4.1). Функція виведе під-меню, в якому необхідно вибрати, за яким критерієм необхідно відсортувати записи із файлу. Ця можливість представлена на рисунку 4.10. Залежно від вибору викликається та чи інша функція сортування.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 5
Are you sure? (Y/ any other input): y
Choose the parameter to sort by:
1. City (ascending)
2. Sum of credit (ascending)
Your choice: |
```

Рисунок 4.10 – Сортування записів у файлі

Щоб відсортувати записи у фалі за містом по алфавітному порядку, необхідно вибрати 1 пункт під-меню, як представлено на рисунку 4.11. Після цього файл буде автоматично відсортовано, як представлено на рисунку 4.12.

```
Menu:
0. Exit
1. View Records
2. Add Record
3. Edit Record
4. Delete Record
5. Sort Records
6. Make a request
Choose an option: 5
Are you sure? (Y/ any other input): y
Choose the parameter to sort by:
1. City (ascending)
2. Sum of credit (ascending)
Your choice: 1
Sorting by city (ascending) completed.
Press any key to continue . . . |
```

Рисунок 4.11 – Сортування записів у файлі за містом

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		46

X	No	City	FullName	Phone Number	Sum of credit	Total payable
X						
1	Chernigiv	Mildon P. N.	+380679989973	200000.00	220000.00	
2	Chernigiv	Mildon P. N.	+380661234566	15000.00	16500.00	
3	Chernigiv	NoIay H. H.	+380661234102	110000.00	121000.00	
4	Chernigiv	Dorean D. D.	+380671589104	1000.00	1100.00	
5	Dnipro	NoIay H. H.	+380669989971	1000.00	1100.00	
6	Dnipro	Fallen B. E.	+380661234568	200000.00	220000.00	
7	Dnipro	Telon M. S.	+380679989107	105000.00	115500.00	
8	Kyiv	Fallen B. E.	+380671589972	100000.00	110000.00	
9	Kyiv	Telon M. S.	+380671589101	550000.00	605000.00	
10	Kyiv	Mildon P. N.	+380669989105	990000.00	1089000.00	
11	Kyiv	NoIay H. H.	+380935589110	110500.00	121550.00	
12	Lviv	Korner B. T.	+380631234565	560000.00	616000.00	
13	Lviv	NoIay H. H.	+380999876547	40000.00	44000.00	
14	Lviv	Dorean D. D.	+380999876106	75000.00	82500.00	
15	Lviv	Fallen B. E.	+380671589108	800000.00	880000.00	
16	Odessa	Dorean D. D.	+380669989974	120500.00	132550.00	
17	Odessa	Korner B. T.	+380679989979	34000.00	37400.00	
18	Odessa	Telon M. S.	+380661234103	24500.00	26950.00	
19	Odessa	Korner B. T.	+380669989109	50000.00	55000.00	
X						

Рисунок 4.12 – Відсортовані записи у файлі за містом

Щоб відсортувати записи у файлі за сумою кредиту по зростанню, необхідно вибрати 2 пункт під-меню, як представлено на рисунку 4.10. Після цього файл буде автоматично відсортований, як наведено на рисунку 4.13.

X	№	City	FullName	Phone Number	Sum of credit	Total payable
X						
1	Chernigiv	Dorean D. D.	+380671589104	1000.00	1100.00	
2	Dnipro	NoIay H. H.	+380669989971	1000.00	1100.00	
3	Chernigiv	Mildon P. N.	+380661234566	15000.00	16500.00	
4	Odessa	Telon M. S.	+380661234103	24500.00	26950.00	
5	Odessa	Korner B. T.	+380679989979	34000.00	37400.00	
6	Lviv	NoIay H. H.	+380999876547	40000.00	44000.00	
7	Odessa	Korner B. T.	+380669989109	50000.00	55000.00	
8	Lviv	Dorean D. D.	+380999876106	75000.00	82500.00	
9	Kyiv	Fallen B. E.	+380671589972	100000.00	110000.00	
10	Dnipro	Telon M. S.	+380679989107	105000.00	115500.00	
11	Chernigiv	NoIay H. H.	+380661234102	110000.00	121000.00	
12	Kyiv	NoIay H. H.	+380935589110	110500.00	121550.00	
13	Odessa	Dorean D. D.	+380669989974	120500.00	132550.00	
14	Chernigiv	Mildon P. N.	+380679989973	200000.00	220000.00	
15	Dnipro	Fallen B. E.	+380661234568	200000.00	220000.00	
16	Kyiv	Telon M. S.	+380671589101	550000.00	605000.00	
17	Lviv	Korner B. T.	+380631234565	560000.00	616000.00	
18	Lviv	Fallen B. E.	+380671589108	800000.00	880000.00	
19	Kyiv	Mildon P. N.	+380669989105	990000.00	1089000.00	
X						

Рисунок 4.13 – Відсортовані записи у файлі за сумою кредиту

Щоб зробити запити до вмісту файлу необхідно вибрати відповідний пункт меню ввівши цифру 6 у головному меню програми та натиснувши клавішу „Enter” (див. рис. 4.1). Функція виведе під-меню у якому можна вибрати які самі дані ви хочете отримати з файлу. Ця можливість представлена на рисунку 4.14.


```

Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: |

```

Рисунок 4.14 – Під-меню запитів до файлу

Для того щоб знайти записи за ПІБ, необхідно в під-меню програми обрати пункт 1 та ввести необхідні дані для пошуку. У випадку якщо знайдено хоча б 1 запис у файлі за вказаними вимогами, його буде виведено на екран. Ця можливість наведена на рисунку 4.15.

```

Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 1
Enter the full name to search: Dorean D. D.
X-----
| № | City | FullName | Phone Number | Sum of cre
X-----
| 1 | Chernigiv | Dorean D. D. | +380671589104 | 1000
| 2 | Lviv | Dorean D. D. | +380999876106 | 75000
| 3 | Odessa | Dorean D. D. | +380669989974 | 120500
X-----
Client Dorean D. D. has 3 credit(s).
Press any key to continue . . . |

```

Рисунок 4.15 – Введення даних для пошуку та сам пошук

У випадку коли не має жодного запису який би задовольнив вказані критерії буде виведено відповідне повідомлення. Ця можливість наведена на рисунку 4.16.

```

Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 1
Enter the full name to search: Gf G. G.
Client Gf G. G. has no credits.
Press any key to continue . . . |

```

Рисунок 4.16 – Результат пошуку при невідповідності даних

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		48

Після виконання запиту користувача автоматично поверне до головного меню де він зможе продовжити роботу з програмою.

Для того щоб знайти суму максимального та мінімального кредиту необхідно в під-меню програми обрати пункт 2 та ввести необхідні дані для пошуку. У випадку якщо знайдено хоча б 1 запис у файлі за вказаними вимогами, його буде виведено на екран. Ця можливість наведена на рисунку 4.17.

```
Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 2
The minimum credit body: 1000.00
The maximum credit body: 990000.00
Press any key to continue . . . |
```

Рисунок 4.17 – Приклад введення даних для пошуку та сам пошук

Для того щоб вивести усі кредити за визначений період необхідно в під-меню програми обрати пункт 3, як зображено на рисунку 4.18. При відсутності кредитів у визначеному проміжку виводиться спеціальне повідомлення. Ця можливість наведена на рисунку 4.19.

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				49
Змн.	Арк.	№ докум.	Підпис	Дат		

```

Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 3
Enter the "first date" (DD.MM.YYYY): 03.05.2021
Enter the "second date" (DD.MM.YYYY): 16.09.2023
X-----X
| № | City | FullName | Phone Number |
X-----X
| 1 | Chernigiv | Dorean D. D. | +380671589104 |
| 2 | Dnipro | Nolay H. H. | +380669989971 |
| 3 | Chernigiv | Mildon P. N. | +380661234566 |
| 4 | Odessa | Korner B. T. | +380679989979 |
| 5 | Odessa | Korner B. T. | +380669989109 |
| 6 | Kyiv | Fallen B. E. | +380671589972 |
| 7 | Kyiv | Telon M. S. | +380671589101 |
| 8 | Lviv | Korner B. T. | +380631234565 |
| 9 | Lviv | Fallen B. E. | +380671589108 |
| 10 | Kyiv | Mildon P. N. | +380669989105 |
X-----X
There are 10 record(s).
Press any key to continue . . . |

```

Рисунок 4.18 – Перелік кредитів за визначений період

```

Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 3
Enter the "first date" (DD.MM.YYYY): 03.05.2021
Enter the "second date" (DD.MM.YYYY): 03.05.2021
There are no records in this range :(
Press any key to continue . . . |

```

Рисунок 4.19 – Вивід відсутності кредитів

Для того щоб вивести усіх боржників необхідно в під-меню програми обрати пункт 4, як на рисунку 4.20.

```

Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 4
X-----X
| № | City | FullName | Phone Number | Sum of credit | Total payable | Monthly Payment | First pay | loanDate |
X-----X
| 1 | Chernigiv | Dorean D. D. | +380671589104 | 1000.00 | 1100.00 | 91.67 | - | 15.09.2023 |
| 2 | Odessa | Telon M. S. | +380661234103 | 24500.00 | 26950.00 | 2245.83 | - | 18.09.2023 |
| 3 | Lviv | Dorean D. D. | +380999876106 | 75000.00 | 82500.00 | 6875.00 | - | 17.09.2023 |
| 4 | Kyiv | Nolay H. H. | +380935589110 | 110500.00 | 121550.00 | 10129.17 | - | 17.09.2023 |
| 5 | Chernigiv | Mildon P. N. | +380679989973 | 200000.00 | 220000.00 | 18333.33 | - | 17.09.2023 |
| 6 | Kyiv | Telon M. S. | +380671589101 | 550000.00 | 605000.00 | 50416.67 | - | 16.09.2023 |
| 7 | Lviv | Korner B. T. | +380631234565 | 560000.00 | 616000.00 | 51333.33 | - | 15.09.2023 |
| 8 | Lviv | Fallen B. E. | +380671589108 | 800000.00 | 880000.00 | 73333.34 | - | 15.09.2023 |
X-----X
There are 8 debtors.
Press any key to continue . . . |

```

Рисунок 4.20 – Вивід боржників банку

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				
Змн.	Арк.	№ докум.	Підпис	Дат		50

Для того щоб знайти дохід банку за місяць необхідно в під-меню програми обрати пункт 5 та ввести необхідні дані для пошуку, як на рисунку 4.21.

```
Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 5

The bank's total profit (for the year) is 4495150.00

Press any key to continue . . . |
```

Рисунок 4.21 – Вивід загального прибутку банку

Також у програмі передбачена перевірка даних введених користувачем на коректність, у разі введення некоректних даних користувач отримує повідомлення про помилку, та запит на повторне введення даних, як наведено на рисунку 4.22.

```
Make a choice what to view:
0. Exit
1. Loans by client's name
2. MAX and MIN loan
3. Loans by a certain period
4. All debtors
5. Bank's total profit (for the year)
Choose an option: 1
Enter the full name to search: 564
Error: Full name must be entered in the format 'Surname I. P.'
Enter the full name to search: |
```

Рисунок 4.22 – Перевірка на коректність даних

Щоб вийти з програми треба спочатку вийти у головне меню за допомогою клавіші «0», а потім вже у головному меню натиснути «0».

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				51
Змн.	Арк.	№ докум.	Підпис	Дат		

ВИСНОВКИ

Програмний продукт має призначення бути використаним у банківських установах для автоматизації обліку видачі кредитів.

Програма надає користувачу детальну інформацію про кредити, що були занесені у систему, а саме: дані клієнта, суму кредиту, строк погашення, відсоткову ставку, статус кредиту та іншу інформацію, яка є важливою для ведення фінансового обліку.

Розроблена програма виконує облік кредитів та дозволяє виконувати різноманітні операції: додавання, редагування, видалення записів, пошук, сортування, виконання запитів та перегляд результатів. Також програма забезпечує перевірку валідності введених даних, збереження унікальності записів і обробку помилок.

При розробці даної програми вдосконалив свої навички роботи з файлами, створення структури проекту без використання глобальних змінних, та забезпечення валідності введених даних за допомогою регулярних виразів. Навчився створювати ефективні алгоритми для пошуку, сортування та обробки запитів. Окрім того, удосконалив вміння працювати із середовищем розробки, включаючи налагодження програмного коду, обробку помилок та оптимізацію роботи програми.

	Вик.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				52
Змн.	Арк.	№ докум.	Підпис	Дат		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 С++ reference [Електронний ресурс] / С++ reference – Режим доступа: <http://cppstudio.com/>

2 Страуструп, Бйорн «Програмування. Принцип та практика використання С++» [Текст]/ Б. Страуструп – С-Пб.: «БХВ - Петербург», 2011 - 1206 с.

3 Пахомов, Б. С/С++ и MS Visual С++ 2011 для начинающих[текст] / Б.Пахомов – С-ПБ.: «БХВ-Петербург»,2011. – 736 с.

4 Мюррей, У. Х. «Налагодження в С++» [Текст] / У.Х. Мюррей, К. Х. Паппас – Бином, 2009. – 509с

5 С++ reference [Електронний ресурс] / С++ reference - Режим доступа: <http://en.cppreference.com/w/>

6 С++ [Електронний ресурс] Вікіпедія – Режим доступа: <http://ru.wikipedia.org/wiki/C+>

	Вук.	Бєдін Т. В.			КП.ПЗ.221.03.ПЗ	Арк.
	Пер.	Сайко Т.С.				53
Змн.	Арк.	№ докум.	Підпис	Дат		

ДОДАТОК А

ВХІДНІ ДАНІ

Список вкладників банку (див. рис. А1)

№	City	FullName	Phone Number	Sum of credit	Total payable	Monthly Payment	First pay	LoanDate
1	Chernigiv	Dorean D. D.	+380671589104	1000.00	1100.00	91.67	-	15.09.2023
2	Dnipro	NoIay H. H.	+380669989971	1000.00	1100.00	91.67	+	15.09.2023
3	Chernigiv	Mildon P. N.	+380661234566	15000.00	16500.00	1375.00	+	16.09.2023
4	Odessa	Telton M. S.	+380661234103	24500.00	26950.00	2245.83	-	18.09.2023
5	Odessa	Korner B. T.	+380679989979	34000.00	37400.00	3116.67	+	15.09.2023
6	Lviv	NoIay H. H.	+380999876547	40000.00	44000.00	3666.67	+	17.09.2023
7	Odessa	Korner B. T.	+380669989109	50000.00	55000.00	4583.33	+	16.09.2023
8	Lviv	Dorean D. D.	+380999876106	75000.00	82500.00	6875.00	-	17.09.2023
9	Dnipro	Mildon P. N.	+380669989111	99999.00	109998.90	9166.58	+	18.09.2023
10	Kyiv	Fallen B. E.	+380671589972	100000.00	110000.00	9166.67	+	16.09.2023
11	Dnipro	Telton M. S.	+380679989107	105000.00	115500.00	9625.00	+	18.09.2023
12	Chernigiv	NoIay H. H.	+380661234102	110000.00	121000.00	10083.33	+	17.09.2023
13	Kyiv	NoIay H. H.	+380935589110	110500.00	121550.00	10129.17	-	17.09.2023
14	Odessa	Dorean D. D.	+380669989974	120500.00	132550.00	11045.83	+	18.09.2023
15	Chernigiv	Mildon P. N.	+380679989973	200000.00	220000.00	18333.33	+	17.09.2023
16	Dnipro	Fallen B. E.	+380661234568	200000.00	220000.00	18333.33	+	18.09.2023
17	Kyiv	Telton M. S.	+380671589101	550000.00	605000.00	50416.67	-	16.09.2023
18	Lviv	Korner B. T.	+380631254565	560000.00	616000.00	51333.33	-	15.09.2023
19	Lviv	Fallen B. E.	+380671589108	800000.00	860000.00	73333.34	-	15.09.2023
20	Kyiv	Mildon P. N.	+380669989105	990000.00	1089000.00	90750.00	+	16.09.2023

Рисунок А.1 – Вхідні дані вкладників

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМИ

Код програми наданий у лістингу Б.1.

Лістинг Б.1 – Код програми

```
#define _CRT_SECURE_NO_WARNINGS
#include <Windows.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstring>
#include <string>
#include <regex>

using namespace std;
#define PATH "data.dat"

struct Client {
    char city[20];
    char fullName[30];
    char phoneNumber[15];
    float loanBody;
    float totalAmount;
    float monthlyPayment;
    char loanDate[11];
    bool isPaid;
};

// Перевірка існування файлу
bool FileExists(const string& path) {
    ifstream file(path);
    return file.good();
}

// Створення файлу, якщо файл відсутній
void EnsureFileExists(const string& path) {
    if (!FileExists(path)) {
        ofstream file(path, ios::binary);
        file.close();
    }
}

// Запис клієнта у файл
void WriteClient(const string& path, const Client& client) {
    ofstream file(path, ios::binary | ios::app);
    if (!file) { // Перевірка, чи файл відкрився успішно
        cerr << "Error: Could not open file for writing!" << endl;
        return;
    }
    file.write((const char*)&client, sizeof(Client));
    file.close();
}

// Перевірка наявності записів у файлі
```



```

bool HasRecords(const string& path) {
    ifstream file(path, ios::binary);
    file.seekg(0, ios::end);
    return file.tellg() > 0;
}

// Перевірка на унікальність номера телефону
bool IsUniquePhoneNumber(const string& path, const char* phoneNumber) {
    ifstream file(path, ios::binary);
    Client current;
    while (file.read((char*)&current, sizeof(Client))) {
        if (strcmp(current.phoneNumber, phoneNumber) == 0) {
            return false;
        }
    }
    return true;
}

// Введення суми кредиту з перевіркою (діапазон від 1,000 до 1,000,000)
void InputLoanBody(float& loanBody) {
    regex validAmountPattern("^\\d+(\\.\\d+)?$"); // Регулярний вираз для числа
    // (ціле або з плаваючою комою)
    string input;

    while (true) {
        cout << "Enter the \"Sum of credit\" (from 1,000 to 1,000,000) - ";
        getline(cin, input);

        // Перевірка на коректність введенного числа
        if (!regex_match(input, validAmountPattern)) {
            cout << "Error: You must enter a valid number!" << endl;
            continue;
        }

        // Конвертуємо введенний рядок в число
        loanBody = stof(input);

        // Перевірка діапазону
        if (loanBody >= 1000 && loanBody <= 1000000) {
            break; // Якщо число в межах діапазону, виходимо з циклу
        }
        else {
            cout << "Error: Amount is not in the acceptable range!" << endl;
        }
    }
}

// Перевірка на правильність вводу дати
bool IsValidDate(const char date[]) {
    bool isLeap;

    // Перевірка довжини (формат DD.MM.YYYY повинен мати рівно 10 символів)
    if (strlen(date) != 10) {
        return false;
    }

    // Перевірка на правильні роздільники (дата повинна мати '.' на 3-й і 6-й
    // позиціях)
    if (date[2] != '.' || date[5] != '.') {
        return false;
    }

    // Отримання дня, місяця та року
    int day = atoi(string(date, 2).c_str());

```

```

int month = atoi(string(date + 3, 2).c_str());
int year = atoi(string(date + 6, 4).c_str());

// Перевірка діапазонів для дня, місяця та року
if (day < 1 || day > 31 || month < 1 || month > 12 || year < 1900) {
    return false;
}

// Перевірка днів у кожному місяці
switch (month) {
case 4: case 6: case 9: case 11:
    if (day > 30) return false;
    break;
case 2:
    // Визначення, чи є рік високосним
    isLeap = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    if (isLeap && day > 29) return false;
    if (!isLeap && day > 28) return false;
    break;
default:
    if (day > 31) return false;
}

return true;
}

// Функція для перевірки довжини рядка
bool isValidLength(const string& str, size_t maxLength) {
    return str.length() <= maxLength;
}

// Функція форматування назви міста
string formatCityName() {
    const size_t maxCityLength = 16; // Максимальна довжина назви міста
    string result;

    while (true) {
        getline(cin, result);

        // Перевірка довжини
        if (!isValidLength(result, maxCityLength)) {
            cout << "Error: The city name cannot exceed "
                 << maxCityLength << " characters. Try again: ";
            continue; // Повертаємось до повторного вводу
        }

        // Форматування: кожне слово починається з великої літери
        bool capitalize = true;
        bool isValid = true; // Змінна для відслідковування валідності вводу
        for (size_t i = 0; i < result.length(); ++i) {
            if (isalpha(result[i])) {
                result[i] = capitalize ? toupper(result[i]) : tolower(result[i]);
                capitalize = false;
            }
            else if (result[i] == ' ' || result[i] == '-') {
                capitalize = true;
            }
            else
            {
                cout << "Error: The city name cannot have numbers or special
symbols. Try again: ";
                isValid = false; // Встановлюємо прапорець невдалого вводу
                break; // Виходимо з for, щоб перевірити isValid
            }
        }
    }
}

```



```

        else if (input == "-") {
            client.isPaid = false;
        }
        break; // Коректне введення
    }
    else {
        cout << "Error: Input must be '+' or '-' only. Try again." << endl;
    }
}

// Введення суми кредиту
void InputLoanDate(Client& client) {
    do {
        while (true) {
            cout << "Please enter the date in DD.MM.YYYY format: ";

            string input; // Використовуємо std::string для зберігання вводу
            getline(cin, input); // Читання всього рядка

            // Перевірка на довжину введенного рядка
            if (input.length() > 10) {
                cout << "Error: Input exceeds maximum allowed length (10
characters). Try again.\n";
                continue; // Повторний запит на введення
            }

            // Копіювання в масив char для збереження у loanDate
            strncpy(client.loanDate, input.c_str(), 10);
            client.loanDate[10] = '\0'; // Забезпечуємо коректний термінатор

            break; // Вихід з циклу, якщо ввід правильний
        }
        if (IsValidDate(client.loanDate)) {
            return;
        }
    } while (true);
}

// Функція для введення та перевірки номера телефону
void ValidatePhoneNumber(Client& client)
{
    regex phonePattern("[0-9]{10}$"); // Шаблон: рівно 10 цифр
    string inputPhone;

    // Цикл для введення номера телефону
    while (true) {
        cout << "Enter the \"Phone number\" (10 digits): +38";
        getline(cin, inputPhone);

        // Перевірка введенного номеру за шаблоном
        if (!regex_match(inputPhone, phonePattern)) {
            cout << "Error: Phone number must consist of exactly 10 digits." <<
endl;
            continue;
        }

        // Запис номеру в client.phoneNumber
        strncpy(client.phoneNumber, inputPhone.c_str(),
sizeof(client.phoneNumber) - 1);
        client.phoneNumber[sizeof(client.phoneNumber) - 1] = '\0'; // Гарантуємо,
що рядок завершений
    }
}

```

```

        break;
    }
}
// Введення даних клієнта з перевітками
void InputClient(Client& client, const string& path) {

    SityInput(client);
    ValidateFullName(client.fullName);
    ValidatePhoneNumber(client);

    InputLoanBody(client.loanBody);
    client.totalAmount = client.loanBody * 1.1;
    client.monthlyPayment = client.totalAmount / 12;

    InputLoanDate(client);
    InputIsPaid(client);
}

// Виведення списку клієнтів
void ViewClients(const string& path) {
    if (!HasRecords(path)) {
        cout << "File is empty." << endl;
        return;
    }

    ifstream file(path, ios::binary);
    if (!file) {
        cout << "Error in file's opening" << endl;
        return;
    }

    cout << "x-----X"
    << endl;
    cout << "| № | City | FullName | Phone Number |
Sum of credit | Total payable | Monthly Payment | First pay | loanDate |"
    << endl;
    cout << "x-----X"
    << endl;

