Фаховий коледж електронних приладів
Івано-Франківського національного технічного університету нафти і газу

Циклова комісія Програмної інженерії

Галузь знань 12 Інформаційні технології

Спеціальність 121 Інженерія програмного забезпечення

КУРСОВИЙ ПРОЄКТ

з дисципліни "Основ програмування та алгоритмічних мов"

на тему: "Розробка програми на мові Сі"

ПОЯСНЮВАЛЬНА ЗАПИСКА

КП.ПІ-18-01.11.00.00.000 ПЗ

Студента ІІІ курсу ПІ-18-01 групи
галузь знань 12
спеціальності 121
А. В. Косак
Керівник курсового проєкту
викладач О. К. Балабаник

Національна шкала_____
Кількість балів:

Члени комісії:

_____	_Береговський В.В._____
(підпис)	(прізвище та ініціали)
_____	_Гнатюк С.О._____
(підпис)	(прізвище та ініціали)
_____	_Балабаник О.К._____
(підпис)	(прізвище та ініціали)

Дата захисту “07” грудня 2020 р.

Івано-Франківськ – 2020

Фаховий коледж електронних приладів

Івано-Франківського національного технічного університету нафти і газу

Відділення Програмної інженерії та метрології
Циклова комісія Програмної інженерії
Освітньо-кваліфікаційний рівень Молодший спеціаліст
Галузь знань 12 Інформаційні технології
Спеціальність 121 "Інженерія програмного забезпечення"

Дисципліна "Основи програмування та алгоритмічні мови"
Курс III Група ПІ-18-01 Семестр V

ЗАВДАННЯ
НА КУРСОВИЙ ПРОЄКТ

Студентки Косак Аліни Василівни

(прізвище, ім'я, по-батькові)

1 Тема курсового проекту «РОЗРОБКА ПРОГРАМИ НА МОВІ СІ»

2 Термін здачі студентом закінченої роботи "06" грудня 2020 року.

3 Вихідні дані до проєкту: масив записів houses з наступними полями:
street – назва вулиці (текстове поле розміром 30);
number_house – номер будинку (числове поле цілого типу);
kol_meshk – кількість жителів (числове поле цілого типу);
year_construct – рік побудови будинку (числове поле цілого типу);
date – дата постановки на облік (текстове поле довжиною 11 символів);
repair – наявність ремонту за останні 20 років (так/ні) (текстове поле довжиною 4 символи).

4 Зміст пояснювальної записки (перелік питань, що їх належить розробити)

Зміст. Вступ. Уточнення постановки завдання. Розробка специфікацій та вибір методу розв'язання задачі. Перелік та призначення режимів та структури діалогу. Структура даних, з якими працює програма та їх характеристика. Опис програми. Висновки. Перелік використаних джерел.

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

Блок-схема програми

6 Дата видачі завдання "09" вересня 2020 р.

Фаховий коледж електронних приладів

Івано-Франківського національного технічного університету нафти і газу

ВІДГУК
на курсовий проєкт (роботу)

Студентки Косак Аліни Василівни
(прізвище, ім'я, по-батькові)

Курсу III Групи ПІ-18-01

Спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Тема проекту (роботи) Розробка програми на мові Сі

Оцінка якісного рівня курсового проекту (роботи)

Найменування виду роботи	**Оцінка за 12-бальною шкалою**
1. Відповідність змісту проекту (роботи) технічному завданню на проектування	
2. Новизна розробки	
3. Ступінь складності розробки	
4. Правильність і якість виконання графічної частини	
5. Повнота і якість оформлення пояснювальної записки	
6. Ступінь самостійності виконання проекту (роботи)	
7. Вміння працювати з науково-технічною літературою	
8. Вміння працювати з комп'ютерною технікою та комп'ютерними програмами	
9. Степінь використання технічної документації, стандартів	
10. Оцінка стилю та грамотності викладення тексту	

Зауваження:

_____

_____

_____

_____

Допускається до захисту

Керівник курсового проекту (роботи) _____ _____
(підпис) (прізвище та ініціали)

"__" _____ 2020 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проєкту	Термін виконання етапів проєкту	Примітка
1.	Одержання завдання, ознайомлення та вибір літератури	**22.09-25.09**	Виконано
2.	Розробка специфікацій та вибір методу розв’язання задачі	**29.09-02.10**	Виконано
3.	Розробка словесного алгоритму, блок-схеми, таблиці ідентифікаторів	**06.10-16.10**	Виконано
4.	Розробка програми	**20.10-30.10**	Виконано
5.	Відладка та тестування програми	**03.11-13.11**	Виконано
6.	Оформлення пояснювальної записки і графічної частини	**17.11-20.11**	Виконано
7.	Підготовка роботи до захисту	**23.11-30.11**	Виконано

Студентка _____ Косак А.В.
(підпис) (прізвище та ініціали)

Керівник проєкту _____ Балабаник О.К.
(підпис) (прізвище та ініціали

ЗМІСТ

ВСТУП

В основі технології розробки програм лежить технологія структурного програмування. Структурне програмування реалізовує розроблення програмних продуктів, яка об'єднує способи розроблення структури програми, зручної для читання та розуміння людиною, стеження за логікою її роботи, внесення до неї виправлень та інших змін.

Принцип поділу програми на окремі модулі полягає у тому, що будь-яку складну програму доцільно розділити на логічно незалежні частини (модулі), дотримуючись при цьому певних зв’язків між ними. Модуль – послідовність логічно пов'язаних команд, який оформлено у вигляді окремої програми. Ці допоміжні програми можна розробляти й аналізувати окремо та незалежно одну від іншої, використовуючи їх потім у основній програмі або інших допоміжних програмах. Структурний підхід до розроблення програм і принцип її модульності також привів до ідеї розподілу робіт серед розробників програм.

Мова програмування С є мовою високого рівня, але в ній закладені можливості, які дозволяють програмістові (користувачеві) працювати безпосередньо з апаратними засобами комп’ютера і спілкуватися з ним на досить низькому рівні. Багато операцій, що виконуються на мові С, схожі на мову Асемблера. Тому мову С часто називають мовою середнього рівня.

Одна із базових концепцій С – високий рівень довіри до програміста. Створена для потреб програмістів-професіоналів, мова програмування надає користувачеві широку свободу вибору форм даних і засобів програмування. Водночас потрібно пам'ятати, що відсутність контролю може призвести до небезпечних помилок у роботі програми, тому на програміста лягає значно вища, ніж в інших мовах, відповідальність за правильність результатів виконання програми.

Для написання програм в практичних розділах часто використовується компілятор мови С++, а програмування ведеться в середовищі Microsoft Visual Studio 2010\19. Visual Studio – інтегроване середовище розробки програмного забезпечення від фірми Microsoft, яке дозволяє створювати різноманітні програмні

продукти: консольні програми, програми з графічним інтерфейсом, наприклад, віконні додатки Windows Forms, а також Web-додатки тощо. Середовище розробки Visual C++ надає всебічну підтримку при управлінні проєктами і їх налаштуванні, редагуванні та перегляді початкового коду, а також потужні засоби відладки. Середовище розробки підтримує технологію IntelliSense (надає при написанні коду детальні підказки, що враховують контекст).

Мета курсової роботи – розробити готове, ефективне програмне рішення для роботи з даними, які зберігаються в файлі.

Завдання курсової роботи:

1) Навчитися правильно взаємодіяти з файлом та великим обсягом даних в ньому за допомогою файлових потоків.

2) Правильно використати динамічну пам’ять.

3) Розробити алгоритм додавання, видалення та сортування даних.

4) Забезпечити валідацію введених даних.

5) Розробити зручний та інтуїтивно зрозумілий інтерфейс.

КП розроблено відповідно до програми навчальної дисципліни «Технології» (Основи програмування).

Робота над курсовим проектом дозволяє сформувати наступні компетентності:

- здатність до алгоритмічного та логічного мислення;

- здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

- здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв’язання завдань інженерії програмного забезпечення;

- володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних, які забезпечуються отриманням концептуальних сучасних знань і вмінь розв’язувати складні задачі, створенням у тих, хто навчається, спроможності донесення до фахівців власного досвіду, а також відповідальності за прийняття рішень у різноманітних умовах.

1 УТОЧНЕННЯ ПОСТАНОВКИ ЗАВДАННЯ

Виконуючи постановку завдання та створюючи програму, яка виконуватиме згодом різноманітні дії, спочатку потрібно проаналізувати предметну область. Дана ПО для дослідження - аварійні будинки в районі міста на конкретних вулицях.

Відповідно для виконання завдання спочатку сформовано текстовий файл, у якому записані дані предметної області, які можуть підлягати ремонту або подальшому зносу, залежно від ситуації. В записі міститься інформація, за якою можна легко прийняти подальше рішення щодо будинку, а саме: вулиця, на якій розташований аварійний будинок, номер будинку, кількість мешканців у ньому, рік побудови, точна дата постановки на облік, а також вказані дані щодо ремонту, тобто, чи проводився він впродовж останніх 20 років.

Наступним кроком є аналіз вимог до програми. Використовуючи, як основу, предметну область, програма виконує ряд операцій над нею, які видають достовірну інформацію. Згідно із постановкою завдання програма сортує дані за зростанням року побудови, здійснює вибірку даних про будинки, у яких ремонт не проводився впродовж останніх 20 років, які побудовані до 1941 року, а також, в яких проживає понад 50 мешканців, здійснює вибірку даних з форматуванням документа, зокрема текстового файлу, про аварійні будинки, які поставлені на облік понад 10 років тому, містить функцію видалення даних про будинки, розташовані на вулиці Пішонівській, а також програма виконує ряд операцій, необхідних для комунальної служби аварійних будинків міста.

Працюючи з програмою, користувач має можливість вводити дані з клавіатури і заносити їх у базу, тобто сформований файл, додавати нові дані як на початок, так і в кінець файлу, здійснювати роботу з валідацією даних, читати дані з бази і виводити їх на екран, редагувати та видаляти записи з БД, виконувати пошук інформації чи упорядковувати записи в базі відповідно до потреб, а також здійснювати необхідні розрахунки тощо.

Проєкт багатофайловий, тобто прототипи функцій містяться в заголовному файлі, а саме визначення – у файлі-джерелі. Крім функцій, у роботі програми застосовуються ще масиви, структура, файли тощо. Програма створена за низхідним проектуванням, кожного разу уточнюючи більше різноманітні деталі.

Використовуючи надбані знання при проєктуванні, програма розбита на кілька менших підзадач, що виконують всі необхідні операції. Програма створена безпосередньо в середовищі MS Visual Studio 2019.