    Client current;
    size_t i = 0;

    while (file.read((char*)&current), sizeof(Client)) {

        cout << "| " << setw(3) << left << ++i << " | "
            << setw(17) << left << current.city << "| "
            << setw(20) << left << current.fullName << " | +38"
            << setw(11) << left << current.phoneNumber << " | "
            << setw(14) << right << fixed << setprecision(2) << current.loanBody
        << " | "
            << setw(15) << right << fixed << setprecision(2) <<
current.totalAmount << " | "
            << setw(17) << right << fixed << setprecision(2) <<
current.monthlyPayment << " | "
            << setw(3) << (current.isPaid ? "+" : "-") << " |"
            << setw(11) << right << fixed << setprecision(2) << current.loanDate
        << " | "
            << endl;
    }
}

```

```

        cout << "x-----X"
    << endl;
    file.close();
}
// Редагування запису цілком
void EditClientRecord(Client& client) {
    cout << "Edit a customer record." << endl;
    cout << "Enter new data for the client: " << endl;
    InputClient(client, PATH);
}
// Редагування окремих полів запису
void EditClientField(Client& client) {
    string choiceInput;
    cout << "Select a field to edit: " << endl;
    cout << "1. City" << endl;
    cout << "2. Full Name" << endl;
    cout << "3. Phone Number" << endl;
    cout << "4. Sum of credit" << endl;
    cout << "5. Loan Date" << endl;
    cout << "6. First pay" << endl;
    cout << "Select an option: ";

    regex validChoicePattern("^[1-6]$"); // Регулярний вираз для вибору числа
    від 1 до 6

    while (true) {
        getline(cin, choiceInput); // Отримуємо введення як рядок

        // Перевірка, чи введення відповідає патерну
        if (regex_match(choiceInput, validChoicePattern)) {
            int choice = stoi(choiceInput); // Перетворюємо введенне значення в
ціле число

            switch (choice) {
                case 1:
                    SityInput(client);
                    return;
                case 2:
                    ValidateFullName(client.fullName);
                    return;
                case 3:
                    ValidatePhoneNumber(client);
                    return;
                case 4:
                    InputLoanBody(client.loanBody);
                    client.totalAmount = client.loanBody * 1.1;
                    client.monthlyPayment = client.totalAmount / 12;
                    return;
                case 5:
                    InputLoanDate(client);
                    return;
                case 6:
                    InputIsPaid(client);
                    return;
                default:
                    cout << "Invalid choice. Try again: "; // По факту default не
потрібен, але залишаю його для можливих подальших модифікацій програми.
            }
        }
        else {
            cout << "Invalid choice. Try again (1-6): ";

```

```

    }
}
}
void EditClient(const string& path) {
    if (!HasRecords(path)) {
        cout << "File is empty. No records to edit." << endl;
        return;
    }

    ViewClients(path); // Displaying the list of clients before editing

    string input;
    cout << "Are you sure? (Y/any other input): ";
    getline(cin, input);

    if (input != "Y" && input != "y") {
        cout << "Your choice was canceled :)" << endl;
        return;
    }

    string recordNumberInput;
    int recordNumber;

    fstream file(path, ios::binary | ios::in | ios::out);
    if (!file) {
        cout << "Error: Cannot open file." << endl;
        return;
    }

    Client client;

    while (true) {
        try {
            cout << "Enter the record number to edit: ";
            getline(cin, recordNumberInput);

            recordNumber = stoi(recordNumberInput);

            file.seekg(0, ios::end);
            size_t numRecords = file.tellg() / sizeof(Client);
            if (recordNumber < 1 || recordNumber > static_cast<int>(numRecords))
            {
                cout << "Error: Invalid record number." << endl;
                continue;
            }
            break;
        } catch (const invalid_argument&) {
            cout << "Error: Invalid input. Please enter a valid number." << endl;
        }
        catch (const out_of_range&) {
            cout << "Error: Number is out of range." << endl;
        }
    }

    // Read the specified record
    file.seekg((recordNumber - 1) * sizeof(Client), ios::beg);
    file.read(reinterpret_cast<char*>(&client), sizeof(Client));

    // Allow the user to edit the record
    int choice;
    cout << "\nEdit:\n1. Full record\n2. Particular field\nYour choice: ";

```

```

string choiceInput;
regex validChoicePattern("^[12]$");

while (true) {
    getline(cin, choiceInput);

    if (regex_match(choiceInput, validChoicePattern)) {
        choice = stoi(choiceInput);
        break;
    }
    else {
        cout << "Wrong choice. Try again: ";
    }
}

if (choice == 1) {
    EditClientRecord(client);
}
else if (choice == 2) {
    EditClientField(client);
}

// Write the modified record back to the file
file.seekp((recordNumber - 1) * sizeof(Client), ios::beg);
file.write(reinterpret_cast<const char*>(&client), sizeof(Client));

file.close();
}

void DeleteRecordByName() {
    FILE* pfile;
    FILE* tempFile;
    Client a;
    bool found = false; // To track if a record is found for deletion

    char fullNameToDelete[30];
    string inputName;
    regex fullNamePattern("([A-Z]{1}[a-z]{1,20} [A-Z]{1}\\.[ ]{1}[A-Z]{1}\\.)");
    // Формат "Surname I. P."

    while (true) {
        cout << "Enter the full name to delete (format: 'Surname I. P.'): ";
        getline(cin, inputName);

        // Видалення зайвих пробілів
        inputName = regex_replace(inputName, regex("^ +| +$|( ) +"), "$1");

        // Перевірка введення
        if (regex_match(inputName, fullNamePattern)) {
            break;
        }
        else {
            cout << "Error: Full name must be entered in the format 'Surname I. P.' (Surname max: 20)\n";
        }
    }

    strncpy(fullNameToDelete, inputName.c_str(), 30);

    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }
}

```



```

    }

    fopen_s(&tempFile, "temp.dat", "wb");
    if (!tempFile) {
        cerr << "Failed to create a temporary file!" << endl;
        fclose(pfile);
        return;
    }

    // Read each record and copy it to the temp file if it doesn't match the name
    to delete
    while (fread(&a, sizeof(Client), 1, pfile) == 1) {
        if (strcmp(a.fullName, fullNameToDelete) != 0) {
            fwrite(&a, sizeof(Client), 1, tempFile);
        }
        else {
            found = true; // Mark that we found and skipped a record
        }
    }

    fclose(pfile);
    fclose(tempFile);

    // Replace the original file with the temp file
    remove(PATH);
    rename("temp.dat", PATH);

    if (found) {
        cout << "Client with name \"" << fullNameToDelete << "\" deleted
successfully." << endl;
    }
    else {
        cout << "No records found with the name \"" << fullNameToDelete << "\"
for deletion." << endl;
    }
}

void DeleteRecordByPhone(const string& path) {
    string inputPhoneNumber;
    char phoneNumberToDelete[15];
    regex phoneNumberPattern("^\\d{10}$"); // Регулярний вираз для перевірки 9
цифр після "+38"
    bool found = false; // To track if a record is found for deletion

    // Введення та перевірка номера телефону
    while (true) {
        cout << "Enter the phone number of the client to delete (+38): +38";
        getline(cin, inputPhoneNumber);

        // Видаляємо зайві пробіли
        inputPhoneNumber = regex_replace(inputPhoneNumber, regex("^ +| +$|( )
+"), "$1");

        if (regex_match(inputPhoneNumber, phoneNumberPattern)) {
            break; // Номер коректний
        }
        else {
            cout << "Error: Phone number must contain exactly 10 digits after
'+38'.\n";
        }
    }

    strncpy(phoneNumberToDelete, inputPhoneNumber.c_str(),
sizeof(phoneNumberToDelete));

```

```

FILE* pfile;
FILE* tempFile;
Client client;

fopen_s(&pfile, PATH, "rb");
if (!pfile) {
    cerr << "Failed to open the file!" << endl;
    return;
}

fopen_s(&tempFile, "temp.dat", "wb");
if (!tempFile) {
    cerr << "Failed to create a temporary file!" << endl;
    fclose(pfile);
    return;
}

// Read each record and copy it to the temp file if it doesn't match the
phone number to delete
while (fread(&client, sizeof(Client), 1, pfile) == 1) {
    if (strcmp(client.phoneNumber, phoneNumberToDelete) != 0) {
        fwrite(&client, sizeof(Client), 1, tempFile);
    }
    else {
        found = true; // Mark that we found and skipped a record
    }
}

fclose(pfile);
fclose(tempFile);

// Replace the original file with the temp file
remove(path.c_str());
rename("temp.dat", path.c_str());

if (found) {
    cout << "Client with phone number \""+38" << phoneNumberToDelete << "\"
deleted successfully." << endl;
}
else {
    cout << "No records found with the phone number \""+38" <<
phoneNumberToDelete << "\" for deletion." << endl;
}
}

void DeleteClient(const string& path) {
    if (!HasRecords(path)) {
        cout << "File is empty. No records to delete." << endl;
        return;
    }

    string input;
    regex validChoicePattern("^[12]$"); // Дозволяємо тільки '1' або '2'

    int choice;

    // Отримуємо і перевіряємо вибір користувача
    while (true) {
        cout << "Choose deletion method:\n1. Delete by Full Name\n2. Delete by
Phone Number\nEnter your choice: ";
        getline(cin, input);

        // Видаляємо зайві пробіли на початку та в кінці

```

```

        input = regex_replace(input, regex("^ +| +$|( ) +"), "$1");

        // Перевірка вводу
        if (regex_match(input, validChoicePattern)) {
            choice = stoi(input); // Конвертуємо введення в число
            break;
        }
        else {
            cout << "Error: Please enter '1' or '2' only." << endl;
        }
    }

    switch (choice) {
    case 1: {
        DeleteRecordByName();
        break;
    }
    case 2: {
        DeleteRecordByPhone(PATH);
        break;
    }
    default:
        cout << "Invalid choice. Please enter 1 or 2." << endl;
        break;
    }
}

void SortRecordsByCity() {
    FILE* pfile;
    Client a, b;
    bool swapped;

    fopen_s(&pfile, PATH, "rb+");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }

    // Count records
    int recordCount = 0;
    while (fread(&a, sizeof(Client), 1, pfile) == 1) {
        recordCount++;
    }

    if (recordCount < 2) {
        cout << "No sorting needed. The file has less than 2 records." << endl;
        fclose(pfile);
        return;
    }

    // Sort records using bubble sort
    do {
        swapped = false;
        rewind(pfile); // Return to the beginning of the file

        for (int i = 0; i < recordCount - 1; i++) {
            // Read two consecutive records
            fread(&a, sizeof(Client), 1, pfile);
            long posA = ftell(pfile) - sizeof(Client); // Position of the first
record
            fread(&b, sizeof(Client), 1, pfile);
            long posB = ftell(pfile) - sizeof(Client); // Position of the second
record

```

```

        // Compare city names (case-insensitive) in ascending order
        if (_stricmp(a.city, b.city) > 0) { // Use _stricmp for case-
insensitive comparison
            // Swap records
            fseek(pfile, posA, SEEK_SET); // Position to write the first
record
            fwrite(&b, sizeof(Client), 1, pfile); // Write the second record
at the first position
            fseek(pfile, posB, SEEK_SET); // Position to write the second
record
            fwrite(&a, sizeof(Client), 1, pfile); // Write the first record
at the second position
            swapped = true;
        }

        // Move the file pointer to the next record pair
        fseek(pfile, posA + sizeof(Client), SEEK_SET);
    }
} while (swapped); // Continue until no swaps are made

fclose(pfile);
cout << "Sorting by city (ascending) completed." << endl;
}

void SortRecordsBySumOfCredit() {
    FILE* pfile;
    Client a, b;
    bool swapped;

    fopen_s(&pfile, PATH, "rb+");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }

    // Count records
    int recordCount = 0;
    while (fread(&a, sizeof(Client), 1, pfile) == 1) {
        recordCount++;
    }

    if (recordCount < 2) {
        cout << "No sorting needed. The file has less than 2 records." << endl;
        fclose(pfile);
        return;
    }

    // Sort records using bubble sort
    do {
        swapped = false;
        rewind(pfile); // Return to the beginning of the file

        for (int i = 0; i < recordCount - 1; i++) {
            // Read two consecutive records
            fread(&a, sizeof(Client), 1, pfile);
            long posA = ftell(pfile) - sizeof(Client); // Position of the first
record
            fread(&b, sizeof(Client), 1, pfile);
            long posB = ftell(pfile) - sizeof(Client); // Position of the second
record

            // Compare sum of credits in ascending order
            if (a loanBody > b loanBody) {

```

```

        // Swap records
        fseek(pfile, posA, SEEK_SET); // Position to write the first
record
        fwrite(&b, sizeof(Client), 1, pfile); // Write the second record
at the first position
        fseek(pfile, posB, SEEK_SET); // Position to write the second
record
        fwrite(&a, sizeof(Client), 1, pfile); // Write the first record
at the second position
        swapped = true;
    }

    // Move the file pointer to the next record pair
    fseek(pfile, posA + sizeof(Client), SEEK_SET);
}
} while (swapped); // Continue until no swaps are made

fclose(pfile);
cout << "Sorting by loan body (ascending) completed." << endl;
}

void SortRecords() {
    string input;
    int choice;
    regex validChoicePattern("^[12]$"); // Дозволяємо тільки "1" або "2"

    // Отримуємо і перевіряємо вибір користувача
    while (true) {
        cout << "Choose the parameter to sort by:\n1. City (ascending)\n2. Sum of
credit (ascending)\nYour choice: ";
        getline(cin, input);

        // Видаляємо зайві пробіли
        input = regex_replace(input, regex("^ +| +$|( ) +"), "$1");

        if (regex_match(input, validChoicePattern)) {
            choice = stoi(input); // Конвертуємо введення в число
            break;
        }
        else {
            cout << "Error: Please enter '1' or '2' only." << endl;
        }
    }

    switch (choice) {
        case 1:
            SortRecordsByCity();
            break;
        case 2:
            SortRecordsBySumOfCredit();
            break;
        default:
            cout << "Invalid choice!" << endl; // Цей варіант неможливий завдяки
перевірці вводу
    }
}

int OnlyOneRecord(const string& path)
{
    ifstream file(path, ios::binary);

    file.seekg(0, ios::end); // Переміщаємось в кінець файлу
    int fileSize = file.tellg(); // Отримуємо розмір файлу в байтах
    file.close();
}

```

```

        return fileSize / sizeof(Client); // Пахуємо кількість записів
    }

void FIRST_ByClientName(const string& path) {
    FILE* pfile;
    Client a;

    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }

    if (!HasRecords(path)) {
        cout << "File is empty. No records to search." << endl;
        fclose(pfile);
        return;
    }

    string inputFullName;
    char fullNameToSearch[30];
    regex fullNamePattern("([A-Z]{1}[a-z]{1,20} [A-Z]{1}\\.[ ]{1}[A-Z]{1}\\.)");
    // Формат "Surname I. P."

    // Введення та перевірка імені
    while (true) {
        cout << "Enter the full name to search: ";
        getline(cin, inputFullName);

        // Видалення зайвих пробілів
        inputFullName = regex_replace(inputFullName, regex("^ +| +$|( ) +"),
"$1");

        if (regex_match(inputFullName, fullNamePattern)) {
            break;
        }
        else {
            cout << "Error: Full name must be entered in the format 'Surname I.
P.'\n";
        }
    }

    strncpy(fullNameToSearch, inputFullName.c_str(), sizeof(fullNameToSearch));

    int count = 0;
    bool found = false;
    size_t i = 0;

    cout << fixed << setprecision(2);

    while (fread(&a, sizeof(Client), 1, pfile) == 1) {
        if (strcmp(a.fullName, fullNameToSearch) == 0) {
            if (!found) {
                cout << "X-----
-----
X" << endl;

                cout << "| № | City | FullName | Phone Number
| Sum of credit | Total payable | Monthly Payment | First pay | loanDate
|" << endl;

                cout << "X-----
-----
X" << endl;
            }

```

```

        found = true;
    }

    cout << "| " << setw(3) << left << ++i << " | "
        << setw(10) << left << a.city << "| "
        << setw(19) << left << a.fullName << " | +38"
        << setw(11) << left << a.phoneNumber << " | "
        << setw(14) << right << a.loanBody << " | "
        << setw(15) << right << a.totalAmount << " | "
        << setw(17) << right << a.monthlyPayment << " | "
        << setw(3) << (a.isPaid ? "+" : "-") << " | "
        << setw(11) << right << a.loanDate << " | "
        << endl;

    count++;
}

if (found) {
    cout << "X-----X" <<
endl;
    cout << "Client " << fullNameToSearch << " has " << count << "
credit(s)." << endl;
}
else {
    cout << "Client " << fullNameToSearch << " has no credits." << endl;
}

fclose(pfile);
}

void Second_MAX_MIN(const string& path)
{
    FILE* pfile;
    Client a;
    float min = 10000, max = -1;

    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }
    if (!HasRecords(path)) {
        cout << "File is empty. No records to search." << endl;
        return;
    }

    while (fread(&a, sizeof(Client), 1, pfile) == 1)
    {
        if (a.loanBody > max)
        {
            max = a.loanBody;
        }
        if (a.loanBody < min)
        {
            min = a.loanBody;
        }
    }
    cout << "The minimum credit body: " << min << endl;
    cout << "The maximum credit body: " << max << endl;
}

```

```

        fclose(pfile);
    }

time_t convertToTimeT(const char date[]) {
    tm tm = {};
    // Parse date in "dd.mm.yyyy" format
    sscanf(date, "%2d.%2d.%4d", &tm.tm_mday, &tm.tm_mon, &tm.tm_year);

    // Adjust month and year for tm (months are 0-11, years since 1900)
    tm.tm_mon -= 1;          // Months are 0-11 in tm
    tm.tm_year -= 1900;      // Years since 1900 in tm

    return mktime(&tm); // Convert to time_t for easy comparison
}

bool compareDates(const char date1[], const char date2[]) {
    return convertToTimeT(date1) < convertToTimeT(date2);
}

bool isDateBetween(const char targetDate[], const char startDate[], const char
endDate[]) {
    time_t target = convertToTimeT(targetDate);
    time_t start = convertToTimeT(startDate);
    time_t end = convertToTimeT(endDate);

    return (target >= start && target <= end);
}

void Third_CreditsByDate(const string& path) {
    FILE* pfile;
    Client a;
    size_t i = 0;
    int count = 0;
    char firstDate[11], secondDate[11];

    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }

    if (!HasRecords(path)) {
        cout << "File is empty. No records to search." << endl;
        fclose(pfile);
        return;
    }

    cout << "Enter the \"first date\" (DD.MM.YYYY): ";
    cin.getline(firstDate, sizeof(firstDate));

    cout << "Enter the \"second date\" (DD.MM.YYYY): ";
    cin.getline(secondDate, sizeof(secondDate));

    // Swap dates if needed
    if (!compareDates(firstDate, secondDate)) {
        swap(firstDate, secondDate);
    }

    // Output header if records exist
    cout << fixed << setprecision(2);
    while (fread(&a, sizeof(Client), 1, pfile) == 1) {
        if (isDateBetween(a.loanDate, firstDate, secondDate)) {
            if (count++ == 0) {

```



```

        cout << "X-----"
-----
X" << endl;
        cout << "| № | City | FullName | Phone Number
| Sum of credit | Total payable | Monthly Payment | First pay | loanDate
|" << endl;
        cout << "X-----"
-----
X" << endl;
    }
    cout << "| " << setw(3) << left << count << " | "
        << setw(10) << left << a.city << "| "
        << setw(19) << left << a.fullName << " | +38"
        << setw(11) << left << a.phoneNumber << " | "
        << setw(14) << right << a.loanBody << " | "
        << setw(15) << right << a.totalAmount << " | "
        << setw(17) << right << a.monthlyPayment << " | "
        << setw(3) << (a.isPaid ? "+" : "-") << " | "
        << setw(11) << right << a.loanDate << " | "
        << endl;
    }
}

fclose(pfile);

if (count > 0) {
    cout << "X-----"
-----X" <<
endl;
    cout << "There are " << count << " record(s)." << endl;
}
else {
    cout << "There are no records in this range :(" << endl;
}
}

void Forth_Debetors(const string& path)
{
    FILE* pfile;
    Client a;
    bool found = true;
    size_t count = 0;
    size_t i = 0;
    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }
    if (!HasRecords(path)) {
        cout << "File is empty. No records to search." << endl;
        return;
    }

    while (fread(&a, sizeof(Client), 1, pfile) == 1)
    {
        if (a.isPaid == false)
        {
            if (found)
            {
                cout << "X-----"
-----
X" << endl;

```

```

        cout << "| № | City | FullName | Phone Number
| Sum of credit | Total payable | Monthly Payment | First pay | loanDate
|" << endl;
        cout << "X-----"
-----
X" << endl;
        found = false;
    }
    cout << "| " << setw(3) << left << ++i << " | "
        << setw(10) << left << a.city << "| "
        << setw(19) << left << a.fullName << " | +38"
        << setw(11) << left << a.phoneNumber << " | "
        << setw(14) << right << fixed << setprecision(2) << a.loanBody <<
" | "
        << setw(15) << right << fixed << setprecision(2) << a.totalAmount
<< " | "
        << setw(17) << right << fixed << setprecision(2) <<
a.monthlyPayment << " | "
        << setw(3) << (a.isPaid ? "+" : "-") << " | "
        << setw(11) << right << fixed << setprecision(2) << a.loanDate <<
" | "
        << endl;
        count++;
    }
}
if (count == 1) {
    cout << "X-----"
-----X" <<
endl;
    cout << "There is " << count << " debetor." << endl;
}
else if (count > 1)
{
    cout << "X-----"
-----X" <<
endl;
    cout << "There are " << count << " debetors." << endl;
}
else
{
    cout << "There are no debetors :) " << endl;
}
}

void Fifth_TotalAmount(const string& path)
{
    FILE* pfile;
    Client a;
    float answer = 0;
    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }
    if (!HasRecords(path)) {
        cout << "File is empty. No records to search." << endl;
        return;
    }

    while (fread(&a, sizeof(Client), 1, pfile) == 1)
    {
        answer += a.totalAmount;
    }
}

```

```

    }

    cout << endl << "The bank's total profit (for the year) is " << answer <<
endl << endl;
}

void Requests() {
    string input;
    int choice;
    regex validChoicePattern("[0-5]$"); // Дозволяємо тільки цифри від 0 до 5

    while (true) {
        system("cls");
        cout << "\nMake a choice what to view:\n"
            << "0. Exit\n"
            << "1. Loans by client's name\n"
            << "2. MAX and MIN loan\n"
            << "3. Loans by a certain period\n"
            << "4. All debtors\n"
            << "5. Bank's total profit (for the year)\n"
            << "Choose an option: ";
        getline(cin, input);

        // Перевірка на коректність вводу
        if (regex_match(input, validChoicePattern)) {
            choice = stoi(input); // Перетворення у число після перевірки
            break;
        }
        else {
            cout << "Error: Please enter a number between 0 and 5.\n";
            system("pause");
        }
    }

    switch (choice) {
        case 0:
            return;
        case 1:
            FIRST_ByClientName(PATH);
            break;
        case 2:
            Second_MAX_MIN(PATH);
            break;
        case 3:
            Third_CreditsByDate(PATH);
            break;
        case 4:
            Forth_Debetors(PATH);
            break;
        case 5:
            Fifth_TotalAmount(PATH);
            break;
        default:
            cout << "Invalid choice!" << endl;
    }
}

int main() {
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);