Завдання згідно з варіантом наведено в таблиці 1.1.

№ вар.	**Вміст бінарного файла**	**Параметри сортування**	**Відбір даних за умовою**	**Умова на відбір даних з формуванням документа (текстового файла)**	**Умова на видалення даних**
11	Список аварійних будинків у районі міста: вулиця, номер будинку, кількість мешканців у будинку, рік побудови, дата постановки на облік, наявність проведення ремонту за останні 20 років (так/ні)	За зростанням року побудови	Вивести відомості про будинки без ремонту за останні 20 років, побудовані до 1941 року, в яких понад 50 мешканців	Аварійні будинки, поставлені на облік понад 10 років тому	Аварійні будинки на вулиці Пішонівська

Таблиця 1.1 - Завдання згідно з варіантом

2 РОЗРОБКА СПЕЦИФІКАЦІЙ ТА ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

Програма, що розв'язує загальне завдання, спочатку розглядається як незалежний модуль, а згодом поділяється на підпрограми, які декомпонуються на підмодулі наступного рівня. Процес декомпозиції триває доти, доки не будуть отримані блоки, що є достатньо малими для їх безпосереднього кодування. При цьому керуючу програму проектують раніше, ніж реалізують її складові частини. Завдяки застосуванню методу низхідного проектування, програма ієрархічно структурується і розробляється шляхом послідовного уточнення на кожному рівні ієрархії. В основу цього процесу, крім принципу ієрархічності, покладено принципи абстрагування, специфікації інтерфейсів і модульності.

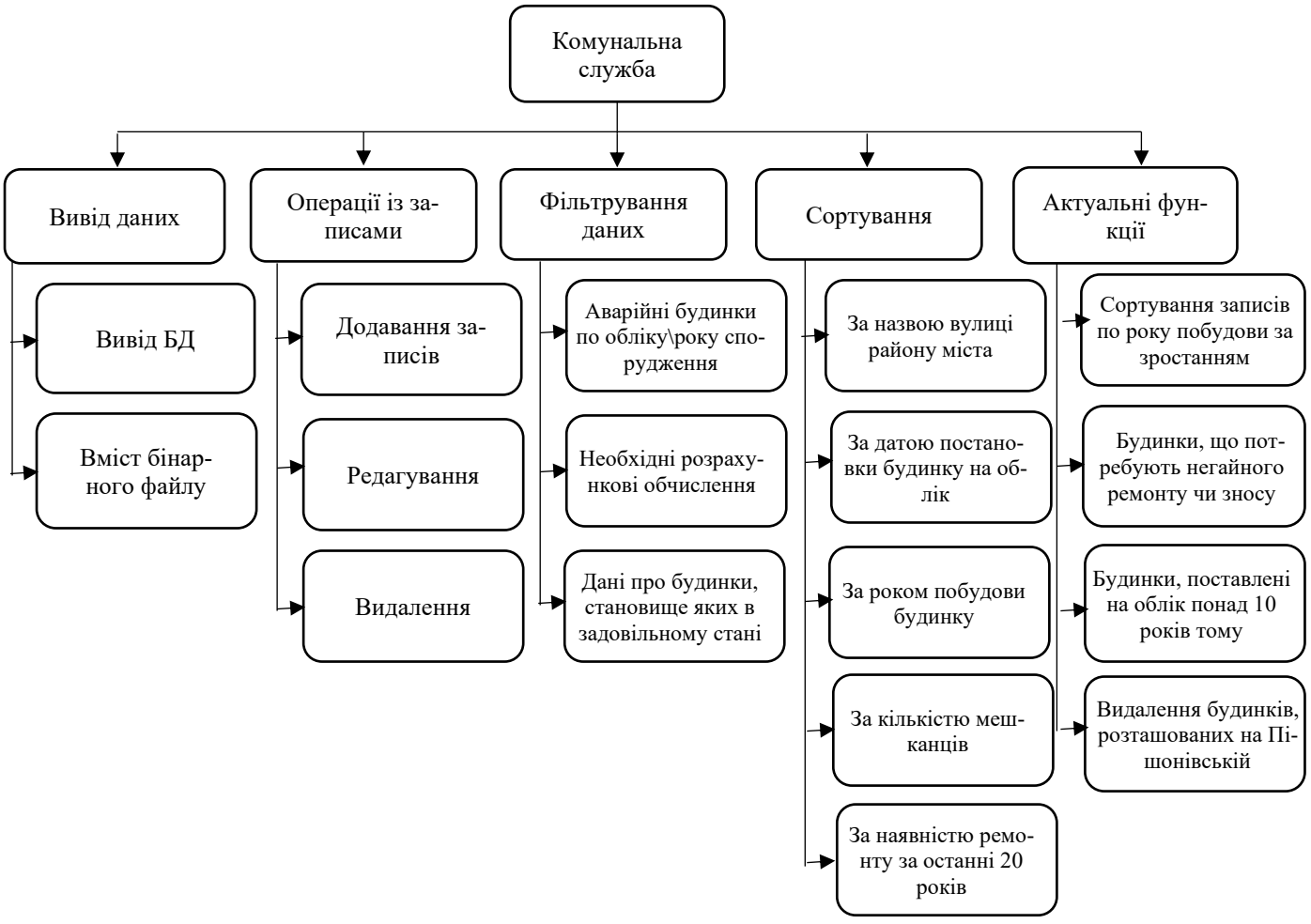


Рисунок 2.1 - Декомпозиція програми

Програма, що розв'язує загальне завдання, спочатку розглядається як

незалежний модуль, а згодом поділяється на підпрограми, які декомпонуються на підмодулі наступного рівня. Процес декомпозиції триває доти, доки не будуть отримані блоки, що є достатньо малими для їх безпосереднього кодування. При цьому керуючу програму проектують раніше, ніж реалізують її складові частини. Завдяки застосуванню методу низхідного проектування, програма ієрархічно структурується і розробляється шляхом послідовного уточнення на кожному рівні ієрархії. В основу цього процесу, крім принципу ієрархічності, покладено принципи абстрагування, специфікації інтерфейсів і модульності.

Модульне програмування — це організація програми у вигляді сукупності незалежних блоків, структура і функції яких підпорядковуються певним вимогам.

Перехід на нижчий рівень опису моделі може здійснюватись шляхом заміни блока моделі вищого рівня низкою звернень до підпрограм, функцій або процедур, які докладніше відображають цей блок для нижчого рівня. Щоб побудувати таку програму моделювання, потрібно уніфікувати процес передавання параметрів від одного програмного блока до іншого. Це дає змогу організувати взаємодію блоків моделі, що мають різні рівні деталізації, і легко замінювати один блок на інший, більш детально описаний.

У даному випадку програма конструюється ієрархічно - зверху вниз: від головної програми до підпрограм нижнього рівня, причому на кожному рівні використовуються тільки прості послідовності інструкцій, цикли і умовні розгалуження.

Для зручності написання коду програми застосовується декомпозиція – процес поділу систем на елементи, зручні для якихось операцій з нею, а саме поділ до елементів, які приймаються за неподільні об’єкти. Зменшуючи складність системи, забезпечуються умови для аналізу та синтезу компонентів, для проектування, побудови, впровадження, експлуатації та вдосконалення систем управління. Поділ, звичайно, виконують у такий спосіб, щоб компоненти піддавались якій-небудь класифікації.

Верифікація програмного забезпечення — процес посвідчення, що програми та їх компоненти виконують запропоновані їм вимоги. Метою верифікації є

посвідчення в тому, що програмне забезпечення відповідає висунутим вимогам. Паралельно з цим фіксуються нові дефекти, додані в процесі розробки. Процес верифікації є складовою частиною більш загального процес забезпечення домовленого рівня якості розроблюваної системи.

Верифікація націлена на скорочення помилок. Але дуже важливо розуміти, що верифікація - це контрольований ззовні процес, що демонструє наявність у системі багів і умови їх прояву.

Валідація програмного забезпечення — процес визначення відповідності розроблюваного програмного забезпечення між очікуваннями і потребами користувача, вимогам до системи. Валідація є одним із основних етапів тестування програмного забезпечення. Мета процесу валідації — переконатися, що специфічні вимоги для програмного продукту виконано.

Коротка характеристика ПК (ноутбука), який використовувався при виконанні курсового проекту:

— Процесор – Intel Pentium CPU B960 2.2 GHz;

— Оперативна пам’ять – DDR3 2400 MHz, 4Gb;

— Відеокарта – Intel HD Graphics;

— Діагональ екрана – 15.6 дюймів, розширення 1920 × 1080;

3 ПЕРЕЛІК ТА ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ

При запуску програми з’являється головне меню з можливостями виконання завдань даної предметної області, при виборі користувачем одного з яких програма переходить до виконання зазначеної дії. Вибрати пункт меню можна за допомогою клавіш (стрілки вверх і вниз) та натиснення кнопки Enter; для виходу з підменю необхідно вибрати відповідний пункт або натиснути клавішу Esc.

Реалізацію меню зображено на рисунку 3.1

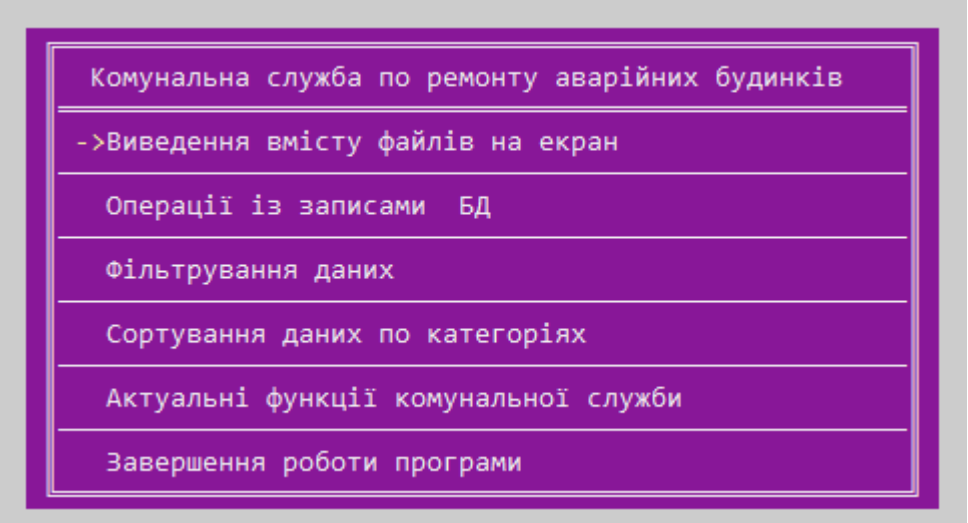


Рисунок 3.1 - Меню програми

Для реалізації пунктів меню застосовуються наступні функції:

1) int output1() – вивід даних з бази даних (початкового текстового файлу);
2) int output2() – вивід даних з бінарного файлу;
3) int record_start() – запис даних на початок файлів;
4) int record_end() – запис даних в кінець файлів;
5) int edit() – редагування даних у файлах (різних полів за бажанням);
6) void del() – видалення записів за порядковим номером;
7) int oblik_current_year() – вивід інформації про будинки, поставлені на облік поточного року;
8) int oblik_year2000() – вивід інформації про будинки, поставлені на облік протягом 2000-х років;

9) int oblik_year_10ago() – вивід інформації про будинки, поставлені на облік понад 10 років тому;

10) int year_created90() – вивід даних про аварійні будинки, споруджені протягом 90-х років;

11) int year_created_1920_1950() – вивід даних про будівлі, споруджені протягом 1920-1950 років;

12) int more_kol_meshk() – вивід інформації про будинки, в яких кількість жителів перевищує середню;

13) int smaller_kol_meshk() – вивід інформації про аварійні будинки, в яких кількість жителів менша, ніж середня;

14) int limit_kol_meshk10(int x) – вивід 10 будинків з найбільшою або найменшою кількістю жителів;

15) int kol_oblik_current() – вивід даних та кількість будинків, які поставлені на облік поточного року;

16) int more_houses_100year() – вивід інформації про аварійні будинки, які через 10 років святкуватимуть ювілей (100+ років);

17) int with_repair() – вивід даних про будинки, що взагалі не потребують ремонту;

18) int oblik_year_recently() – вивід інформації про відремонтовані будинки, які недавно поставлені на облік;

19) int year_created2000() – вивід даних про відремонтовані будинки, споруджені впродовж 2000-х років;

20) int sort_street(int x) – вивід відсортованих даних за назвою вулиці за зростанням або за спаданням;

21) int sort_kol_meshk(int x) – вивід відсортованих даних за кількістю мешканців будинку за спаданням або за зростанням;

22) int sort_year_constructed(int x) – вивід відсортованих даних за роком побудови за зростанням чи спаданням;

23) int sort_date(int x) – вивід даних, відсортованих за датою за зростанням чи спаданням;

24) int sort_repair(int x) – вивід інформації, відсортованої за полем щодо ремонту протягом останніх 20 років залежно від наявності;

25) void del_street() – видалення всіх даних, що стосуються вулиці Пішонівської;

26) int repair() – вивід інформації про аварійні будинки, що потребують негайного ремонту або зносу.

Детальний опис усіх функцій програми:

1) int valid_street(houses *s, int i) – функція, що містить в якості параметрів покажчик *s типу структури houses, а також змінну і цілого типу. Проходячи циклом, відразу перевіряється умова на наявність введених помилкових знаків у полі назви вулиці та рахується кількість таких введень. Якщо кількість не дорівнює 0, то користувач повинен ввести назву вулиці заново, зважаючи на правильність введення.

2) int valid_number(houses* s, int i) - функція, що містить в якості параметрів покажчик *s типу структури houses, а також змінну і цілого типу. Перевіряється правильність введення значення номера будинку, зокрема увага приділяється тому, щоб номер не був менший за 0 або більший 10 000. При некоректному вводі користувач має можливість ввести дані знову.

3) int valid_meshk(houses* s, int i) - функція, що містить в якості параметрів покажчик *s типу структури houses, а також змінну і цілого типу. Перевіряється правильність введення значення кількості мешканців будинку, окрема увага приділяється тому, щоб номер не був менший за 0 або більший 10 000. При некоректному вводі користувач має можливість ввести дані знову.

4) int valid_year(houses* s, int i) - функція, що містить в якості параметрів покажчик *s типу структури houses, а також змінну і цілого типу. Перевіряється правильність введення значення року побудови будинку, зокрема важливо, щоб номер не був менший за 0 або більший 10 000. При некоректному вводі користувач має можливість ввести дані знову.

5) int valid_date(houses* s, int i) - функція, що містить в якості параметрів покажчик *s типу структури houses, а також змінну і цілого типу. Перевіряється

умова правильності введення дати по кожному символу, враховуючи правильність розстановки знаків пунктуації. При неправильному вводі користувачу забезпечено можливість ввести дані знову.

6) int valid_repair(houses *s, int i) - функція, що містить в якості параметрів покажчик *s типу структури houses, а також змінну і цілого типу. Перевіряється умова правильності введення значення щодо наявності ремонту, враховуючи можливість вводу як українською, так і російською. При неправильному вводі користувачу забезпечено можливість ввести дані знову.

7) int valid_n(int n) - функція, що містить в якості параметра змінну n цілого типу. Перевіряється умова правильності введення значення. При неправильному вводі користувачу забезпечено можливість ввести дані знову.

8) void line(int n), void linee(int n), void line1(int n), void line2(int n), void line3(int n) – функції, що в якості параметра містять змінну цілого типу n. Підключено спеціальне кодування 866 для того, щоб була можливість виводити певну кількість різноманітних ліній.

9) void gotoxy(int x, int y) – функція, що встановлює курсор на вказану позицію за допомогою COORD.

10) int main() – основна програма;

11) void menu(), void menu2(), void menu3(), void menu31(), void menu4(), void menu41(), void menu42(), void menu43(), void menu5(), void menu51(), void menu52(), void menu53(), void menu54(), void menu55(), void menu6() – окремі функції меню для повної взаємодії з програмою та підтримання головного меню програми.