    EnsureFileExists(PATH);

```

```

string input;
int choice;
regex integerRegex("[0-6]$"); // Дозволяємо лише числа від 0 до 6

while (true) {
    system("cls");
    cout << "\nMenu:\n"
        << "0. Exit\n"
        << "1. View Records\n"
        << "2. Add Record\n"
        << "3. Edit Record\n"
        << "4. Delete Record\n"
        << "5. Sort Records\n"
        << "6. Make a request\n"
        << "Choose an option: ";
    getline(cin, input);

    // Перевіряємо ввід
    if (!regex_match(input, integerRegex)) {
        cout << "Invalid input. Please enter a number between 0 and 6." <<
endl;
        system("pause");
        continue;
    }

    choice = stoi(input); // Перетворення на число

    // Обробка вибору
    switch (choice) {
    case 0:
        return 0;

    case 1:
        ViewClients(PATH);
        break;

    case 2: {
        Client client;
        cout << "Are you sure? (Y/ any other input): ";
        getline(cin, input);

        if (input == "Y" || input == "y") {
            InputClient(client, PATH);
            WriteClient(PATH, client);
        }
        else {
            cout << "Your choice was canceled :)" << endl;
        }
        break;
    }

    case 3:
        EditClient(PATH);
        break;

    case 4:
        if (!HasRecords(PATH)) {
            cout << "File is empty. No records to delete." << endl;
        }
        else {
            ViewClients(PATH);
            cout << "Are you sure? (Y/ any other input): ";
            getline(cin, input);
        }
    }
}

```

```

        if (input == "Y" || input == "y") {
            DeleteClient(PATH);
        }
        else {
            cout << "Your choice was canceled :)" << endl;
        }
    }
    break;

case 5:
    if (!HasRecords(PATH)) {
        cout << "File is empty." << endl;
    }
    else if (OnlyOneRecord(PATH) < 2) {
        cout << "There are not enough records in the file to sort!" <<
endl;
    }
    else {
        cout << "Are you sure? (Y/ any other input): ";
        getline(cin, input);

        if (input == "Y" || input == "y") {
            SortRecords();
        }
        else {
            cout << "Your choice was canceled :)" << endl;
        }
    }
    break;

case 6:
    Requests();
    break;

default:
    cout << "Invalid input." << endl;
}

system("pause");
}
}

```

Лістинг Б.2 – Код функції WriteClient

```

void WriteClient(const string& path, const Client& client) {
    ofstream file(path, ios::binary | ios::app);
    if (!file) { // Перевірка, чи файл відкрився успішно
        cerr << "Error: Could not open file for writing!" << endl;
        return;
    }
    file.write((const char*)&client, sizeof(Client));
    file.close();
}

```

Лістинг Б.3 – Код функції InputLoanBody

```

void InputLoanBody(float& loanBody) {
    regex validAmountPattern("^\\d+(\\.\\d+)?$"); // Регулярний вираз для числа
    (ціле або з плаваючою комою)
}

```

```

string input;

while (true) {
    cout << "Enter the \"Sum of credit\" (from 1,000 to 1,000,000) - ";
    getline(cin, input);

    // Перевірка на коректність введеного числа
    if (!regex_match(input, validAmountPattern)) {
        cout << "Error: You must enter a valid number!" << endl;
        continue;
    }

    // Конвертуємо введений рядок в число
    loanBody = stof(input);

    // Перевірка діапазону
    if (loanBody >= 1000 && loanBody <= 1000000) {
        break; // Якщо число в межах діапазону, виходимо з циклу
    }
    else {
        cout << "Error: Amount is not in the acceptable range!" << endl;
    }
}
}

```

Лістинг Б.4 – Код функції formatCityName

```

string formatCityName() {
    const size_t maxCityLength = 16; // Максимальна довжина назви міста
    string result;

    while (true) {
        getline(cin, result);

        // Перевірка довжини
        if (!isValidLength(result, maxCityLength)) {
            cout << "Error: The city name cannot exceed "
                 << maxCityLength << " characters. Try again: ";
            continue; // Повертаємось до повторного вводу
        }

        // Форматування: кожне слово починається з великої літери
        bool capitalize = true;
        bool isValid = true; // Змінна для відслідковування валідності вводу
        for (size_t i = 0; i < result.length(); ++i) {
            if (isalpha(result[i])) {
                result[i] = capitalize ? toupper(result[i]) : tolower(result[i]);
                capitalize = false;
            }
            else if (result[i] == ' ' || result[i] == '-') {
                capitalize = true;
            }
            else {
                cout << "Error: The city name cannot have numbers or special
symbols. Try again: ";
                isValid = false; // Встановлюємо прапорець невдалого вводу
                break; // Виходимо з for, щоб перевірити isValid
            }
        }
        // Якщо ввід некоректний, повторюємо введення
        if (!isValid) {
            continue;
        }
    }
}

```



```

    }
}

```

Лістинг Б.7 – Код функції InputLoanDate

```

void InputLoanDate(Client& client) {
    do {
        while (true) {
            cout << "Please enter the date in DD.MM.YYYY format: ";

            string input; // Використовуємо std::string для зберігання вводу
            getline(cin, input); // Читання всього рядка

            // Перевірка на довжину введенного рядка
            if (input.length() > 10) {
                cout << "Error: Input exceeds maximum allowed length (10
characters). Try again.\n";
                continue; // Повторний запит на введення
            }

            // Копіювання в масив char для збереження у loanDate
            strncpy(client.loanDate, input.c_str(), 10);
            client.loanDate[10] = '\0'; // Забезпечуємо коректний термінатор

            break; // Вихід з циклу, якщо ввід правильний
        }
        if (IsValidDate(client.loanDate)) {
            return;
        }
    } while (true);
}

```

Лістинг Б.8 – Код функції ValidatePhoneNumber

```

void ValidatePhoneNumber(Client& client)
{
    regex phonePattern("[0-9]{10}$"); // Шаблон: рівно 10 цифр
    string inputPhone;

    // Цикл для введення номера телефону
    while (true) {
        cout << "Enter the \"Phone number\" (10 digits): +38";
        getline(cin, inputPhone);

        // Перевірка введенного номеру за шаблоном
        if (!regex_match(inputPhone, phonePattern)) {
            cout << "Error: Phone number must consist of exactly 10
digits." << endl;
            continue;
        }

        // Запис номеру в client.phoneNumber
        strncpy(client.phoneNumber, inputPhone.c_str(),
sizeof(client.phoneNumber) - 1);
    }
}

```



```

        client.phoneNumber[sizeof(client.phoneNumber) - 1] = '\\0';
// Гарантуємо, що рядок завершений
        break;
    }
}

```

Лістинг Б.9 – Код функції ViewClients

```

void ViewClients(const string& path) {
    if (!HasRecords(path)) {
        cout << "File is empty." << endl;
        return;
    }

    ifstream file(path, ios::binary);
    if (!file) {
        cout << "Error in file's opening" << endl;
        return;
    }

    cout << "x-----X"
    << endl;
    cout << "| № | City | FullName | Phone Number |
Sum of credit | Total payable | Monthly Payment | First pay | loanDate |"
    << endl;
    cout << "x-----X"
    << endl;

    Client current;
    size_t i = 0;

    while (file.read((char*)&current), sizeof(Client)) {

        cout << "| " << setw(3) << left << ++i << " | "
            << setw(17) << left << current.city << "| "
            << setw(20) << left << current.fullName << " | +38"
            << setw(11) << left << current.phoneNumber << " | "
            << setw(14) << right << fixed << setprecision(2) << current.loanBody
        << " | "
            << setw(15) << right << fixed << setprecision(2) <<
current.totalAmount << " | "
            << setw(17) << right << fixed << setprecision(2) <<
current.monthlyPayment << " | "
            << setw(3) << (current.isPaid ? "+" : "-") << " | "
            << setw(11) << right << fixed << setprecision(2) << current.loanDate
        << " | "
            << endl;
    }

    cout << "x-----X"
    << endl;
    file.close();
}

```

Лістинг Б.10 – Код функції EditClientField

```

void EditClientField(Client& client) {
    string choiceInput;

```

```

cout << "Select a field to edit: " << endl;
cout << "1. City" << endl;
cout << "2. Full Name" << endl;
cout << "3. Phone Number" << endl;
cout << "4. Sum of credit" << endl;
cout << "5. Loan Date" << endl;
cout << "6. First pay" << endl;
cout << "Select an option: ";

regex validChoicePattern("^[1-6]$"); // Регулярний вираз для вибору числа
від 1 до 6

while (true) {
    getline(cin, choiceInput); // Отримуємо введення як рядок

    // Перевірка, чи введення відповідає патерну
    if (regex_match(choiceInput, validChoicePattern)) {
        int choice = stoi(choiceInput); // Перетворюємо введенне значення в
ціле число

        switch (choice) {
            case 1:
                SityInput(client);
                return;
            case 2:
                ValidateFullName(client.fullName);
                return;
            case 3:
                ValidatePhoneNumber(client);
                return;
            case 4:
                InputLoanBody(client.loanBody);
                client.totalAmount = client.loanBody * 1.1;
                client.monthlyPayment = client.totalAmount / 12;
                return;
            case 5:
                InputLoanDate(client);
                return;
            case 6:
                InputIsPaid(client);
                return;
            default:
                cout << "Invalid choice. Try again: ";
        }
    }
    else {
        cout << "Invalid choice. Try again (1-6): ";
    }
}
}

```

Лістинг Б.11 – Код функції EditClient

```

void EditClient(const string& path) {
    if (!HasRecords(path)) {
        cout << "File is empty. No records to edit." << endl;
        return;
    }

    ViewClients(path); // Displaying the list of clients before editing

    string input;
    cout << "Are you sure? (Y/any other input): ";
}

```

```

getline(cin, input);
if (input != "Y" && input != "y") {
    cout << "Your choice was canceled :)" << endl;
    return;
}
string recordNumberInput;
int recordNumber;

fstream file(path, ios::binary | ios::in | ios::out);
if (!file) {
    cout << "Error: Cannot open file." << endl;
    return;
}
Client client;
while (true) {
    try {
        cout << "Enter the record number to edit: ";
        getline(cin, recordNumberInput);
        recordNumber = stoi(recordNumberInput);
        file.seekg(0, ios::end);
        size_t numRecords = file.tellg() / sizeof(Client);
        if (recordNumber < 1 || recordNumber > static_cast<int>(numRecords))
        {
            cout << "Error: Invalid record number." << endl;
            continue;
        }
        break;
    }
    catch (const invalid_argument&) {
        cout << "Error: Invalid input. Please enter a valid number." << endl;
    }
    catch (const out_of_range&) {
        cout << "Error: Number is out of range." << endl;
    }
}
// Read the specified record
file.seekg((recordNumber - 1) * sizeof(Client), ios::beg);
file.read(reinterpret_cast<char*>(&client), sizeof(Client));
// Allow the user to edit the record
int choice;
cout << "\nEdit:\n1. Full record\n2. Particular field\nYour choice: ";

string choiceInput;
regex validChoicePattern("^[12]$");
while (true) {
    getline(cin, choiceInput);
    if (regex_match(choiceInput, validChoicePattern)) {
        choice = stoi(choiceInput);
        break;
    }
    else {
        cout << "Wrong choice. Try again: ";
    }
}
if (choice == 1) {
    EditClientRecord(client);
}
else if (choice == 2) {
    EditClientField(client);
}
// Write the modified record back to the file
file.seekp((recordNumber - 1) * sizeof(Client), ios::beg);
file.write(reinterpret_cast<const char*>(&client), sizeof(Client));

```

```

    file.close();
}

```

Лістинг Б.12 – Код функції DeleteRecordByName

```

void DeleteRecordByName() {
    FILE* pfile;
    FILE* tempFile;
    Client a;
    bool found = false; // To track if a record is found for deletion

    char fullNameToDelete[30];
    string inputName;
    regex fullNamePattern("[A-Z]{1}[a-z]{1,20} [A-Z]{1}\\.[ ]{1}[A-Z]{1}\\.");
    // Формат "Surname I. P."

    while (true) {
        cout << "Enter the full name to delete (format: 'Surname I. P.'): ";
        getline(cin, inputName);

        // Видалення зайвих пробілів
        inputName = regex_replace(inputName, regex("^ +| +$|( ) +"), "$1");

        // Перевірка введення
        if (regex_match(inputName, fullNamePattern)) {
            break;
        }
        else {
            cout << "Error: Full name must be entered in the format 'Surname I. P.' (Surname max: 20)\n";
        }
    }

    strncpy(fullNameToDelete, inputName.c_str(), 30);

    fopen_s(&pfile, PATH, "rb");
    if (!pfile) {
        cerr << "Failed to open the file!" << endl;
        return;
    }

    fopen_s(&tempFile, "temp.dat", "wb");
    if (!tempFile) {
        cerr << "Failed to create a temporary file!" << endl;
        fclose(pfile);
        return;
    }

    // Read each record and copy it to the temp file if it doesn't match the name to delete
    while (fread(&a, sizeof(Client), 1, pfile) == 1) {
        if (strcmp(a.fullName, fullNameToDelete) != 0) {
            fwrite(&a, sizeof(Client), 1, tempFile);
        }
        else {
            found = true; // Mark that we found and skipped a record
        }
    }

    fclose(pfile);
    fclose(tempFile);

    // Replace the original file with the temp file

```

```
remove(PATH);
rename("temp.dat", PATH);