12) void menuK(), void menuK2(), void menuK3(), void menuK31(), void menuK4(), void menuK41(), void menuK42(), void menuK43(), void menu5(), void menuK51(), void menuK52(), void menuK53(), void menuK54(), void menuK55(), void menuK6() – окремі функції, що викликаються у функціях меню для повної взаємодії з програмою та підтримання головного меню програми, а також для визначення необхідних позицій для стрілки керування у програмі.

13) void choice(), void choice2(), void choice3(), void choice31(), void

choice4(), void choice41(), void choice42(), void choice43(), void choice5(), void choice51(), void choice52(), void choice53(), void choice54(), void choice55(), void choice6() - функції, пов’язані з menu, які визначають, яку клавішу натиснуто, після чого виконується перехід до інших функцій для забезпечення цілісності програми.

14) void go(int b), void go2(int b), void go3(int b), void go31(int b), void go4(int b), void go41(int b), void go42(int b), void go43(int b), void go5(int b), void go51(int b), void go52(int b), void go53(int b), void go54(int b), void go55(int b), void go6(int b) – функції, що приймають в якості параметра змінну цілого типу b, та здійснюють виклик функцій, які вибрано в menu за допомогою choice.

15) void SetColor(int text, int background) – функція, що встановлює колір тексту та колір фону в консолі;

16) void fullConsole() – функція, що автоматично при відладці відкриває консоль в розширеному режимі.

17) void output_struct_txt(FILE* newf, houses h[], int i) – функція, яка в якості параметрів містить покажчик на змінну типу FILE, масив h типу структури houses, змінну цілого типу i. За допомогою цієї функції реалізовується вивід структури у вихідний текстовий файл.

18) void output_struct_console(houses h[], int i) - функція, яка в якості параметрів містить покажчик на змінну типу FILE, масив h типу структури houses, змінну цілого типу i. За допомогою цієї функції реалізовується вивід структури у консоль, використовуючи оператори для задання довжини та розміщення даних певного поля.

19) void shapka() – функція, що виводить шапку в консолі, використовуючи при цьому обрамлення різними лініями.

20) void shapka_txt(FILE* newf, int n) - функція, яка в якості параметрів містить покажчик на змінну типу FILE та змінну цілого типу i. Функція реалізовує виведення шапки у вихідний текстовий файл.

21) int kilkzap() – функція, яка повертає як результат кількість записів для вводу, здійснюючи також валідацію даних.

22) int output1(), int output2() – функції, які реалізовують вивід на екран

даних з бази даних, тобто текстового файлу, а також з бінарного файлу. Прямо у функції відкриваються та закриваються файли, з яких необхідно зчитати дані, а також за допомогою циклів здійснюється зчитування та вивід на екран відповідних даних.

23) int record_start() – функція, яка забезпечує можливість для користувача вводити дані та записувати їх на початок файлів. Реалізовано все завдяки методу перезапису, щоб усунути проблеми зі зсуванням рядків одні на одних. Записи для додавання записуються додатково у вихідний текстовий файл.

24) int record_end() – функція, яка реалізовує ввід даних та запис їх в кінець файлів, використовуючи при цьому функцію переміщення fseek з кінцевим параметром SEEK_END. Відкриття та закриття файлів відбувається прямо у функції.

25) int edit() – функція, яка забезпечує можливість редагування даних. Безпосередньо у функції відбувається відкриття та закриття файлів. Поміщений «вічний» цикл, який дозволяє редагувати не один запис, а необхідну кількість. При вводі нових даних здійснюється валідація по кожному полю.

26) void del() – функція, яка надає користувачу можливість видалити запис за порядковим номером. Спочатку виводяться всі дані з файлу, користувач обирає запис, що підлягає видаленню і, відповідно до номера того запису і після нього, не включаючи його самого, здійснюється переписування даних заново у файли.

27) int more_kol_meshk() – функція, що забезпечує вивід даних користувачу про аварійні будинки, в яких кількість мешканців більша за середню. Організована функція таким чином, що спочатку в циклі при зчитуванні даних з файлу підраховується сума кількості жителів у будинку, а далі відповідно до кількості обчислюється середнє значення та виводиться на екран разом із даними про будинки, що задовольняють умову. Запис паралельно ведеться і у вихідному текстовому файлі.

28) int smaller_kol_meshk() - функція, що забезпечує вивід даних користувачу про аварійні будинки, в яких кількість мешканців менша, ніж середня. Організована функція таким чином, що спочатку в циклі при зчитуванні даних з файлу підраховується сума кількості жителів у будинку, а далі відповідно до кількості

обчислюється середнє значення та виводиться на екран разом із даними про будинки, що задовольняють умову. Запис паралельно ведеться і у вихідному текстовому файлі.

29) int more_houses_100year() – «ювілейна» функція, яка надає користувачу можливість дізнатися, яким аварійним будинкам від дня спорудження виповниться через 10 років 100+ років. Як і в інших функціях, відкриття та закриття файлів відбувається всередині, при зчитуванні паралельно здійснюється підрахунок кількості таких будинків відповідно до умови. Унікальність полягає в тому, що у функції використовуються додаткові функції для визначення поточного часу, використовуючи бібліотеку #include <time.h>. Для виконання операцій використовується бінарний файл, а вивід здійснюється не тільки на консоль, але й паралельно у вихідний текстовий файл.

30) int oblik_current_year() – функція, що виводить інформацію про аварійні будинки, поставлені на облік поточного року. Відкриття та закриття файлів відбувається всередині функції, при зчитуванні паралельно здійснюється підрахунок кількості таких будинків відповідно до умови. У функції використовуються додаткові функції для визначення поточного року, використовуючи бібліотеку #include <time.h>. Для виконання операцій використовується бінарний файл, а вивід здійснюється не тільки на консоль, але й паралельно у вихідний текстовий файл.

31) int oblik_year2000() – функція, яка забезпечує вивід на екран користувачу даних про аварійні будинки, поставлені на облік протягом 2000-х років. Унікальність полягає у визначенні року серед всієї дати, яка записується у символьний масив розмірністю 11. При цьому враховується порядок розміщення додаткових символів (крапок) для одержання точного результату без похибок. Стандартно, як і в інших попередніх функціях, спочатку здійснюється зчитування даних з бінарного файлу, підраховується кількість записів, які задовольняють умову, а далі залежно від кількості виводиться результат на екран: повідомлення про те, що таких будинків немає або ж вивід усіх будинків, що відповідають поставленій умові.

32) int oblik_year_recently() – функція, яка виводить на вимогу користувача

дані про аварійні будинки, які недавно поставлені на облік, а саме починаючи з 2010 року включно. Перевірка здійснюється, як і в усіх попередніх схожих функціях, в яких було реалізовано правильну та коректну перевірку по даті згідно з поставленою умовою. Варто виокремити те, що для забезпечення правильного виконання функції, операції здійснюються, використовуючи динамічний масив.

33) int year_created90(), int year_created2000(), int year_created_1920_1950() – функції, що виконують схожий алгоритм дій відповідно до заданої певної умови. Як і в інших функціях, відкриття та закриття файлів здійснюється всередині, далі після зчитування даних з бінарного файлу підраховується кількість записів, які задовольняють відповідну умову, а потім залежно від кількості виводиться повідомлення або список аварійних будинків, що задовольняють умову.

34) int repair_yes(), int with_repair(), int repair() – функції, які діють за спільним алгоритмом відповідно до вказаних умов. Функція int repair_yes() надає користувачу можливість дізнатися, в яких будинках здійснювався ремонт за останні 20 років. Функція int with_repair() надає достовірну інформацію про аварійні будинки, які взагалі не потребують ремонту. Тобто умова покладена таким чином: здійснюється пошук будинків, в яких наявний ремонт протягом останніх 20 років та які поставлені на облік порівняно недавно. Функція int repair() реалізована згідно із варіантом і суть її полягає в тому, що вона допомагає здійснити пошук будинків, які потребують негайного ремонту або взагалі зносу. Тобто це будинки, кількість мешканців в яких більша, ніж 50, дата побудови менша 1941 року, а ремонт протягом останніх 20 років не проводився. Пошук у функціях відбувається стандартним методом. Спочатку відкривається файл, з якого зчитуються дані. При зчитуванні одночасно рахується кількість записів, що відповідають певній умові. Відповідно до кількості таких записів далі здійснюється або вивід повідомлення про те, що у базі даних таких будинків немає, або відбувається вивід на екран списку вулиць і будинків, які задовольняють умові. Після отриманого результату звільняється динамічна пам’ять, якщо вона була виділена, а також відбувається закриття бінарного файлу і вихідного текстового файлу, куди також був здійснений паралельний запис.

35) int limit_kol_meshk10(int x) – функція, яка виводить на екран обмежену кількість даних щодо кількості мешканців залежно від вибору користувачем умови. Якщо вхідний параметр функції дорівнюватиме нулю, то буде здійснено вивід на екран 10 записів з найбільшою кількістю жителів у аварійних будинках, а якщо 1 – то виведуться записи з найменшою кількістю жителів. Реалізована функція за допомогою гілок умов та сортування даних за методом «бульбашки». Паралельно вивід здійснюється у вихідний текстовий файл.

36) int kol_oblik_current() - функція, що виводить інформацію про кількість аварійних будинків, поставлених на облік поточного року, та їхній список. Відкриття та закриття файлів відбувається всередині функції, при зчитуванні паралельно здійснюється підрахунок кількості таких будинків відповідно до умови. У функції використовуються додаткові функції для визначення поточного року, використовуючи бібліотеку #include <time.h>. Для виконання операцій використовується бінарний файл, а вивід здійснюється не тільки на консоль, але й паралельно у вихідний текстовий файл.

37) int sort_street(int x), int sort_date(int x), int sort_repair(int x) – функції, що виводять на екран користувачу записи відсортовані за зростанням чи спаданням залежно від вибору. У наведених функціях здійснюється сортування полів символьного типу, використовуючи стандартний метод сортування «бульбашкою». Операції здійснюються, використовуючи динамічний масив, де після виконання функції він звільняється.

38) int sort_kol_meshk(int x), int sort_year_constructed(int x) – функції, що виводять на консоль записи відсортовані за зростанням чи спаданням залежно від вибору користувача. У наведених функціях здійснюється сортування полів цілочисельного типу, використовуючи стандартний метод сортування «бульбашкою». Операції здійснюються, використовуючи динамічний масив, де після виконання функції він звільняється. Варто відзначити те, що вивід відсортованих даних здійснюється лише на екран, паралельного виводу у вихідний текстовий файл не реалізовано з метою економного використання пам’яті.