if (found) {
    cout << "Client with name \"" << fullNameToDelete << "\" deleted
successfully." << endl;
}
else {
    cout << "No records found with the name \"" << fullNameToDelete << "\" for
deletion." << endl;
}
}
```

ДОДАТОК В

СТРУКТУРНІ СХЕМИ АЛГОРИТМУ РІШЕННЯ ЗАДАЧІ

Структурні схеми алгоритму рішення задачі зображені на рисунках В.1-В.35.

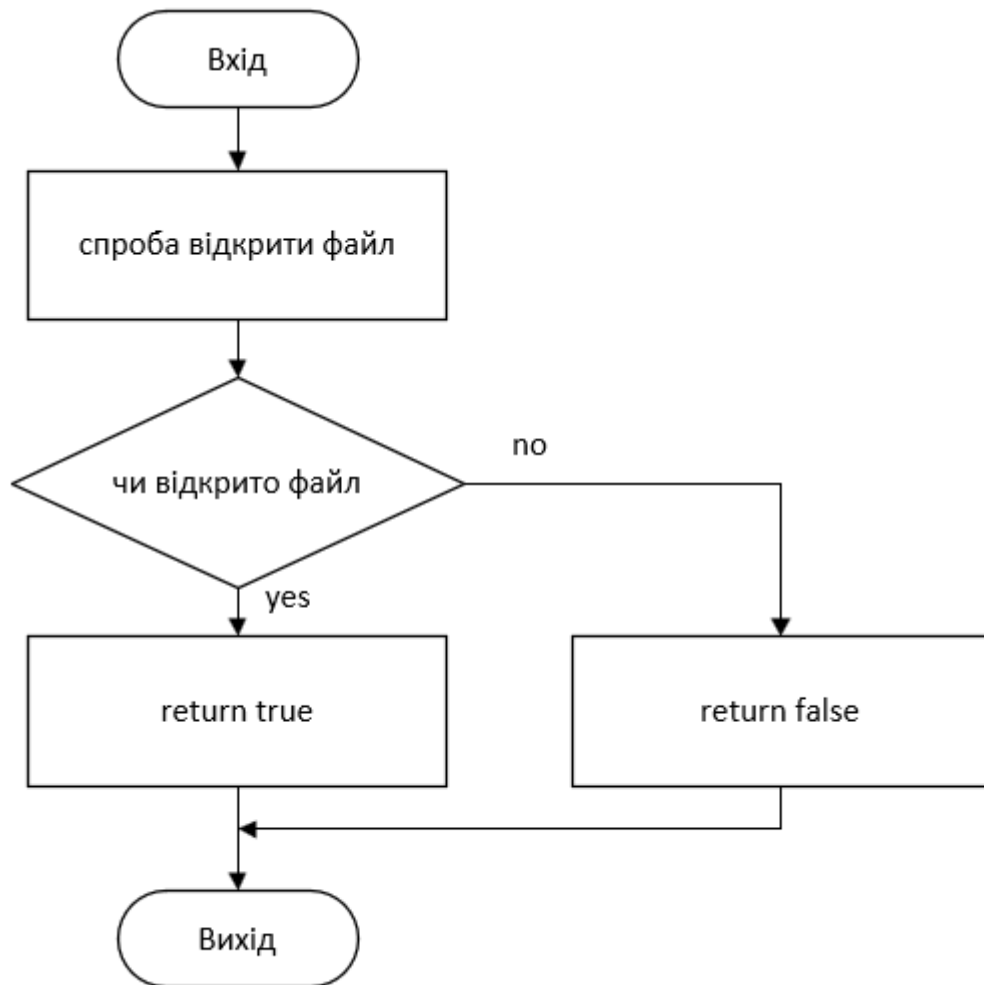


Рисунок В.1 – Структурна схема функції `bool FileExists(const string& path)`

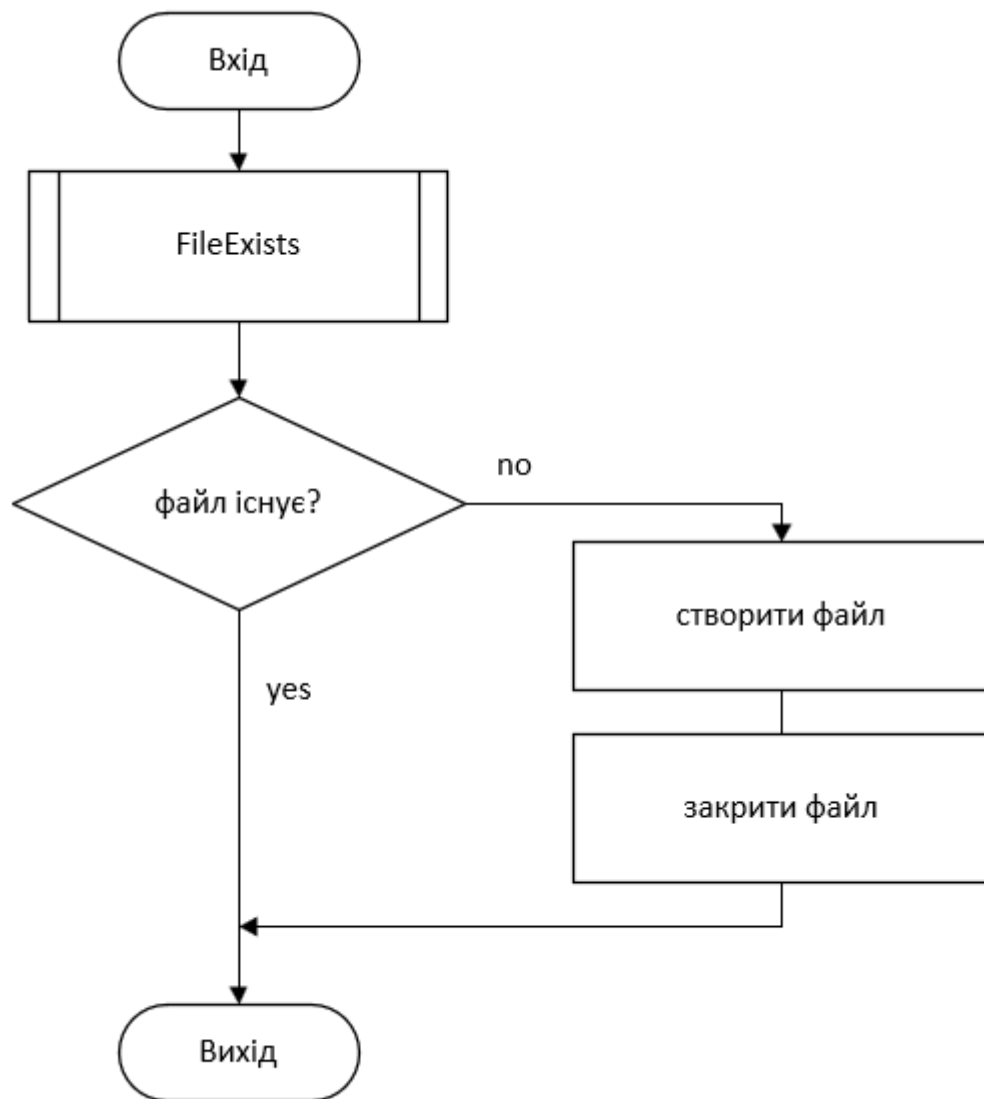


Рисунок В.2 – Структурна схема функції `void EnsureFileExists(const string& path)`

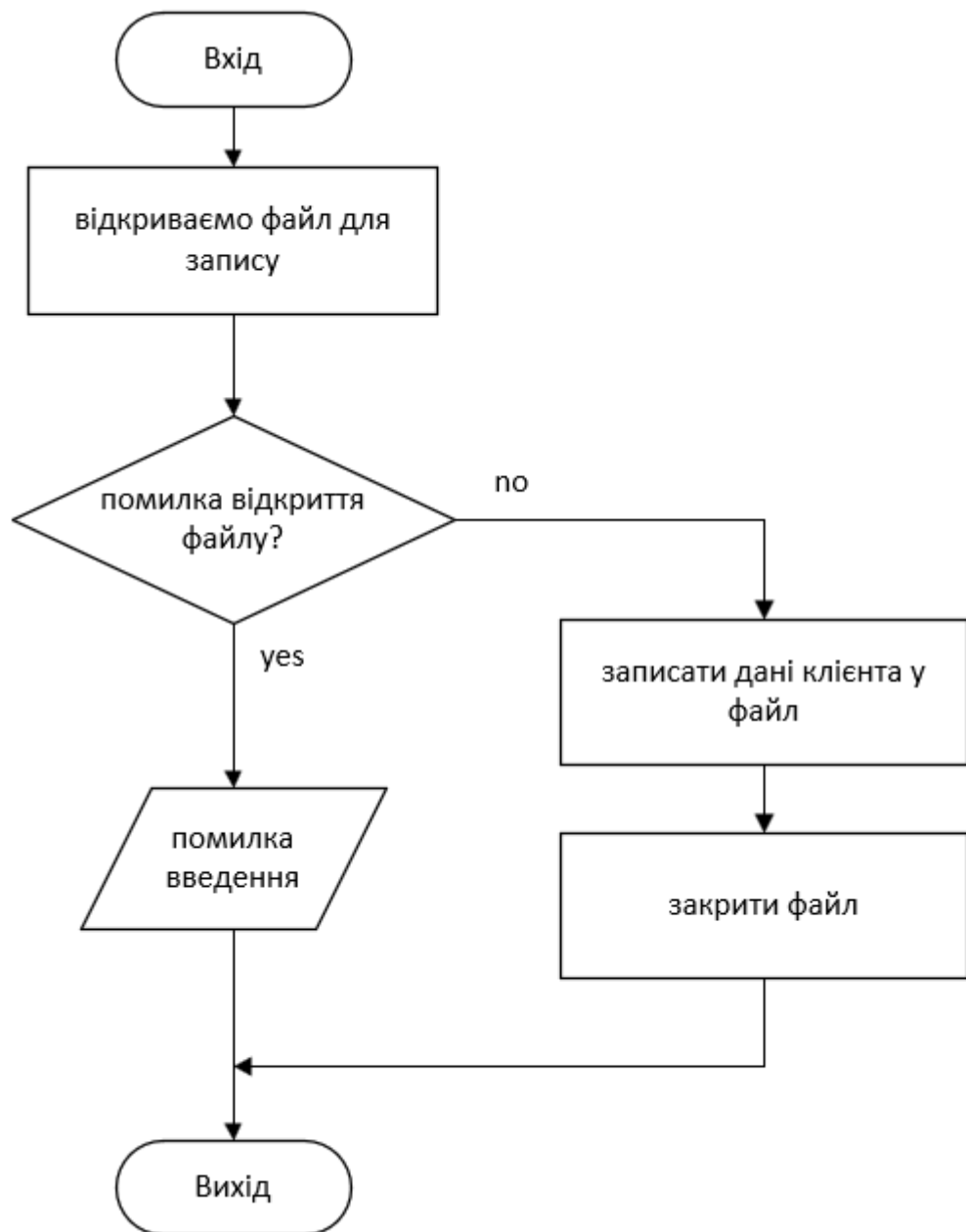


Рисунок В.3 – Структурна схема функції `void WriteClient(const string& path, const Client& client)`

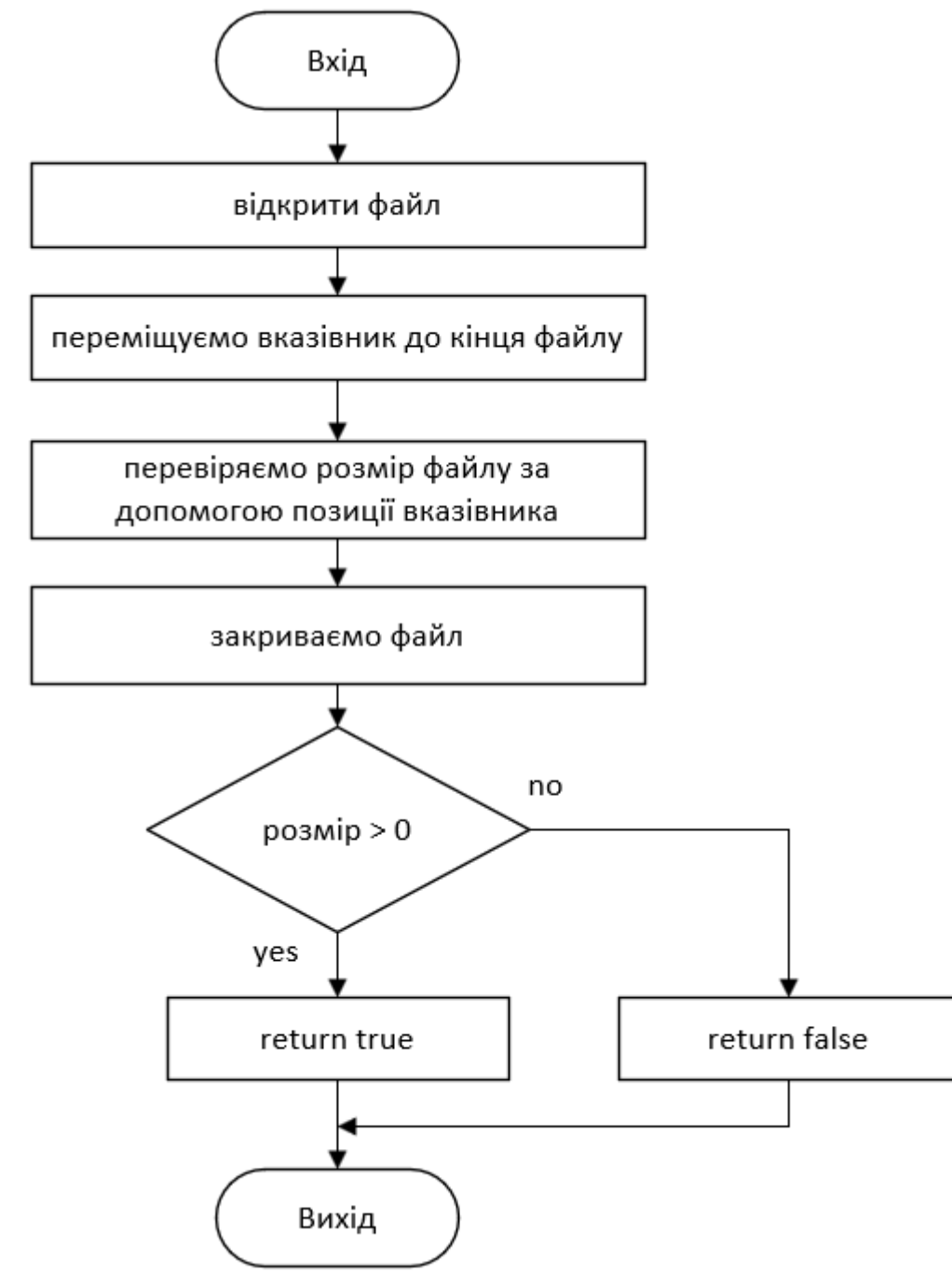


Рисунок В.4 – Структурна схема функції `bool HasRecords(const string& path)`

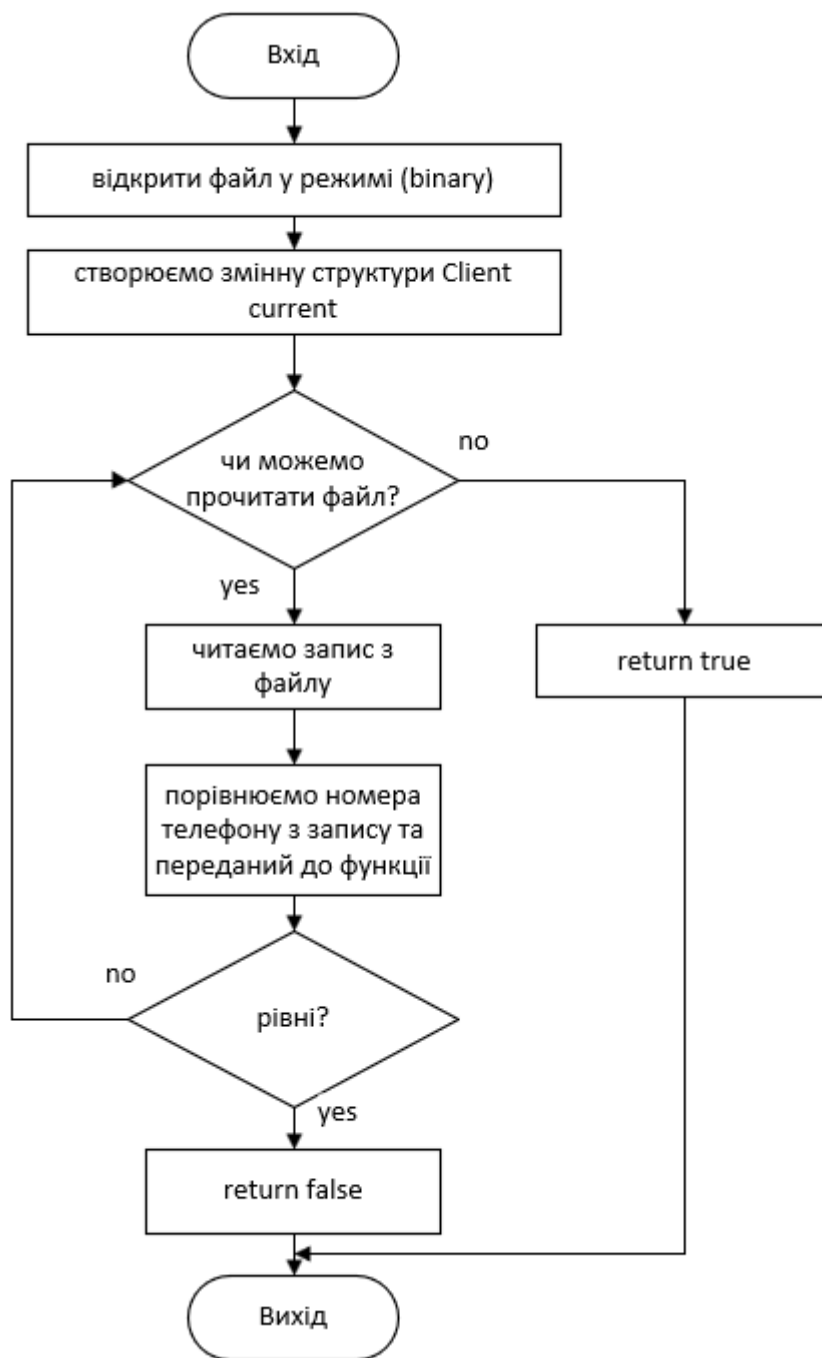


Рисунок В.5 – Структурна схема функції `bool IsUniquePhoneNumber(const string& path, const char* phoneNumber)`

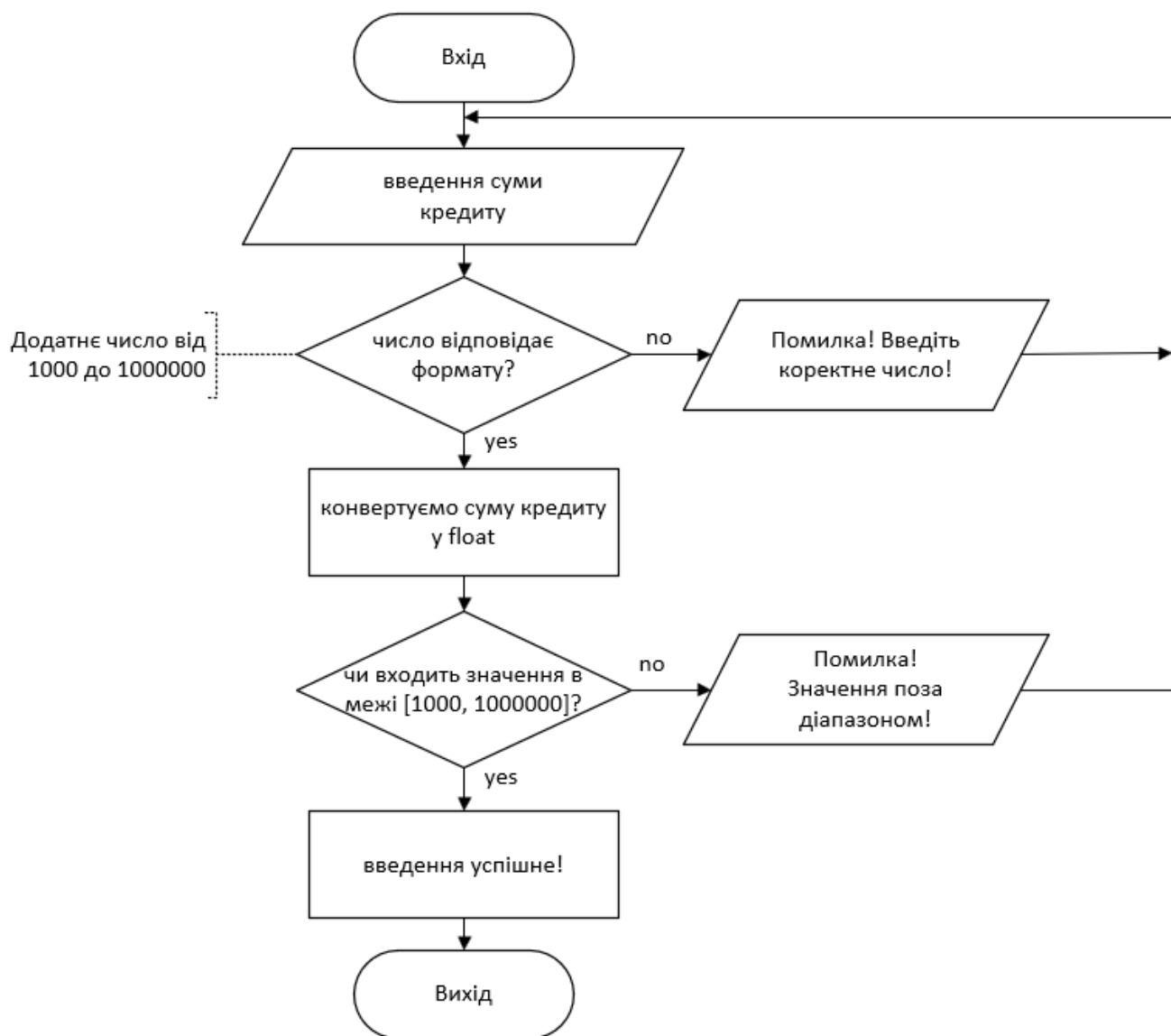


Рисунок В.6 – Структурна схема функції void InputLoanBody(float& loanBody)

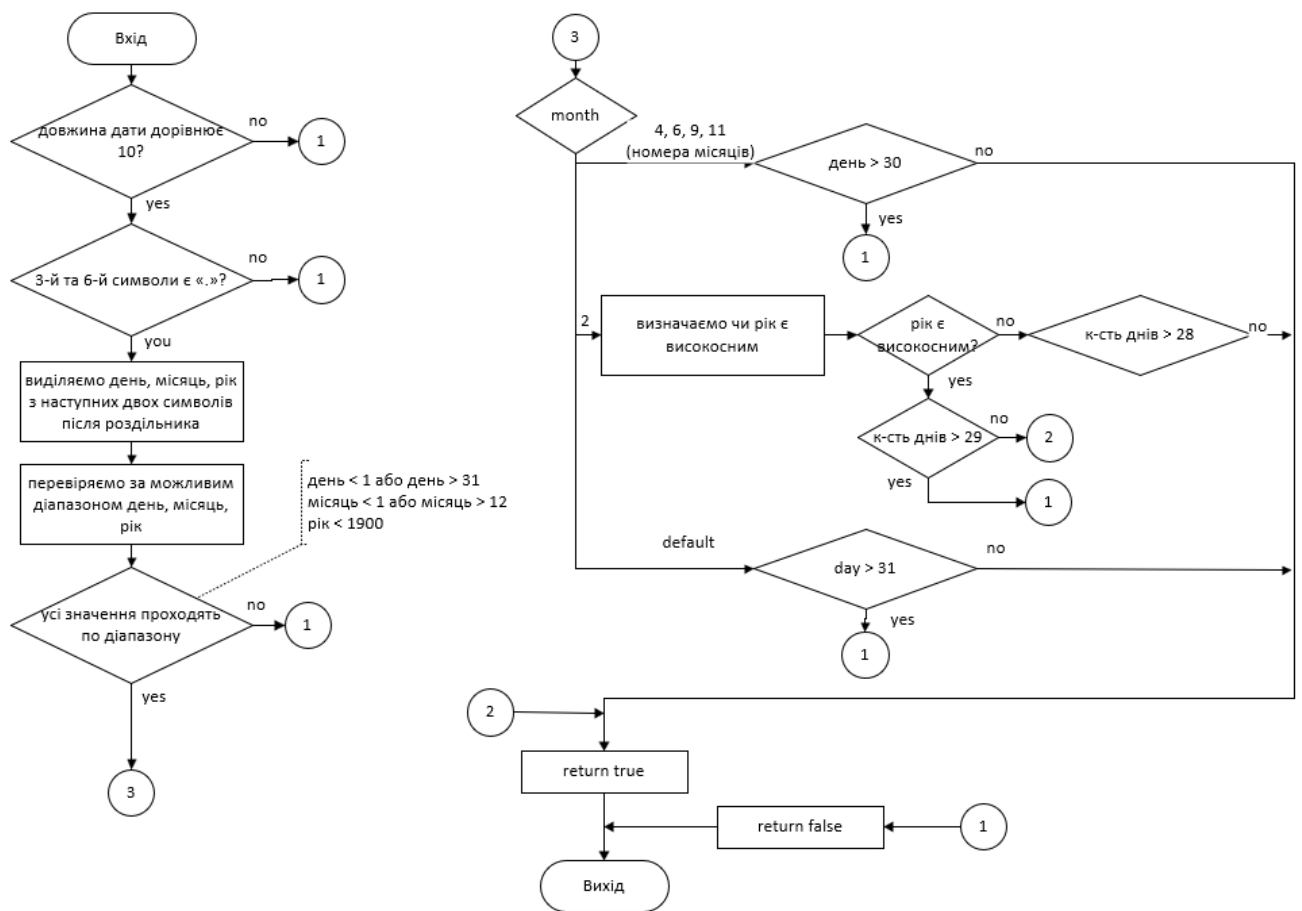


Рисунок В.7 – Структурна схема функції bool IsValidDate(const char date[])

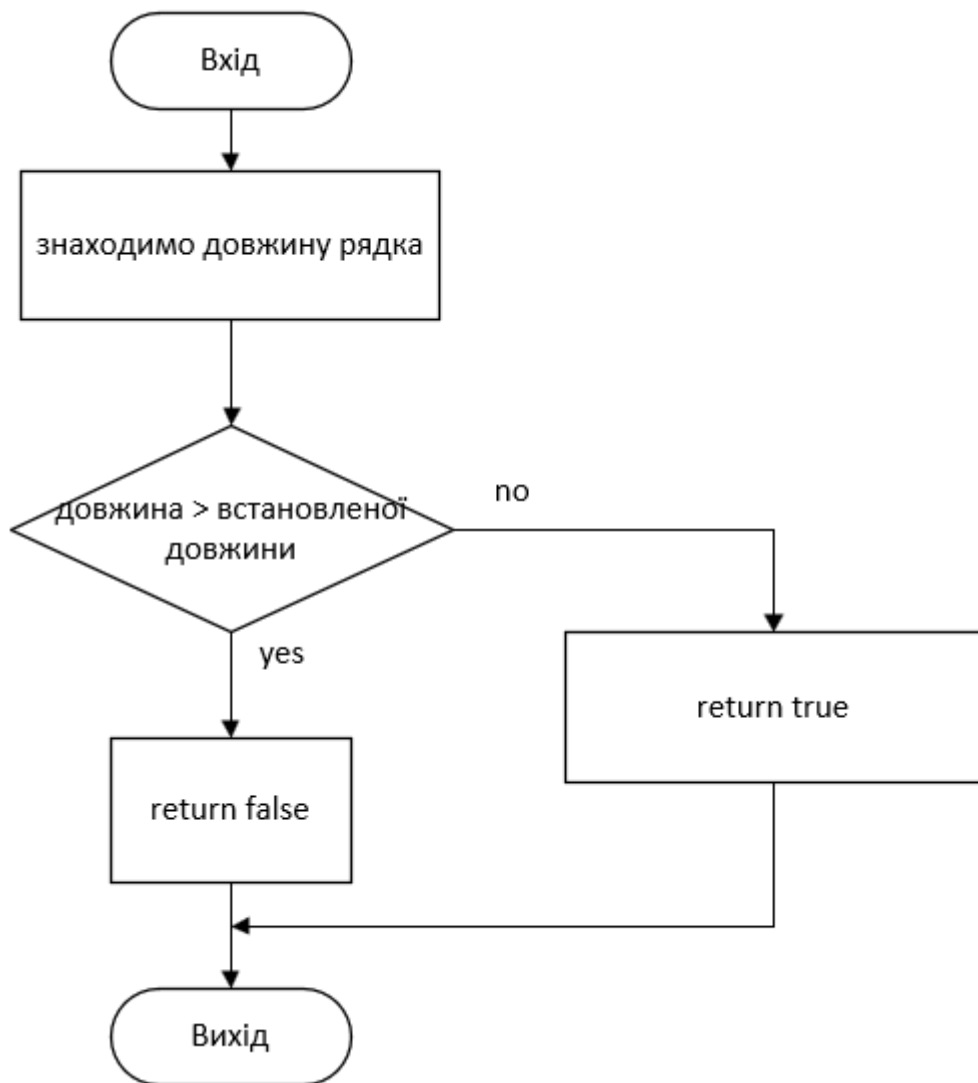


Рисунок В.8 – Структурна схема функції `bool isValidLength(const string& str, size_t maxLength)`

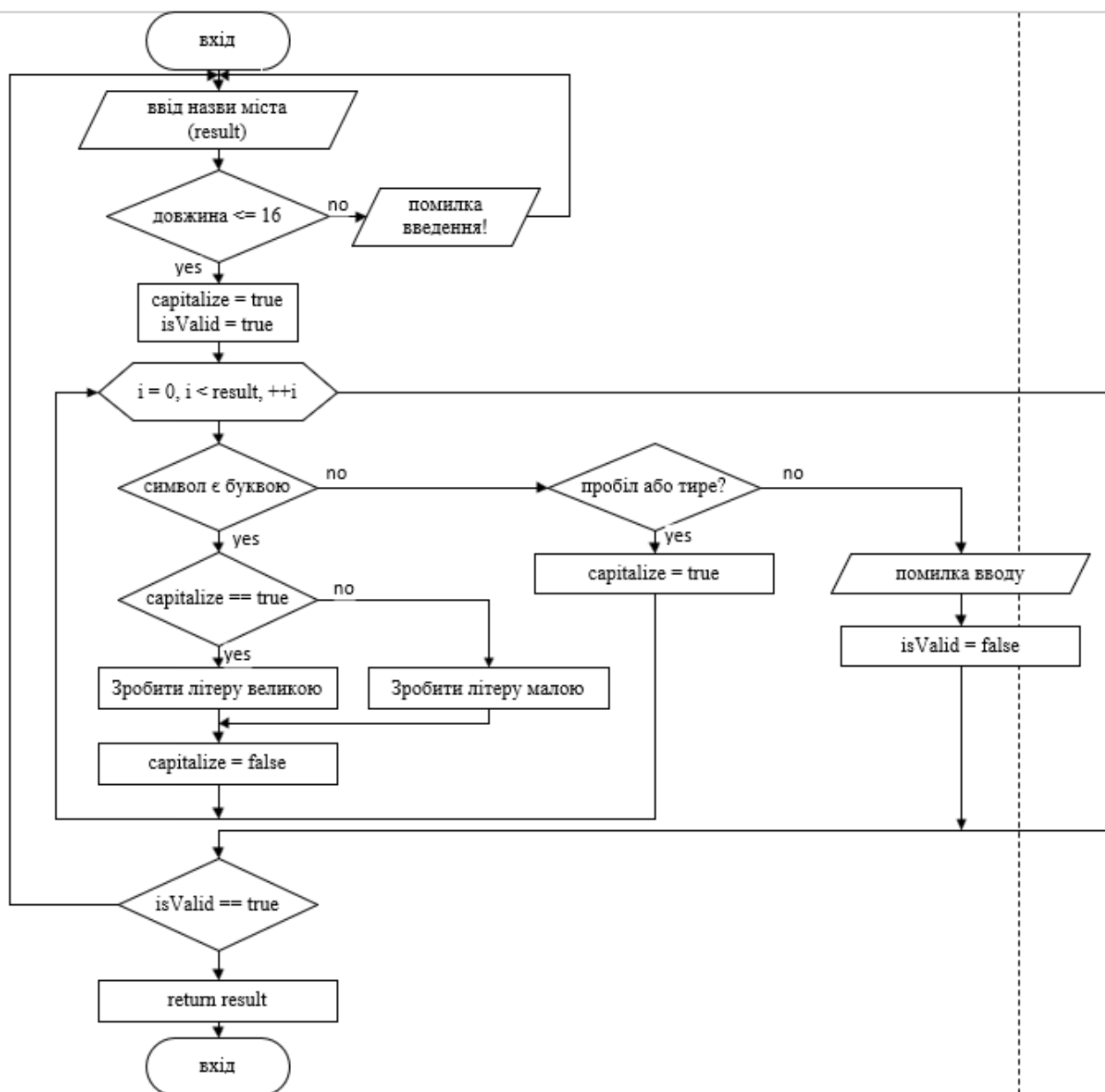


Рисунок В.9 – Структурна схема функції string formatCityName()

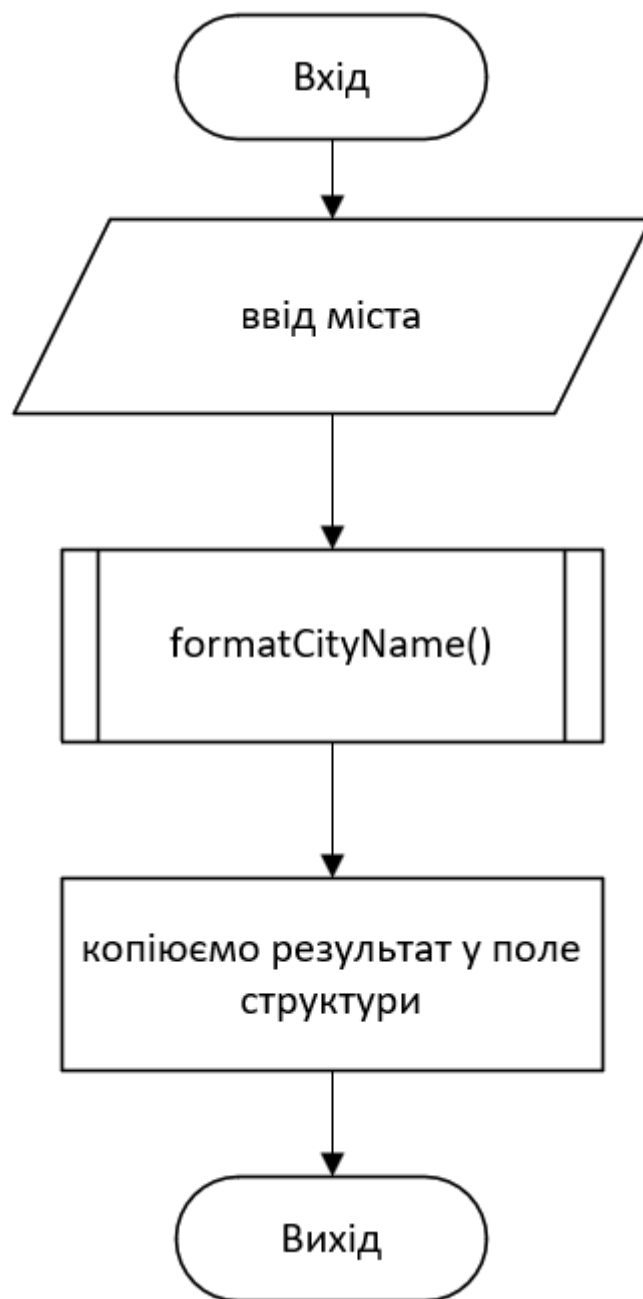


Рисунок В.10 – Структурна схема функції void SityInput(Client& client)

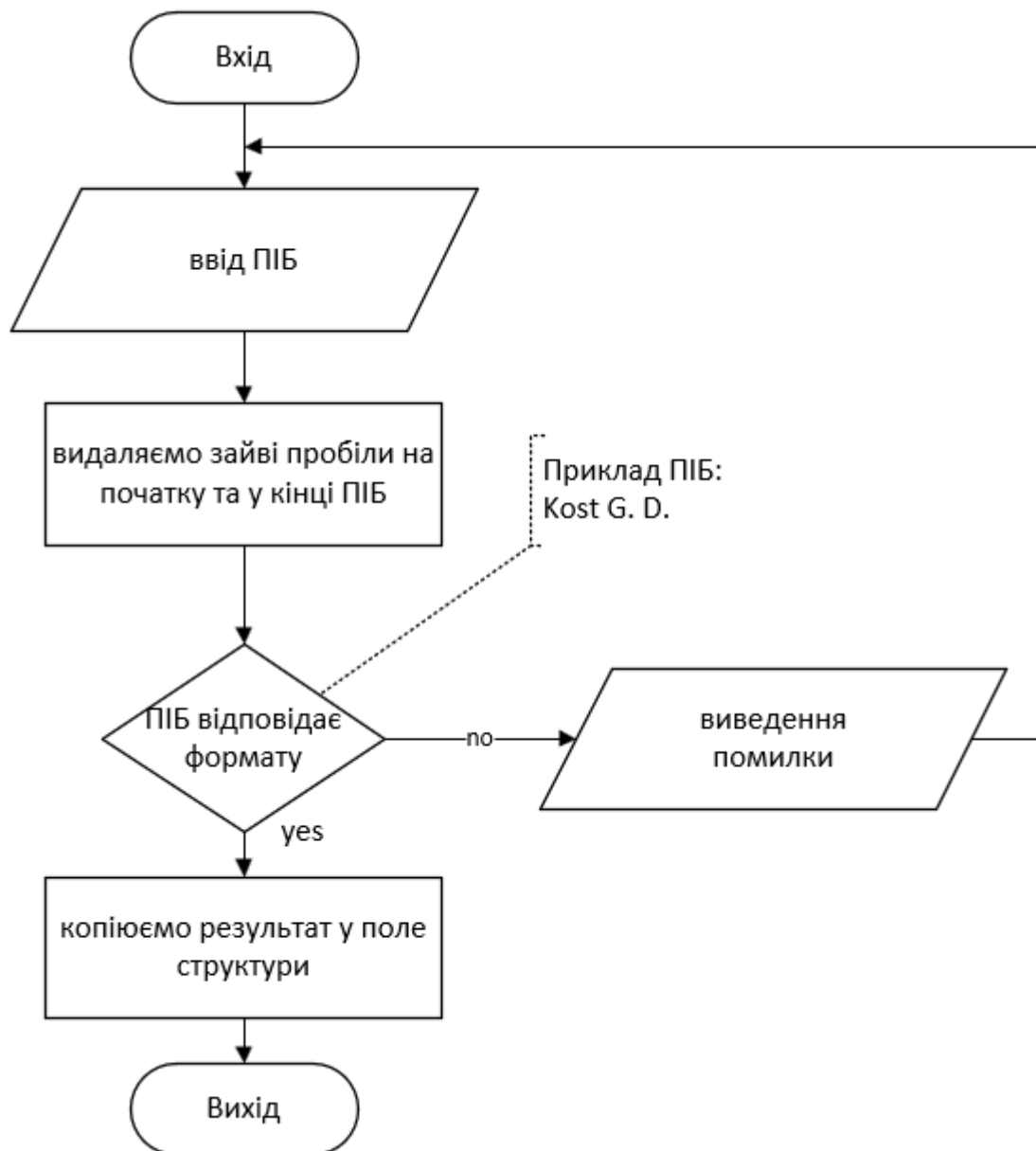


Рисунок В.11 – Структурна схема функції void ValidateFullName(char fullName[30])

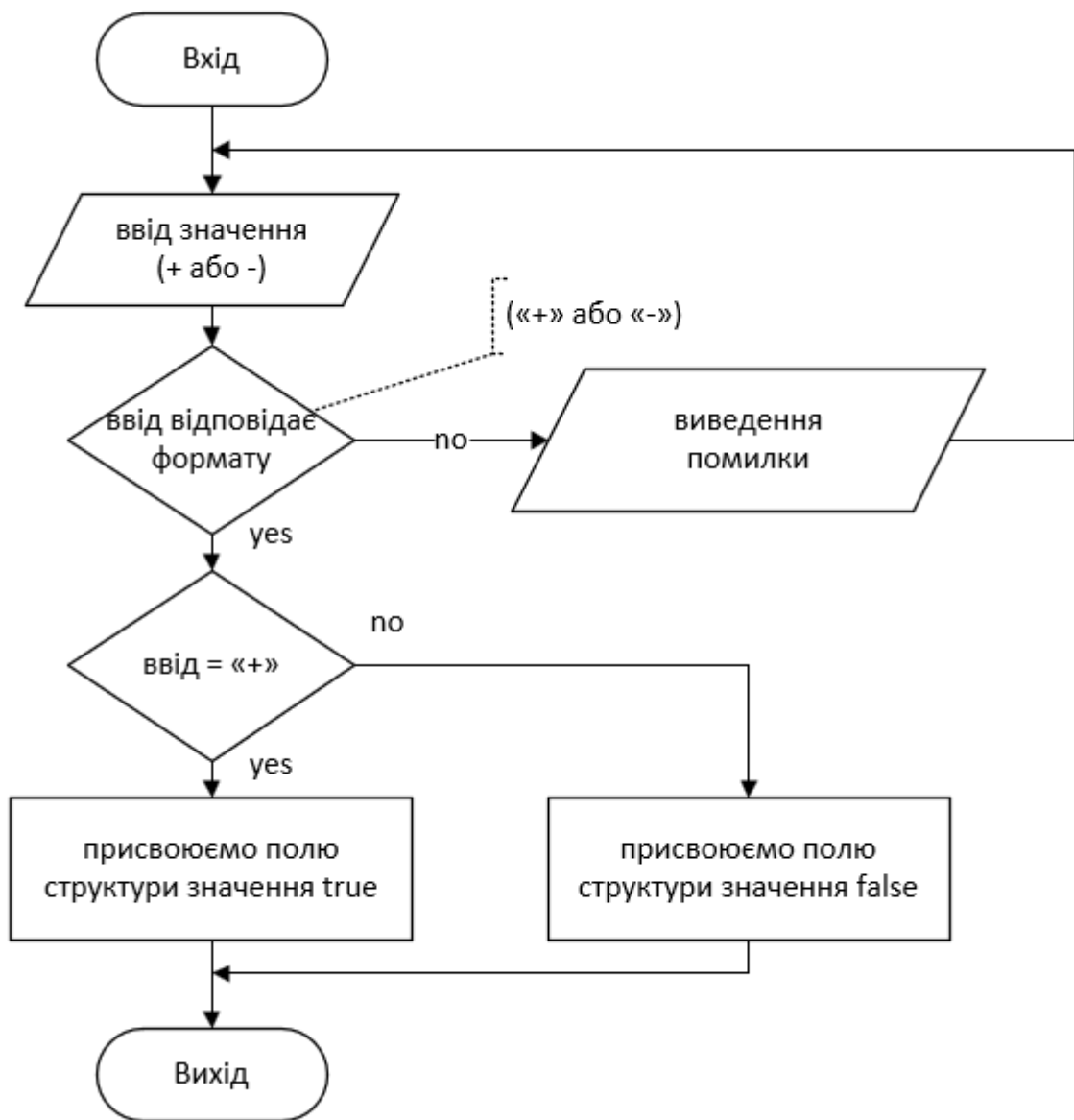


Рисунок В.12 – Структурна схема функції `void InputIsPaid(Client& client)`

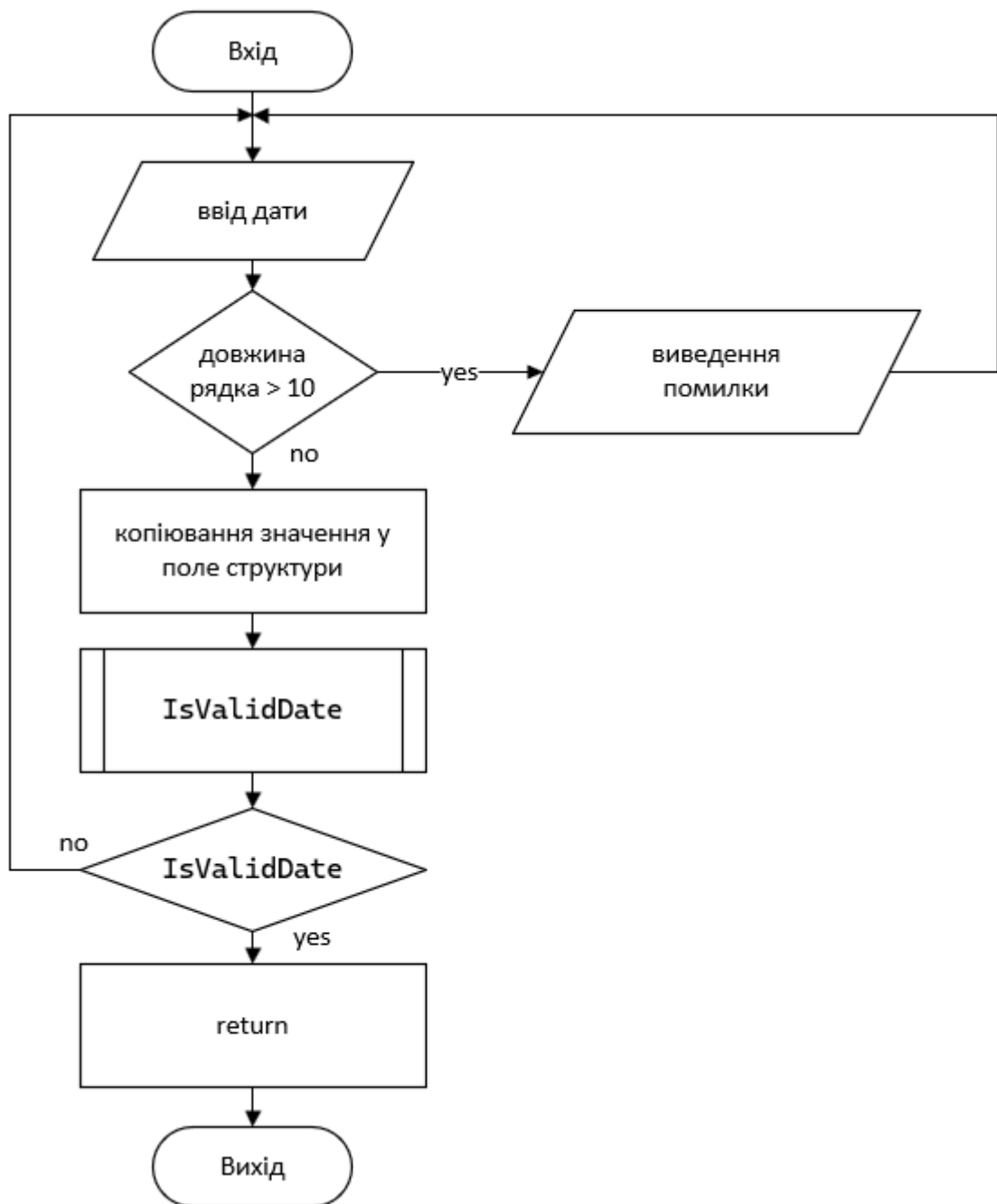


Рисунок В.13 – Структурна схема функції void InputLoanDate(Client& client)

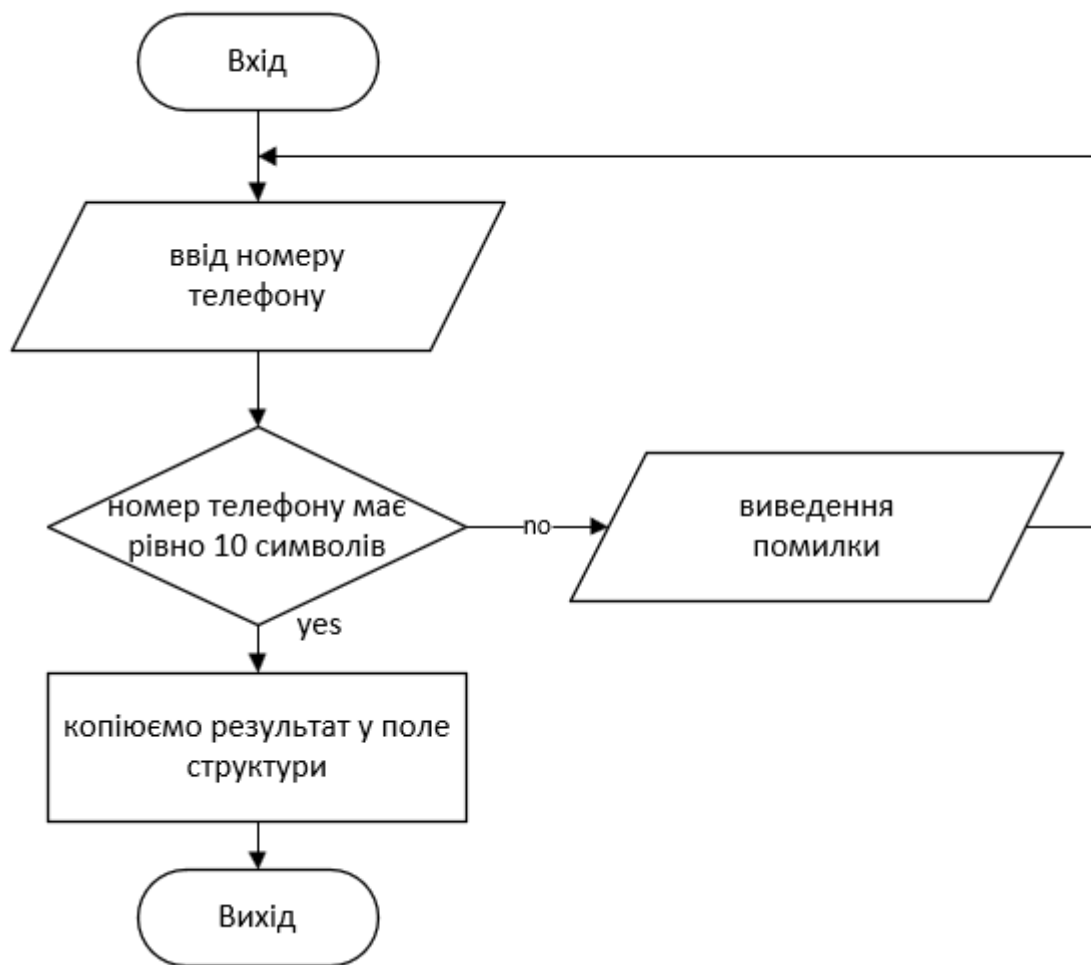


Рисунок В.14 – Структурна схема функції `void ValidatePhoneNumber(Client& client)`

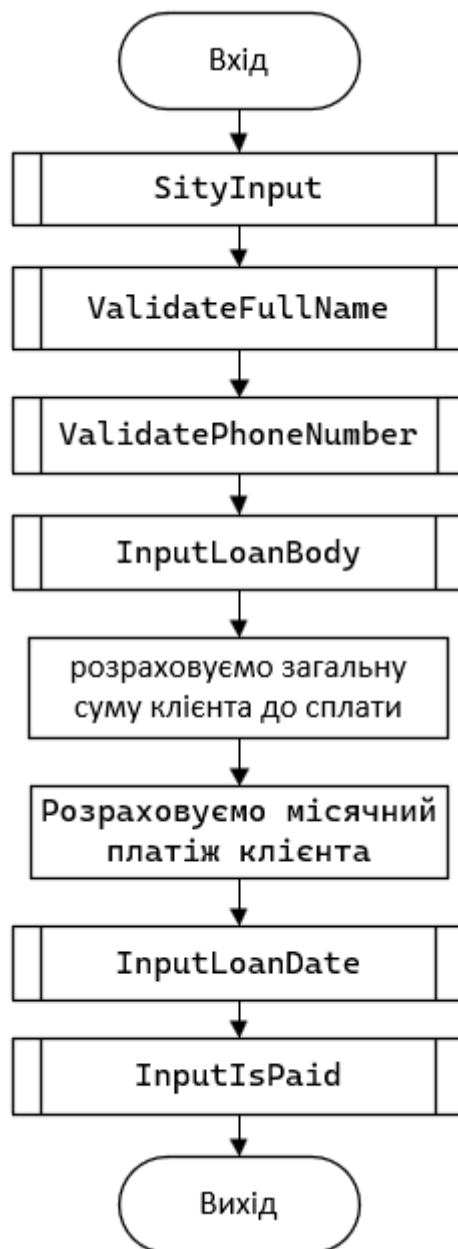


Рисунок В.15 – Структурна схема функції `void InputClient(Client& client, const string& path)`

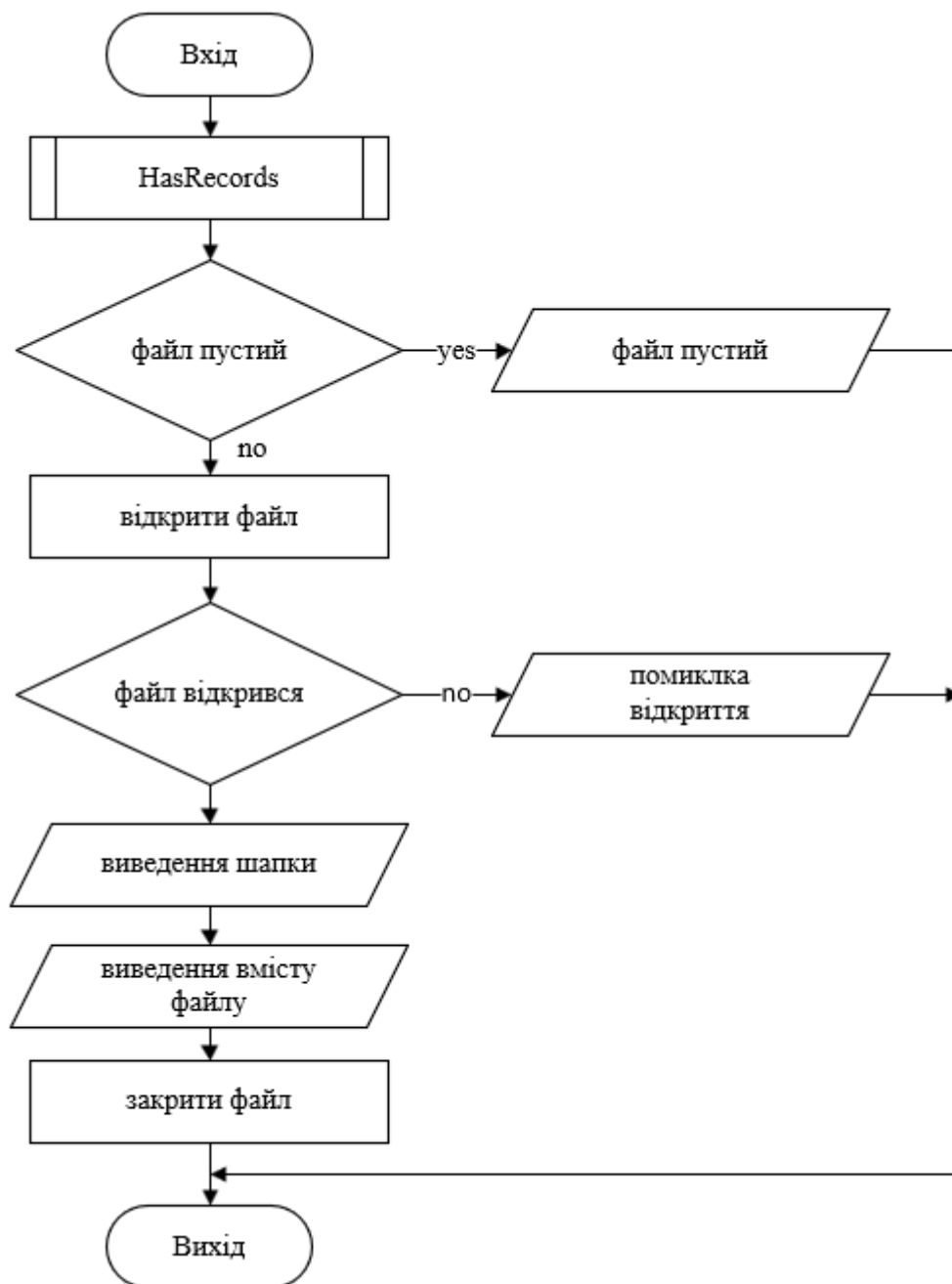


Рисунок В.16 – Структурна схема функції `void ViewClients(const string& path)`

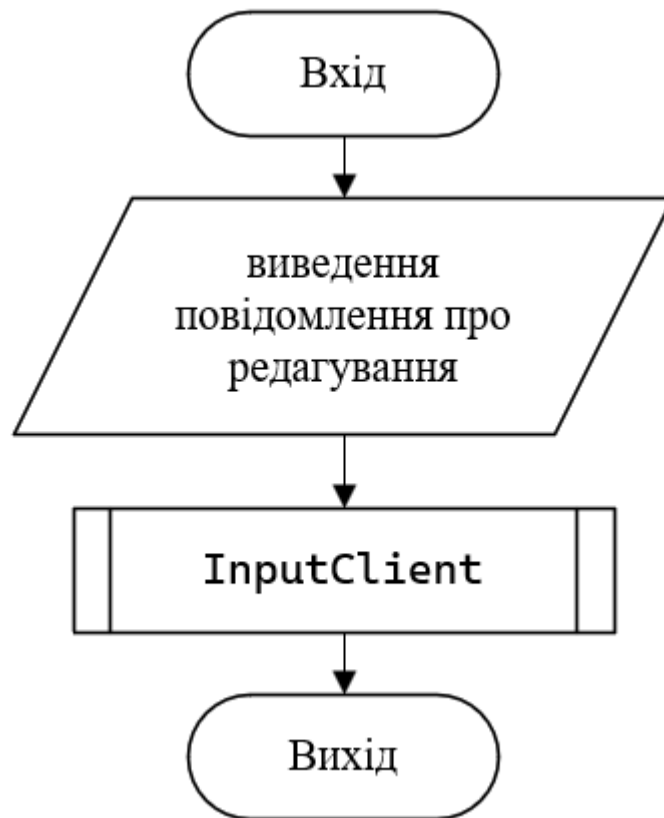


Рисунок В.17 – Структурна схема функції `void EditClientRecord(Client& client)`

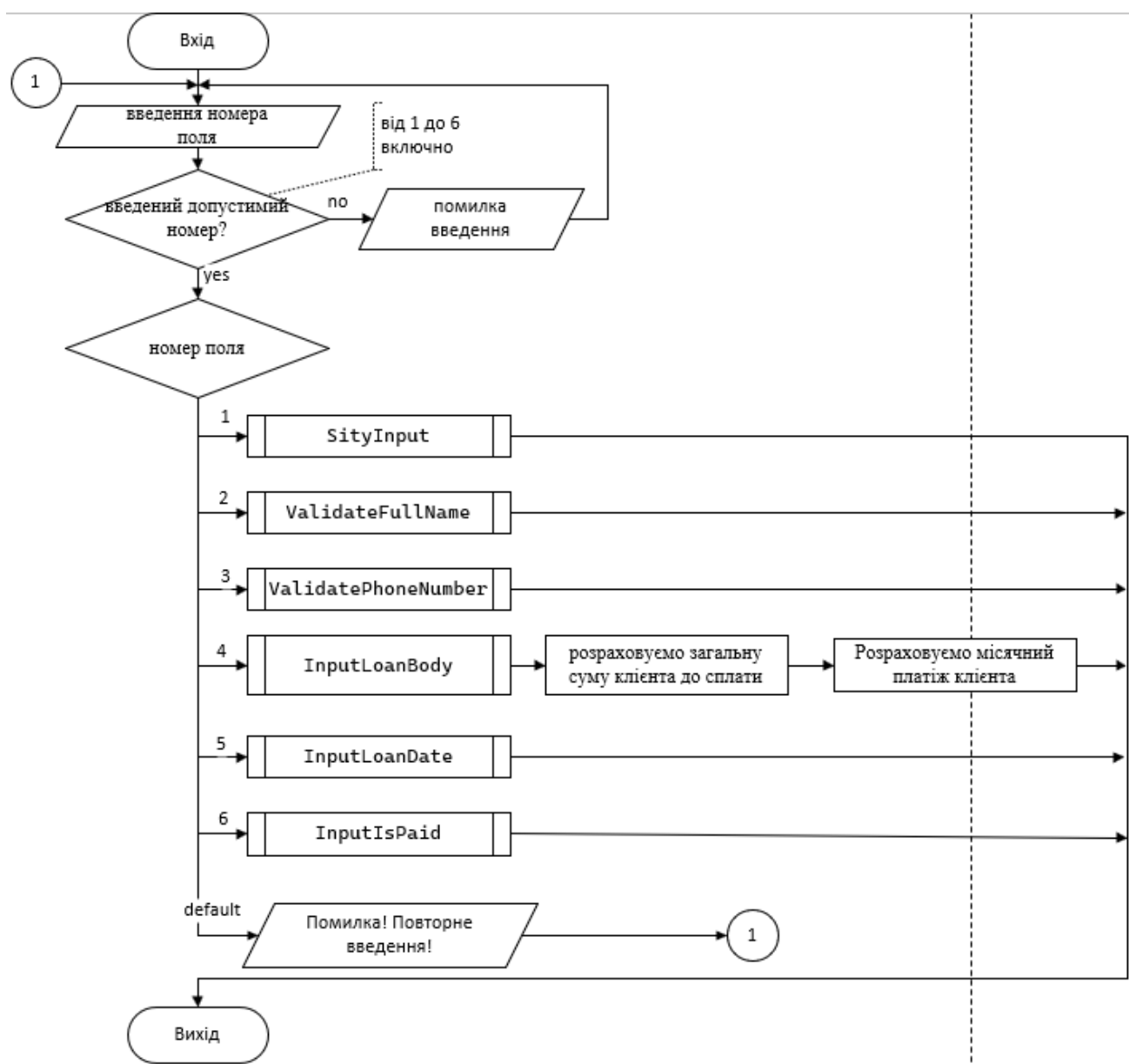


Рисунок В.18 – Структурна схема функції void EditClientField(Client& client)

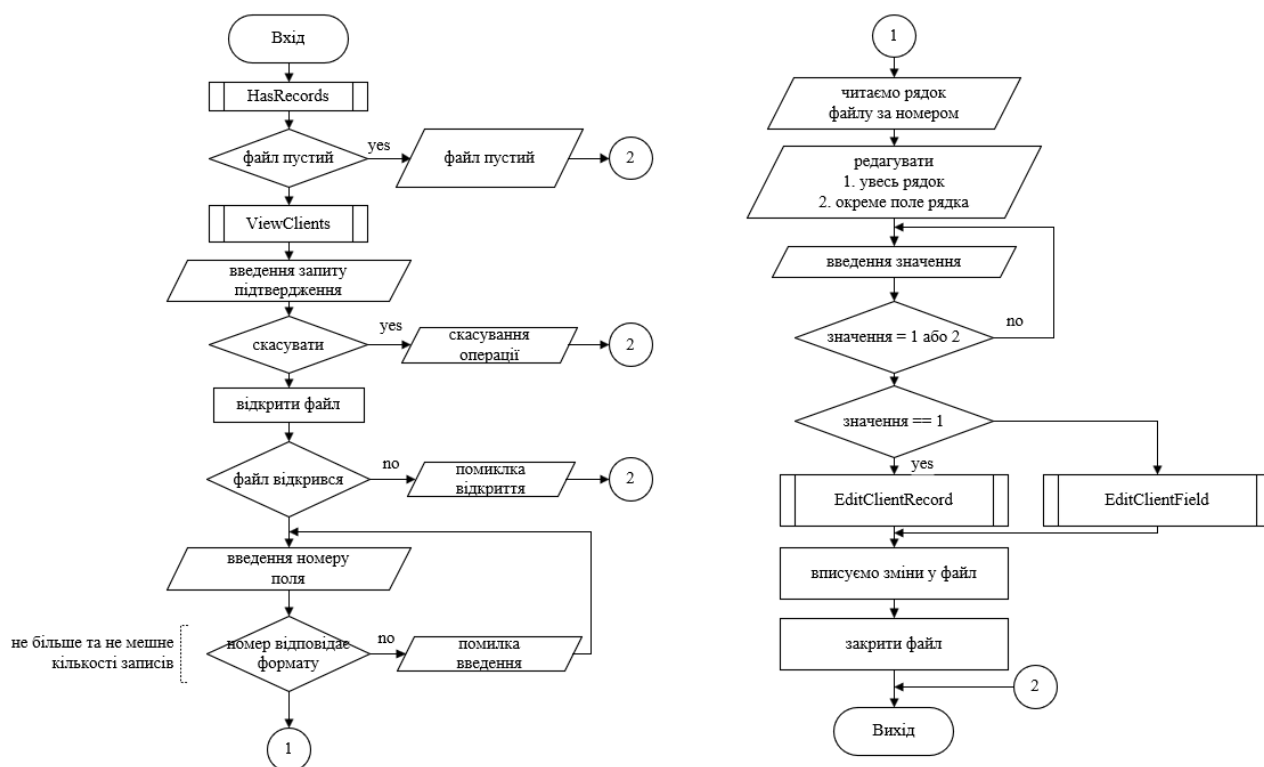


Рисунок В.19 – Структурна схема функції void EditClient(const string& path)

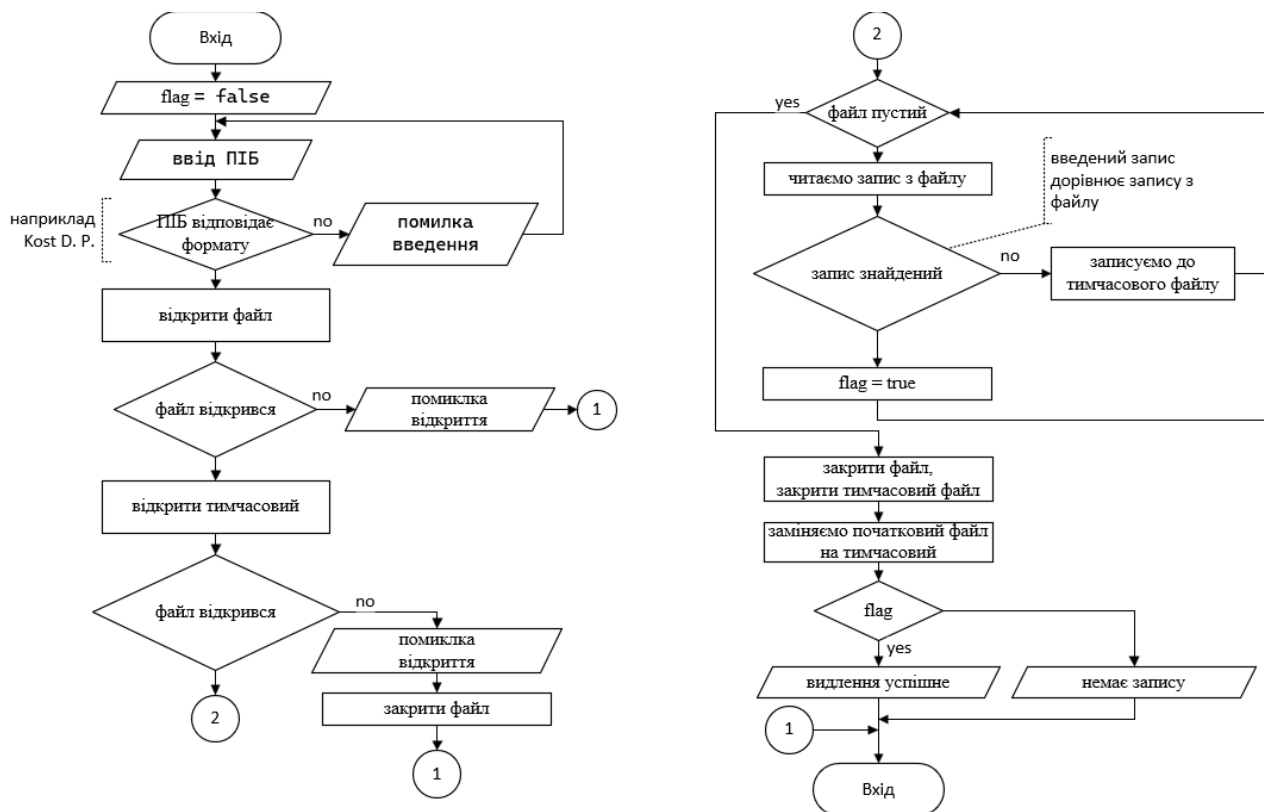


Рисунок В.20 – Структурна схема функції void DeleteRecordByName()

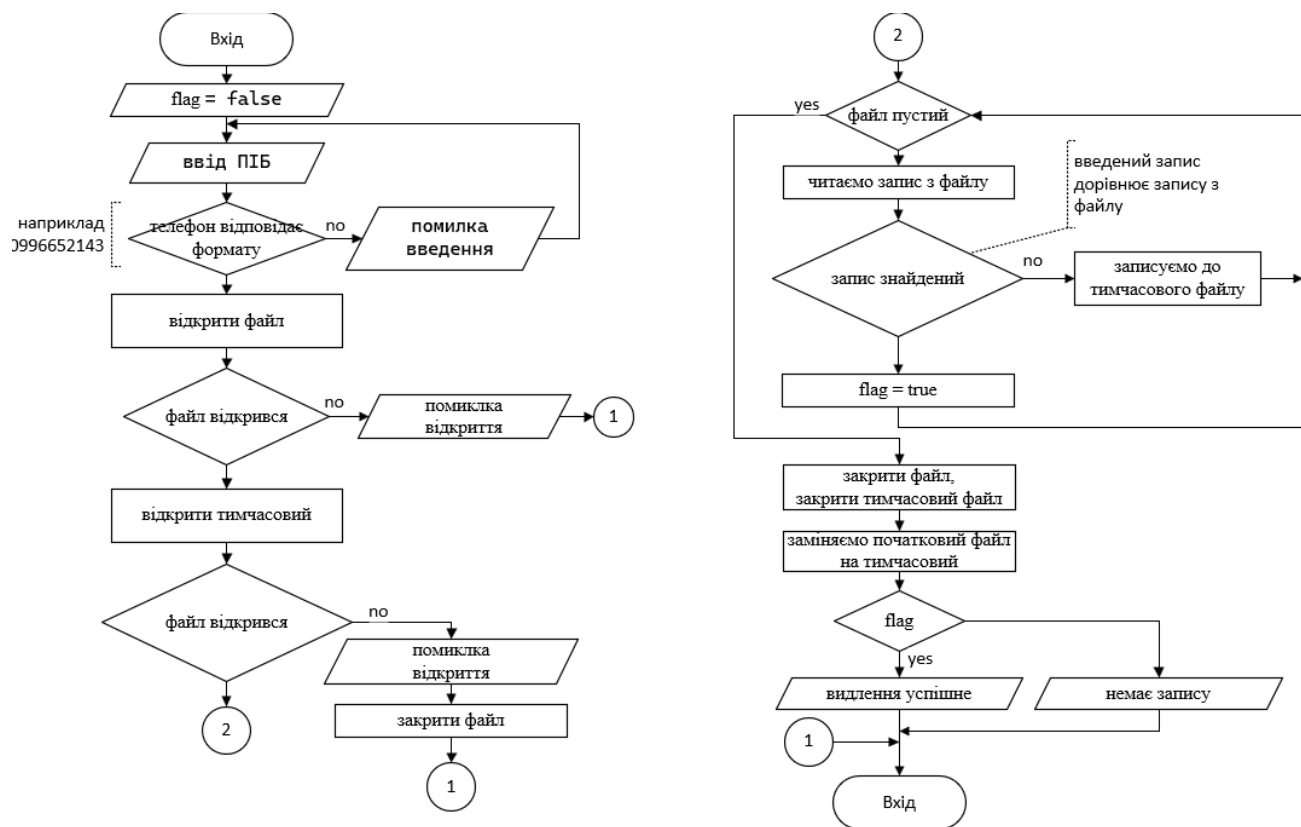


Рисунок В.21 – Структурна схема функції void DeleteRecordByPhone(const string& path)

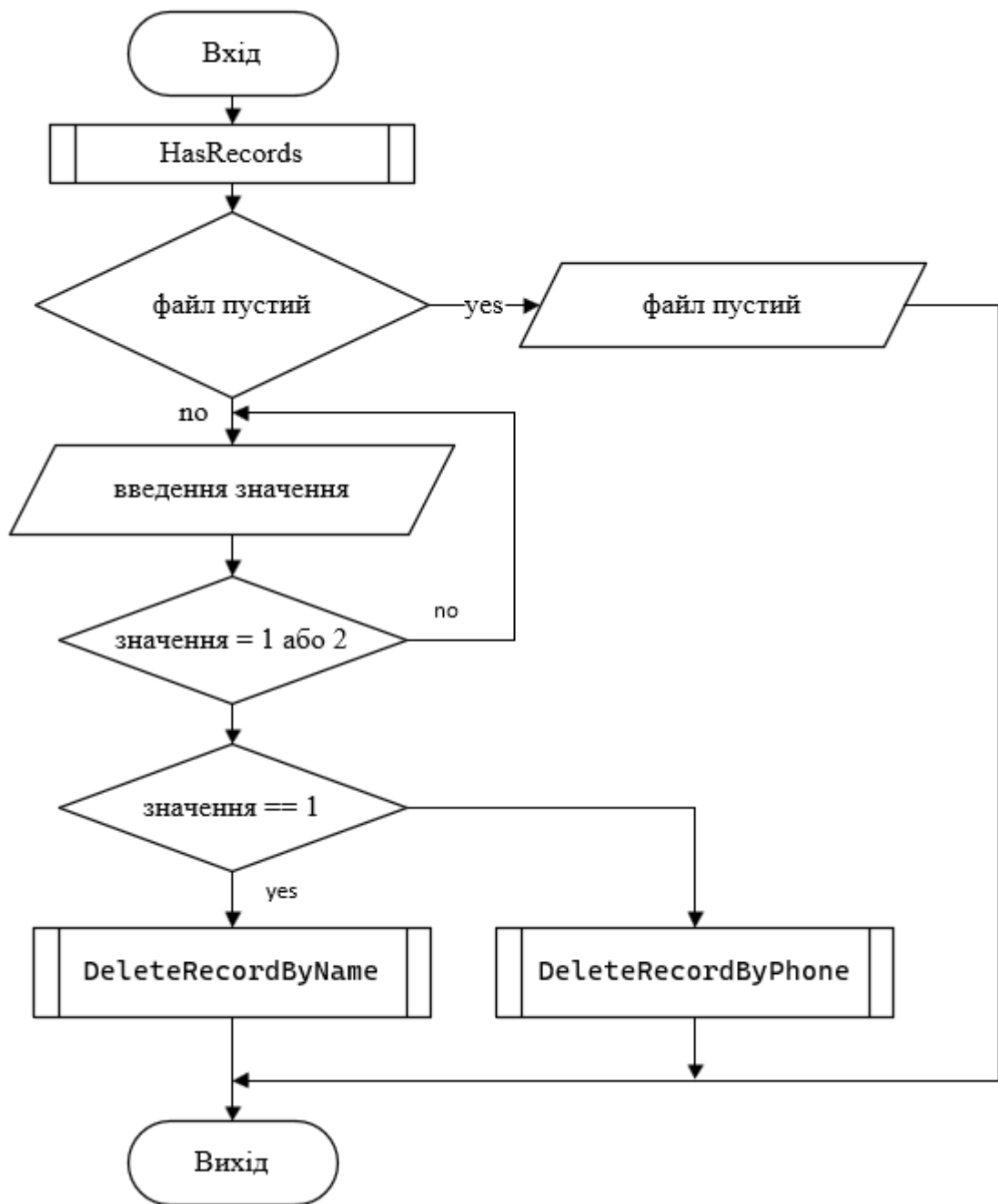


Рисунок В.22 – Структурна схема функції `void DeleteClient(const string& path)`

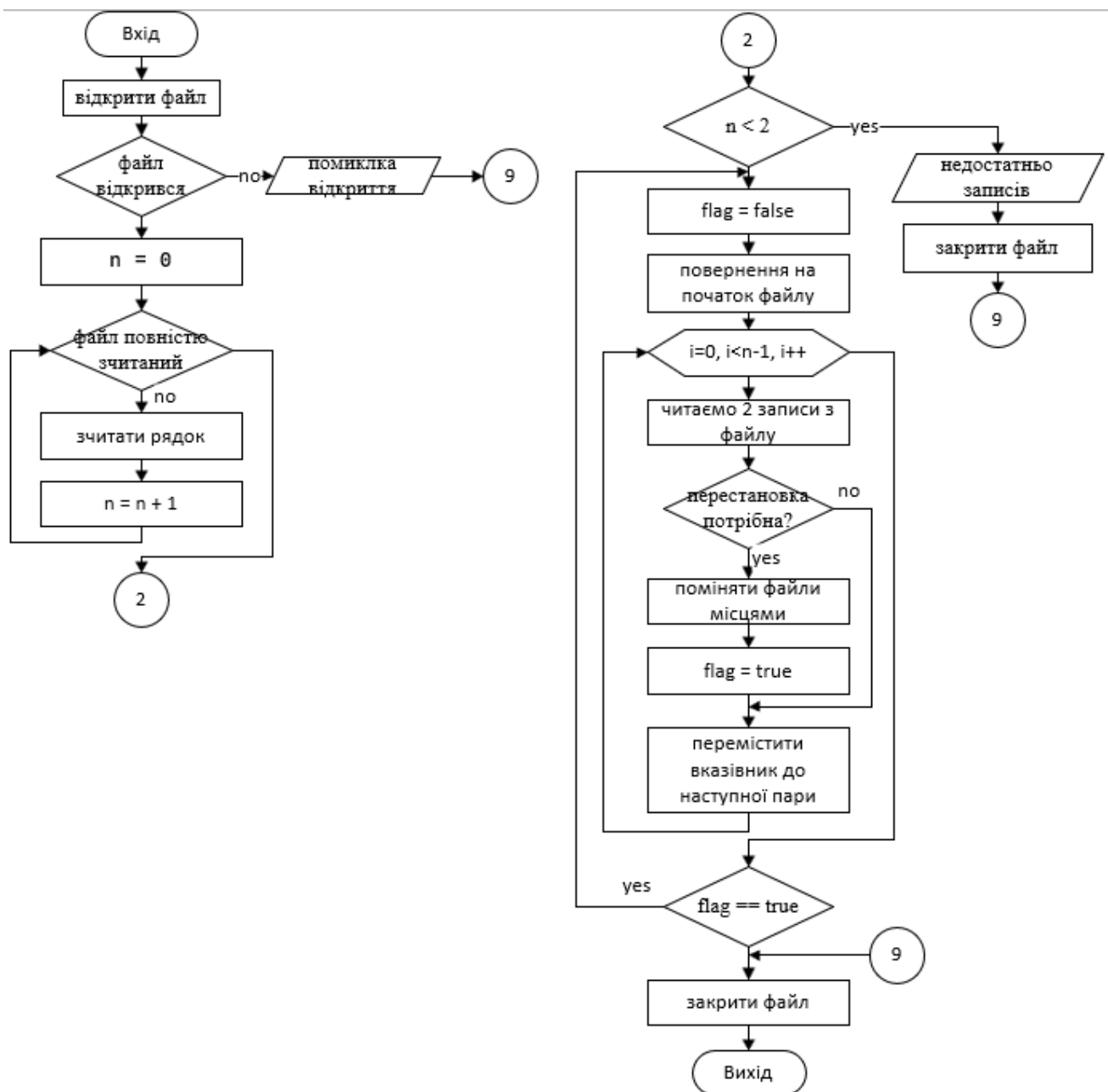


Рисунок В.23 – Структурна схема функції void SortRecordsByCity()

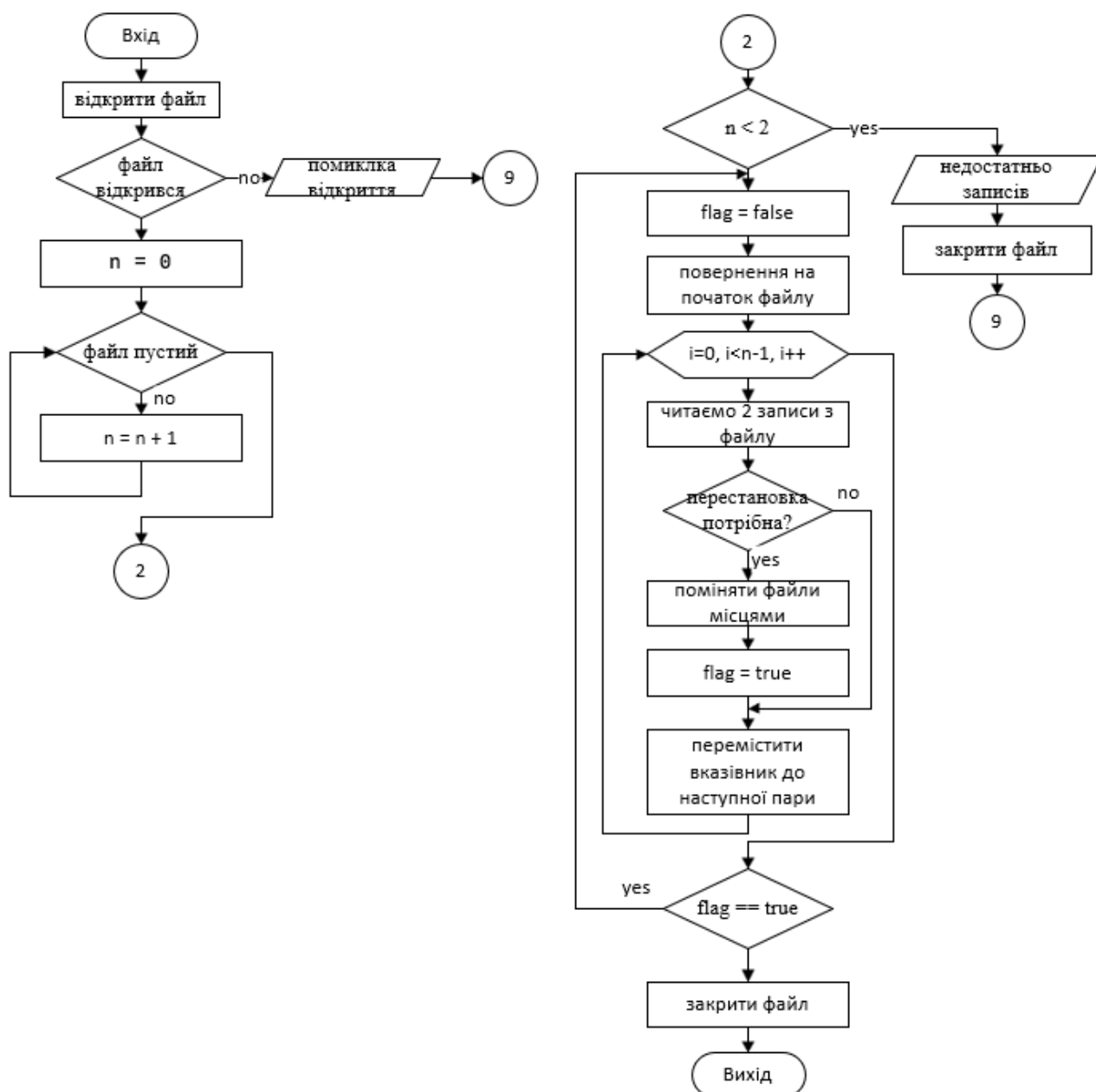


Рисунок В.24 – Структурна схема функції void SortRecordsBySumOfCredit()

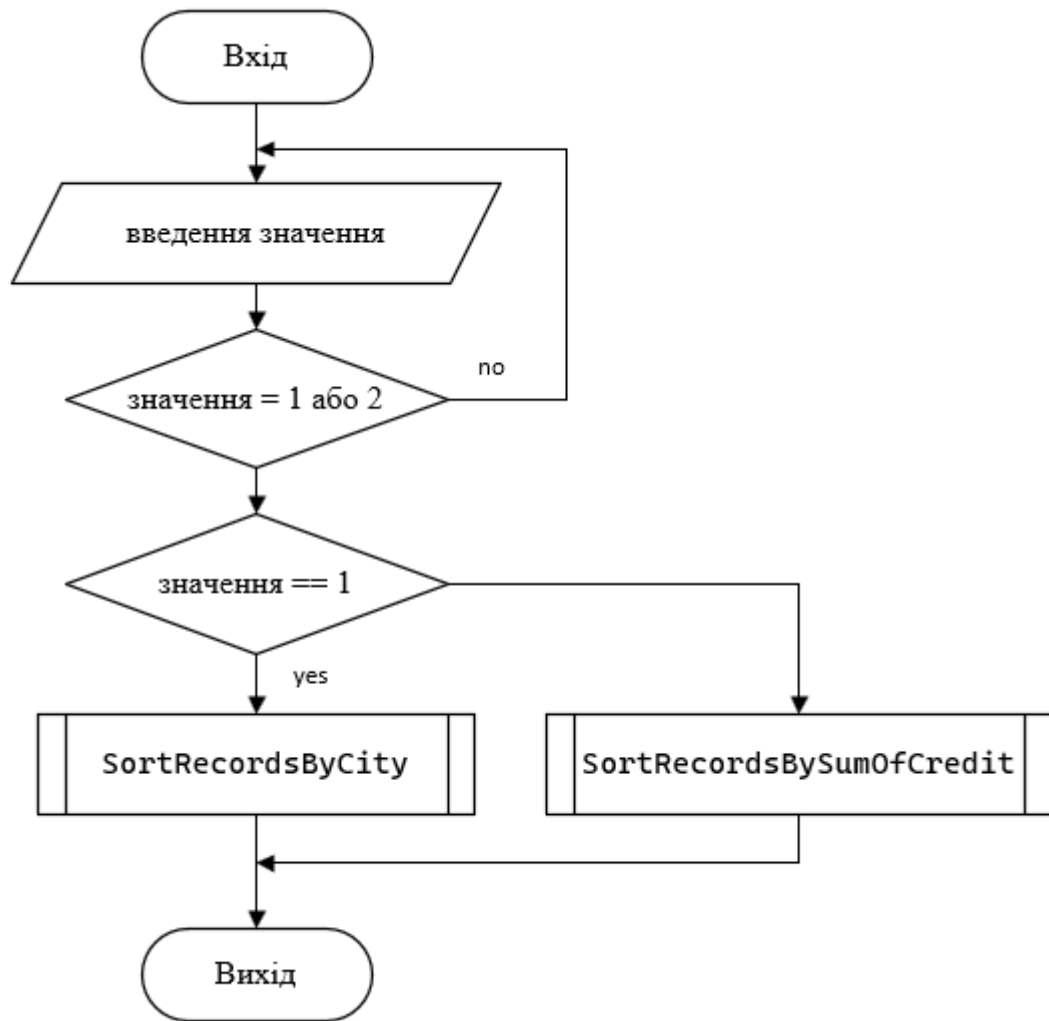


Рисунок В.25 – Структурна схема функції void SortRecords()

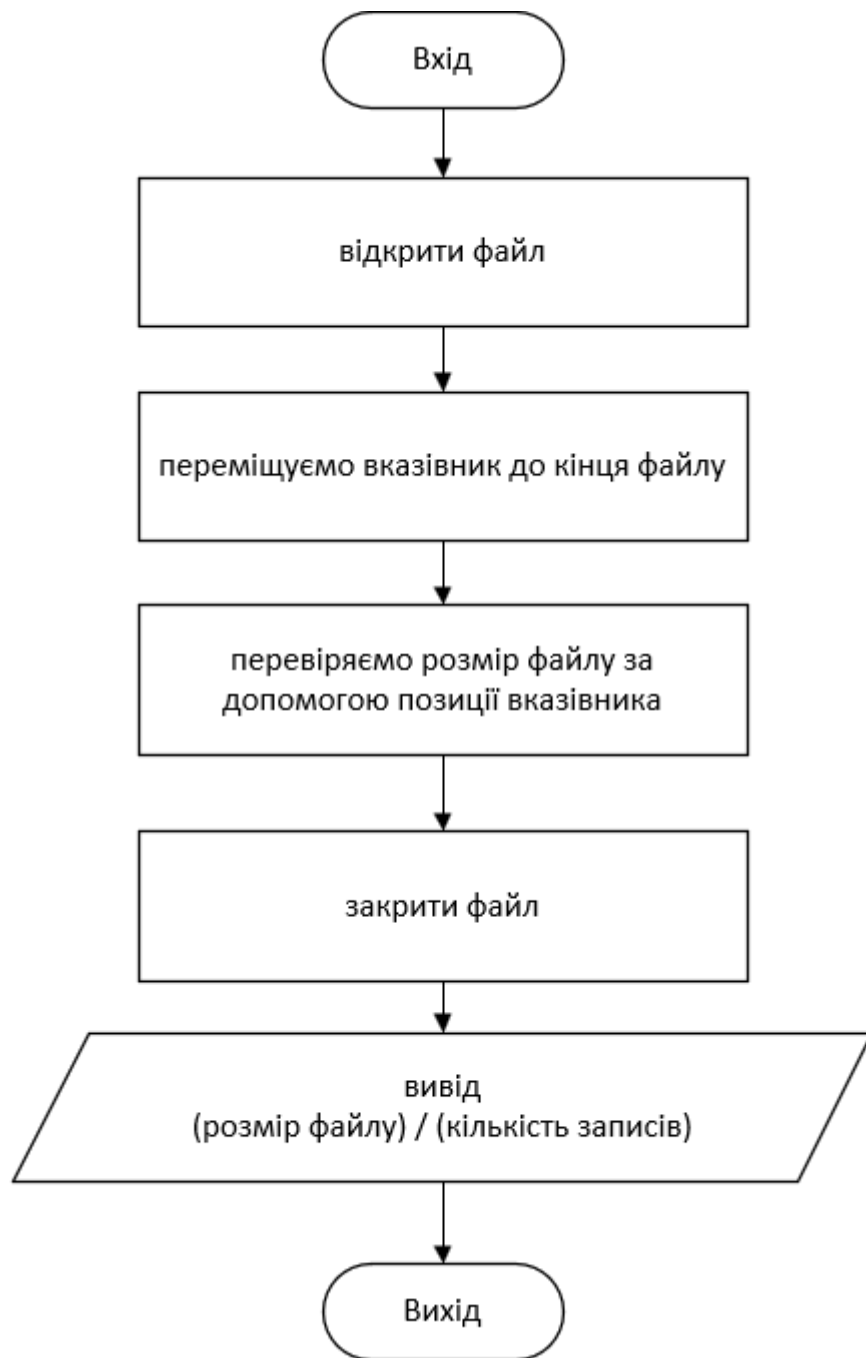


Рисунок В.26 – Структурна схема функції `int OnlyOneRecord(const string& path)`

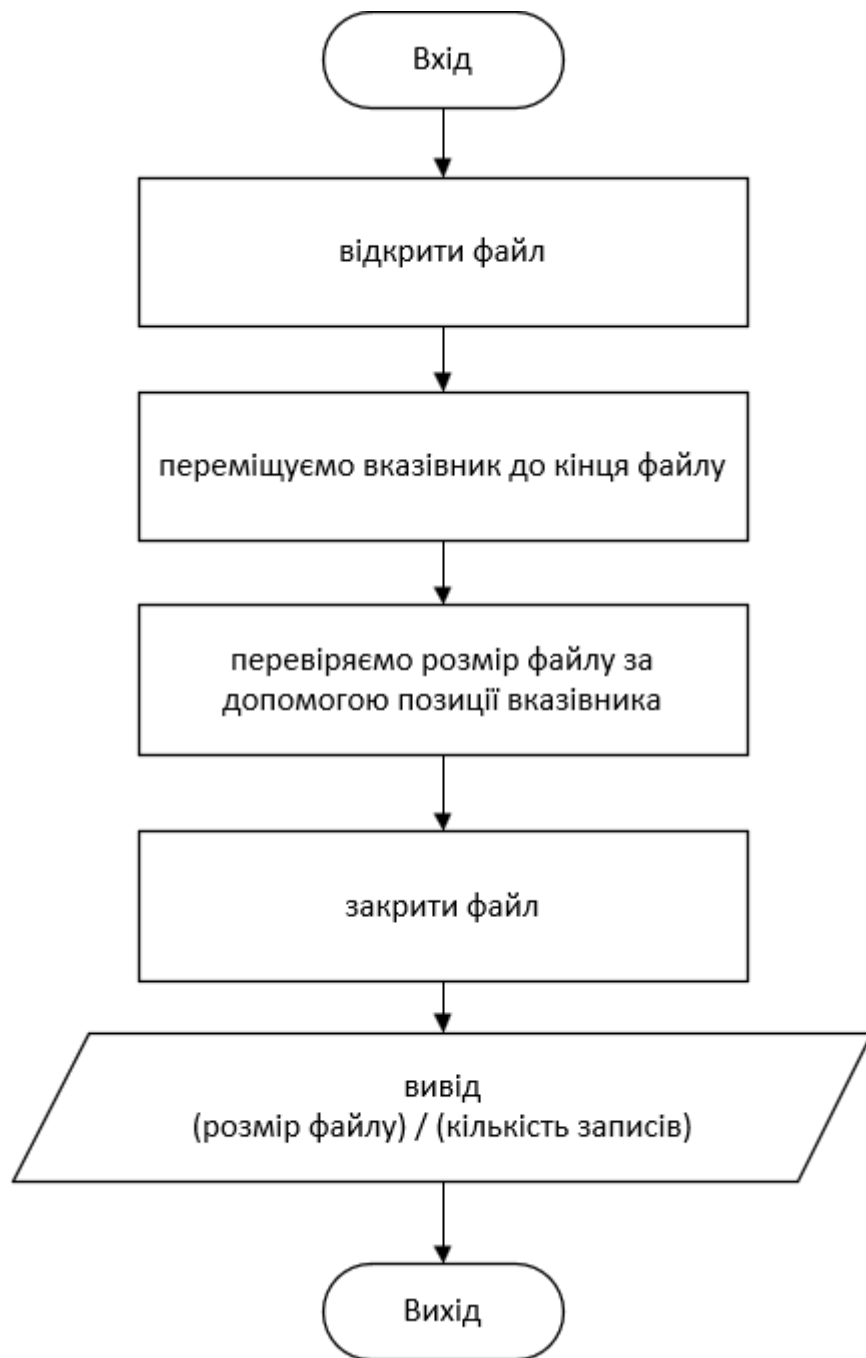


Рисунок В.27 – Структурна схема функції `int OnlyOneRecord(const string& path)`

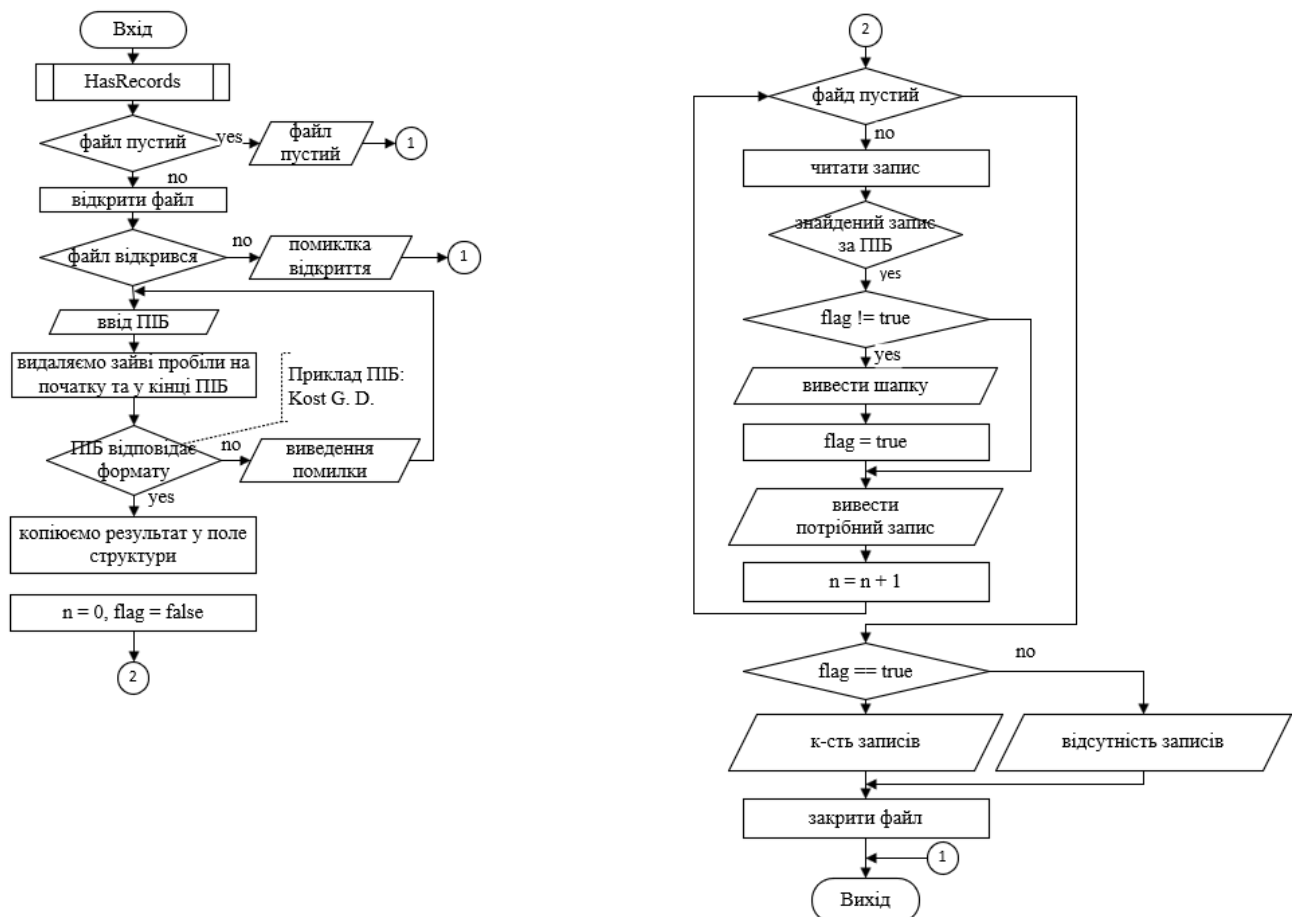


Рисунок В.28 – Структурна схема функції void FIRST_ByClientName(const string& path)

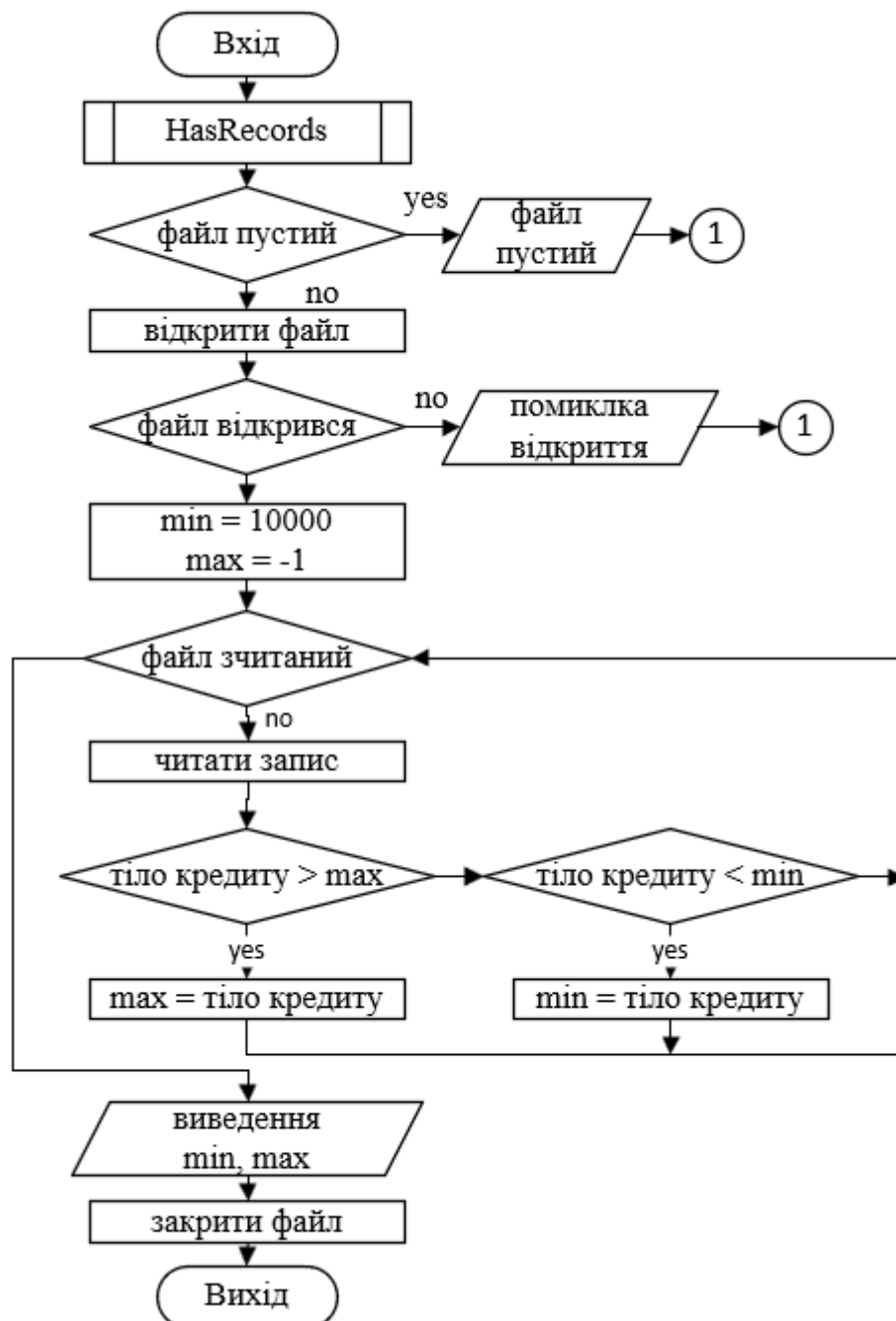


Рисунок В.29 – Структурна схема функції void Second_MAX_MIN(const string& path)

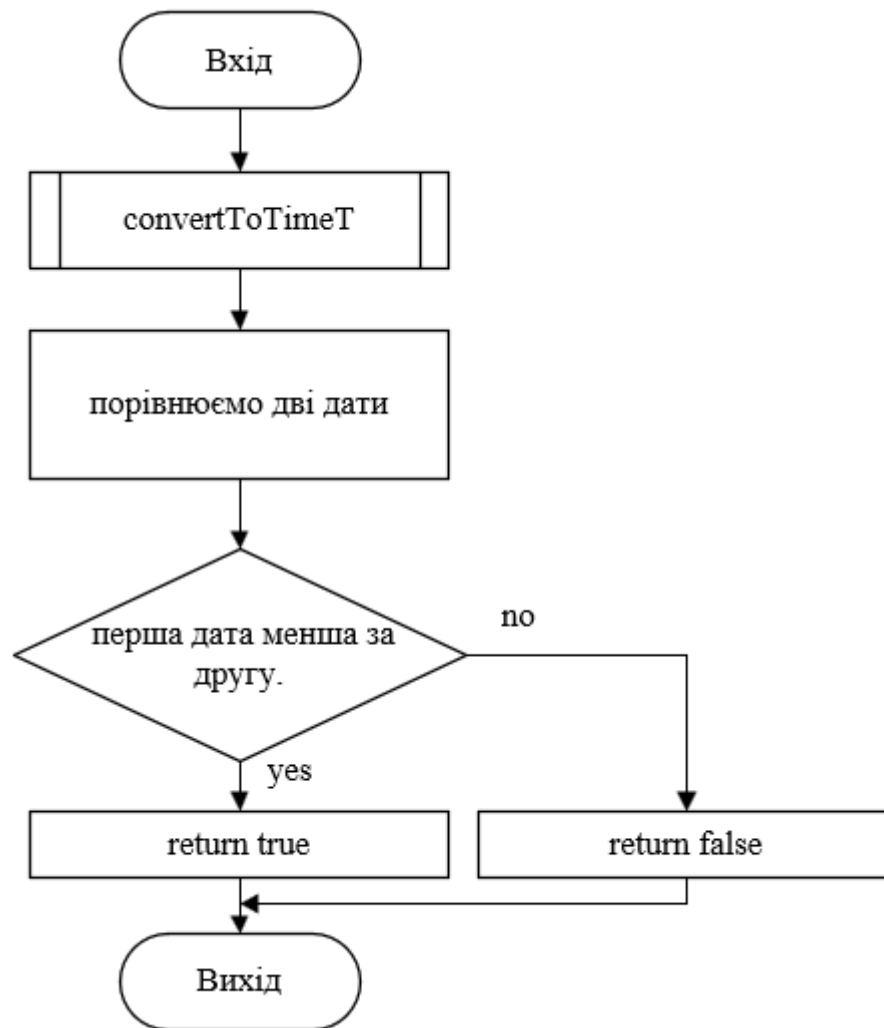


Рисунок В.30 – Структурна схема функції `bool compareDates(const char date1[], const char date2[])`

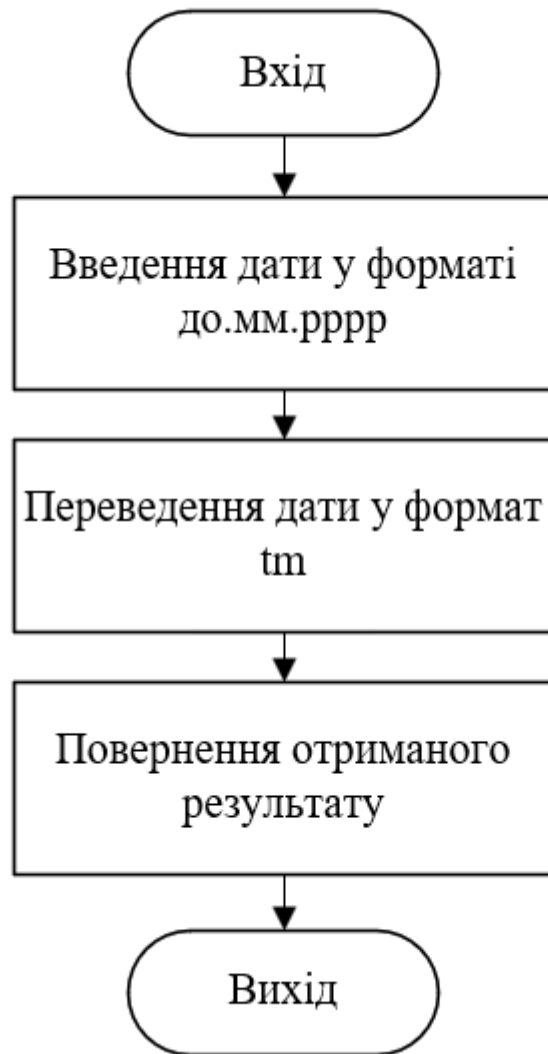


Рисунок В.31 – Структурна схема функції `time_t convertToTimeT(const char date[])`

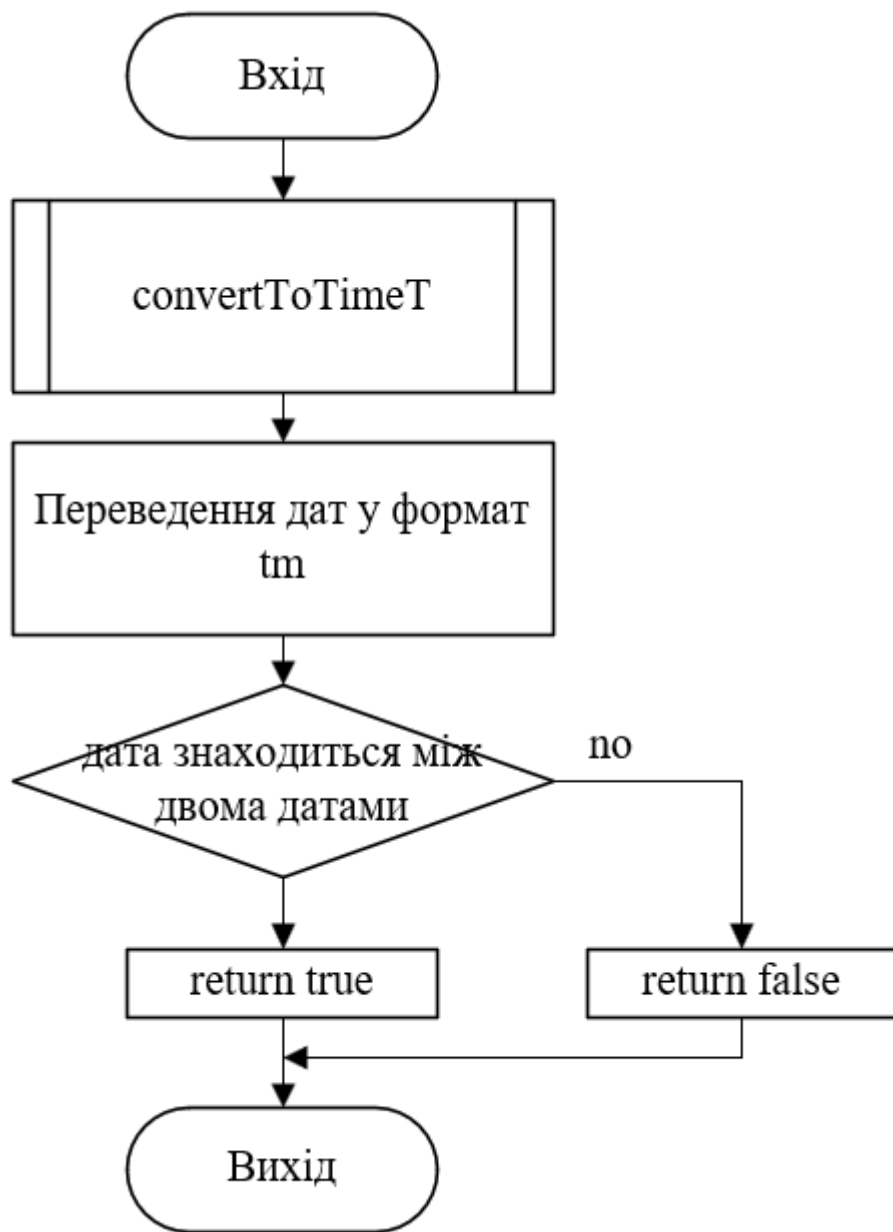


Рисунок В.32 – Структурна схема функції `bool isDateBetween(const char targetDate[], const char startDate[], const char endDate[])`

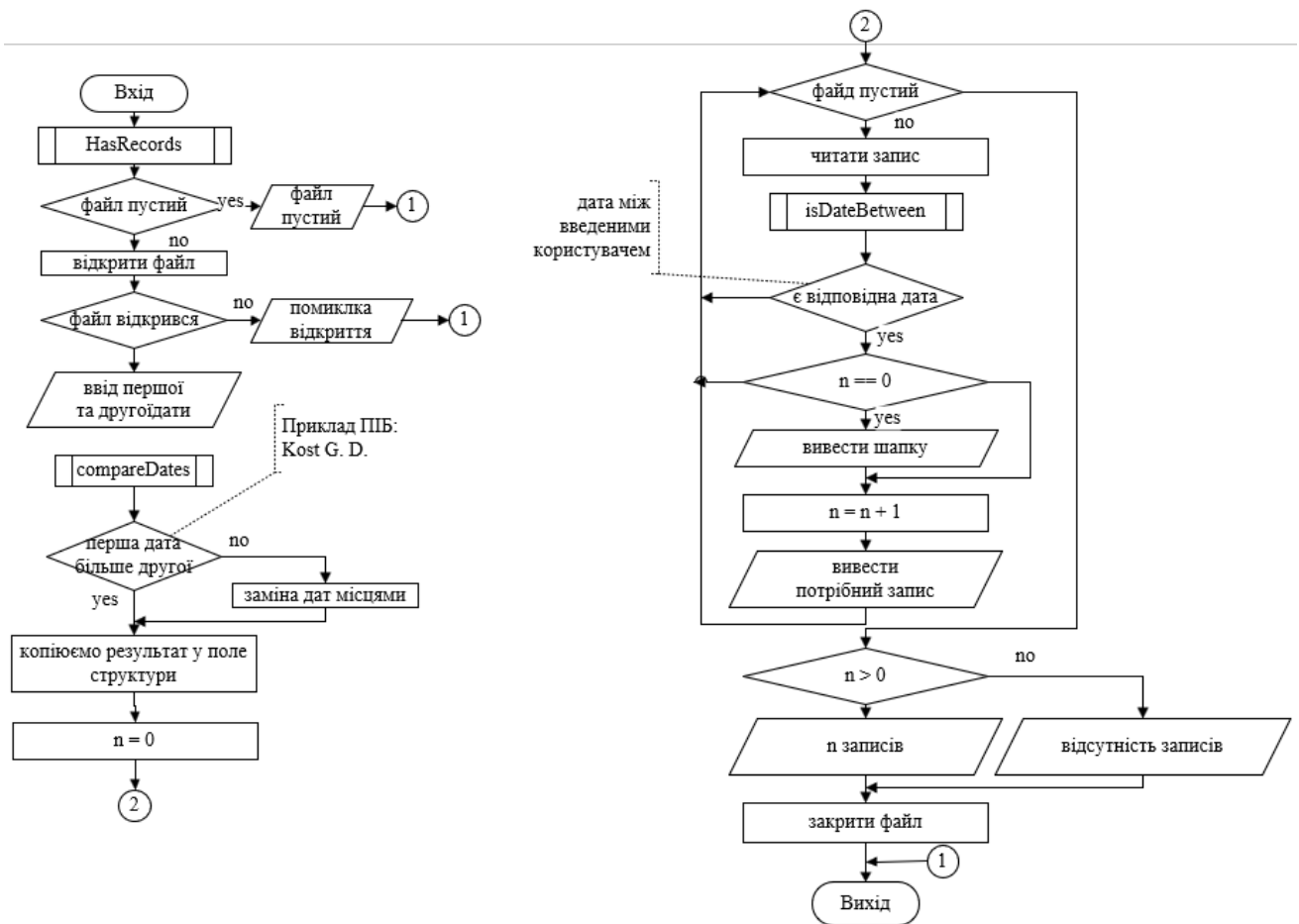


Рисунок В.32 – Структурна схема функції void Third_CreditsByDate(const string& path)

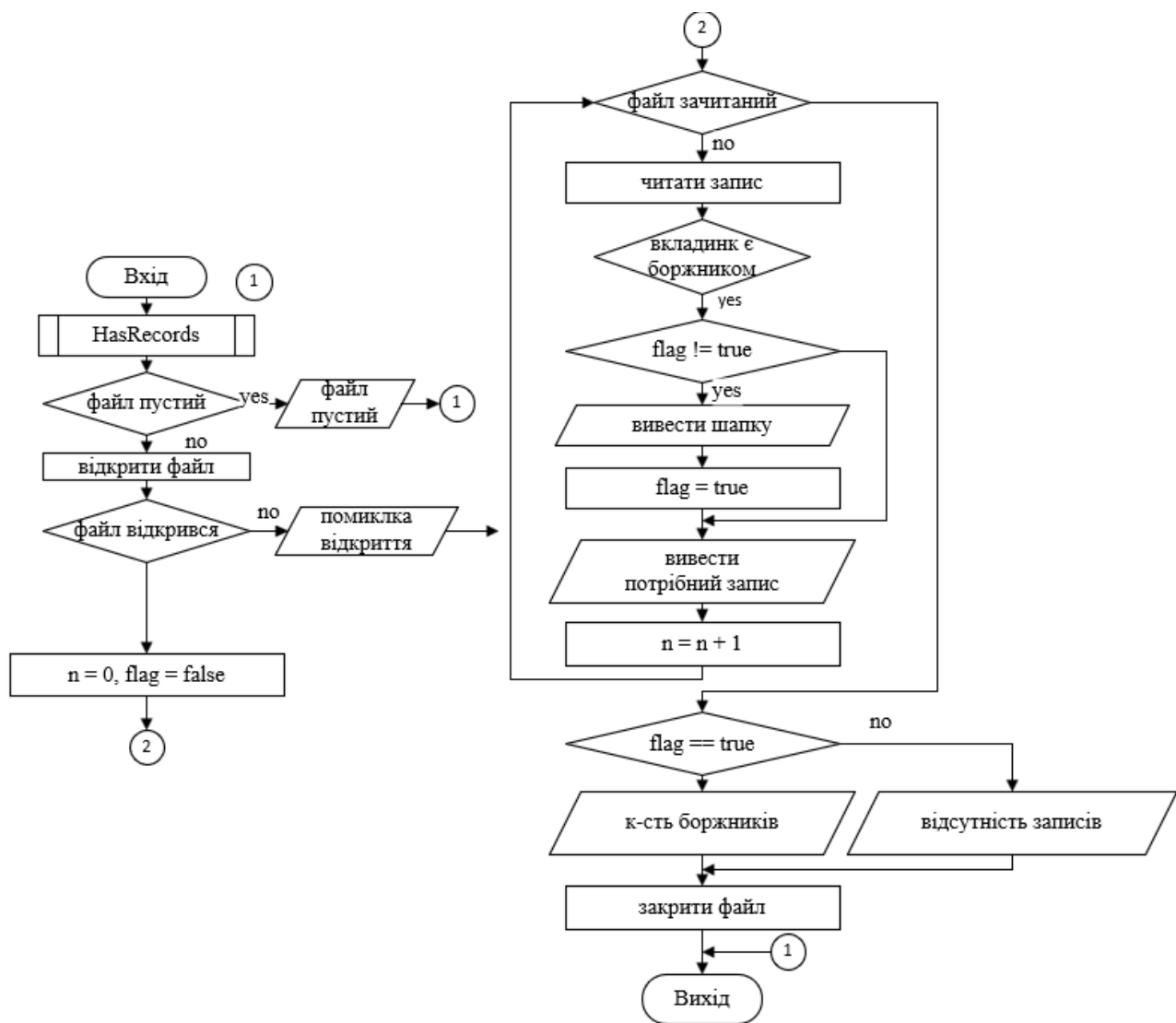


Рисунок В.33 – Структурна схема функції void Forth_Debetors(const string& path)

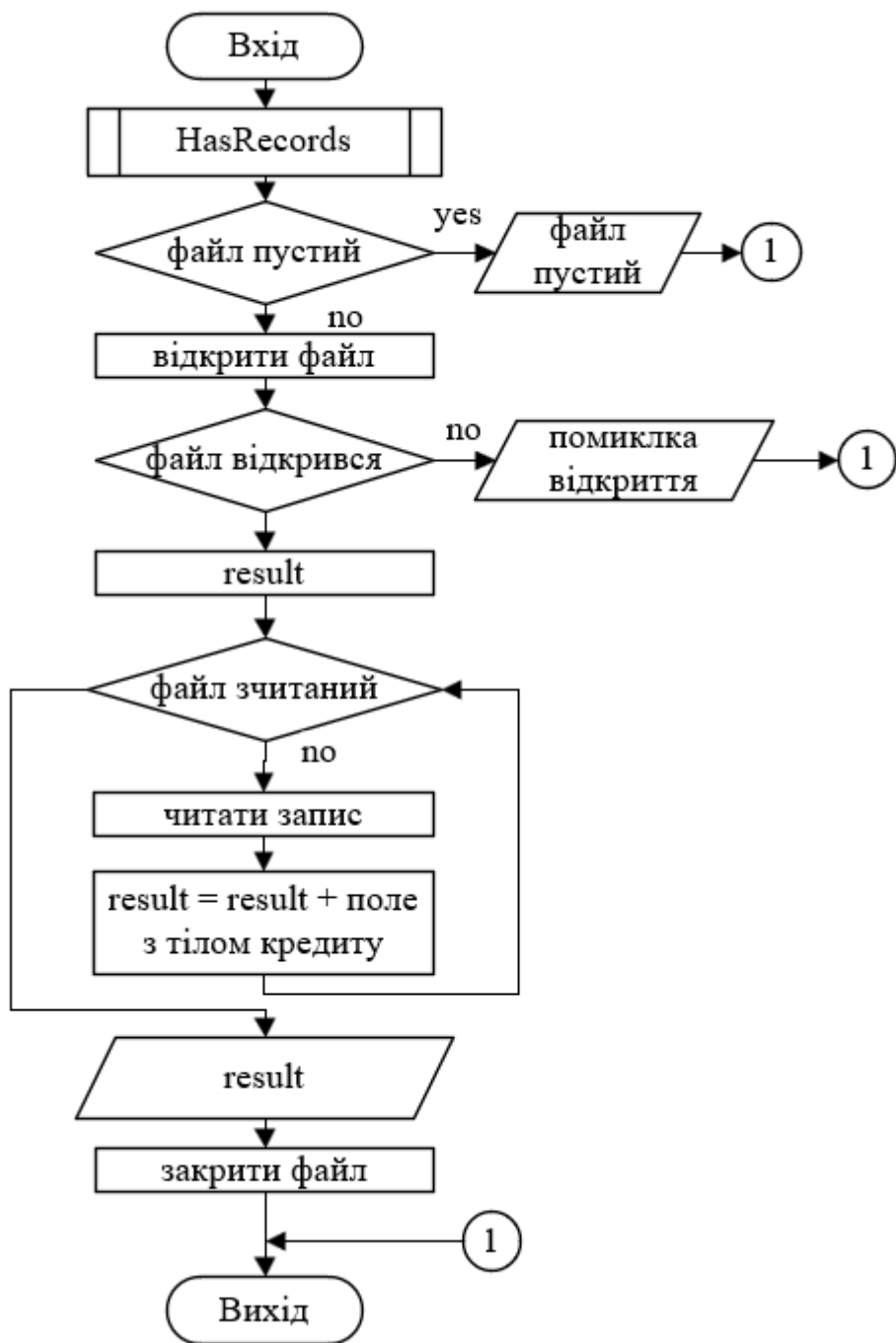


Рисунок В.34 – Структурна схема функції void Fifth_TotalAmount(const string& path)

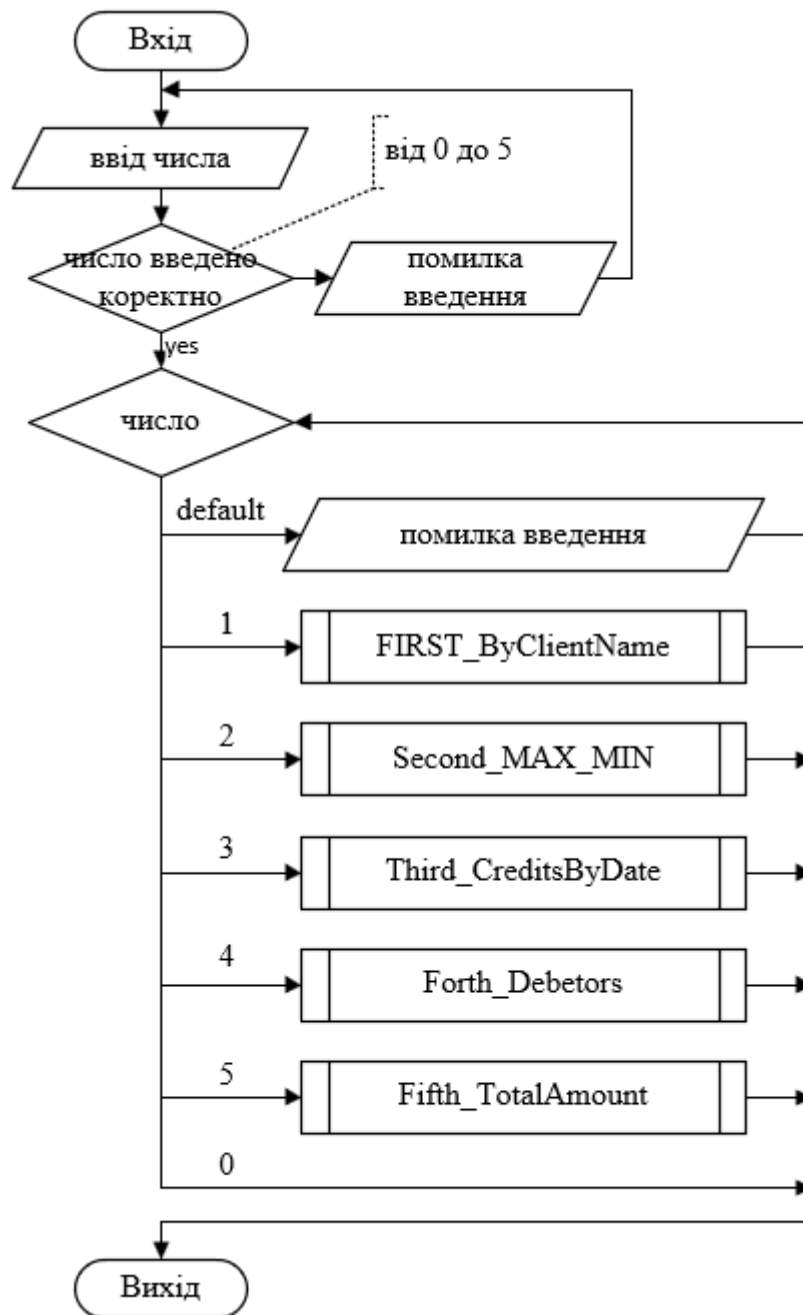


Рисунок В.35 – Структурна схема функції void Requests()