39) void del_street() – функція, що забезпечує можливість користувачу

видалити всі записи, пов’язані із вулицею Пішонівською згідно із варіантом. Функція реалізована таким чином, що спочатку виводяться всі записи згідно з умовою, здійснюється вивід інформації про кількість таких записів і надається можливість повернутися назад і не видаляти записи, проте якщо користувач справді бажає здійснити видалення, то його бажання задовільниться.

40) int oblik_year_10ago() - функція, що забезпечує можливість користувачу побачити дані щодо аварійних будинків, поставлених на облік понад 10 років тому, згідно із варіантом. Функція реалізована таким чином, що спочатку зчитуються дані з бінарного файлу, перевіряється умова, а далі залежно від кількості записів, що задовольняють поставлену умову, здійснюється подальше виконання програмного коду.

41) void cls(short x, short y, int n) – функція, що здійснює перехід по координатах та здійснює очищення вказаної ділянки згідно з координатами.

4 ХАРАКТЕРИСТИКА ТА СТРУКТУРА ДАНИХ, З ЯКИМИ ПРАЦЮЄ ПРОГРАМА

Підготовлені дані, що містять інформацію про аварійні будинки районів міста. Структура запису є наступною:

- Назва вулиці,
- Номер будинку,
- Кількість жителів у будинку,
- Дата спорудження,
- Дата постановки на облік,
- Наявність ремонту за останні 20 років (так/ні).

street – назва вулиці (текстове поле довжиною 30 символів);

number_house – номер будинку (цілочисельне поле);

kol_meshk – кількість жителів у будинку (цілочисельне поле);

year_construct – рік побудови (числове поле цілого типу);

date – дата постановки на облік (текстове поле довжиною 11 символів);

repair – наявність ремонту за останні 20 років (так/ні) (текстове поле довжиною 4 символи).

Вивчивши детальніше постановку завдання, була створена відповідно наступна структура:

```
struct houses
{
        char street[30];
        int number_house;
        int kol_meshk;
        int year_construct;
        char date[11];
        char repair[4];
};
```

Створена структура має назву houses(будинки), в якій розміщені наступні поля:

— char street[30]; - масив з назвою street символьного типу з максимальною кількістю елементів 30. В цьому полі міститься вся інформація, що стосується вулиць, тобто їхні назви в районі. За максимальну кількість взято число 30, тому що невідома точна назва кожної вулиці в районі, яка може попасти в список шляхом додавання нових записів користувачем;

— int number_house; - поле з назвою number_house цілого типу, яке зберігає значення номера аварійного будинку, який занесений в список;

— int kol_meshk; - поле з назвою kol_meshk цілого типу, яке зберігає дані щодо кількості жителів будинку;

— int year_construct; - поле з назвою year_construct цілого типу, в якому зберігаються дані щодо року побудови аварійного будинку, який занесений в список;

— char date[11]; - масив з назвою date символьного типу, який зберігає всю інформацію про дату постановки будинку на облік. Формат дати наступний: ДД.ММ.РРРР, де ДД – день постановки, ММ – місяць постановки, РРРР – рік постановки.

char repair[4]; - символьний масив з назвою repair з розмірністю 4, який використовується для інформації щодо проведення ремонту в аварійних будинках на певній вулиці району міста впродовж останніх 20 років.

Для зручності використання кольорів у програмі та й загалом у псевдографіці оголошено enum ConsoleColor, де об’єднані всі кольори, які можна зручно використовувати.

```
enum ConsoleColor
{
    Black = 0,
    Blue = 1,
    Green = 2,
    Cyan = 3,
    Red = 4,
    Magenta = 5,
```

```
        Brown = 6,
        LightGray = 7,
        DarkGray = 8,
        LightBlue = 9,
        LightGreen = 10,
        LightCyan = 11,
        LightRed = 12,
        LightMagenta = 13,
        Yellow = 14,
        White = 15
};
```

Глобальні змінні у програмі не використовуються з декількох причин, найосновнішими з яких є те, що відбувається захаращення простору імен, неявне зчеплення, повсюдність та стоїть питання конкурентності. У невеликих програмах використання глобальних змінних можливе, проте все-таки це вважається поганим тоном у програмуванні.

Створена програма працює з трьома файлами: текстовий файл streets.txt (поточна база даних), бінарний файл street.dat (для зручності при здійсненні певних операцій) та вихідний текстовий файл str_operations.txt (формується при обранні користувачем та виконанні певних функцій).

Сформований уже наявний текстовий файл, який містить базу даних, спочатку при запуску програми перезаписується у двійковий для подальшої роботи. На початку цього файлу записана розмірність, яка впродовж виконання дій протягом всієї роботи програми змінюється. Крім функцій, які розроблено відповідно до варіанту, реалізовано ще багато інших, які також необхідні для комунальної служби аварійних будинків району міста. Валідація даних здійснюється по всіх полях розробленої структури. Таким чином перевіряється правильність введення даних, в разі неправильності видається відповідне повідомлення і забезпечено можливість додаткового вводу в такому випадку.

Вихідним є сформований текстовий файл, де записані проведені операції над даними. Паралельно здійснюється вивід на екран у вигляді таблиць за бажанням користувача.

5 ОПИС ПРОГРАМИ

5.1 Блок-схема алгоритму та його опис

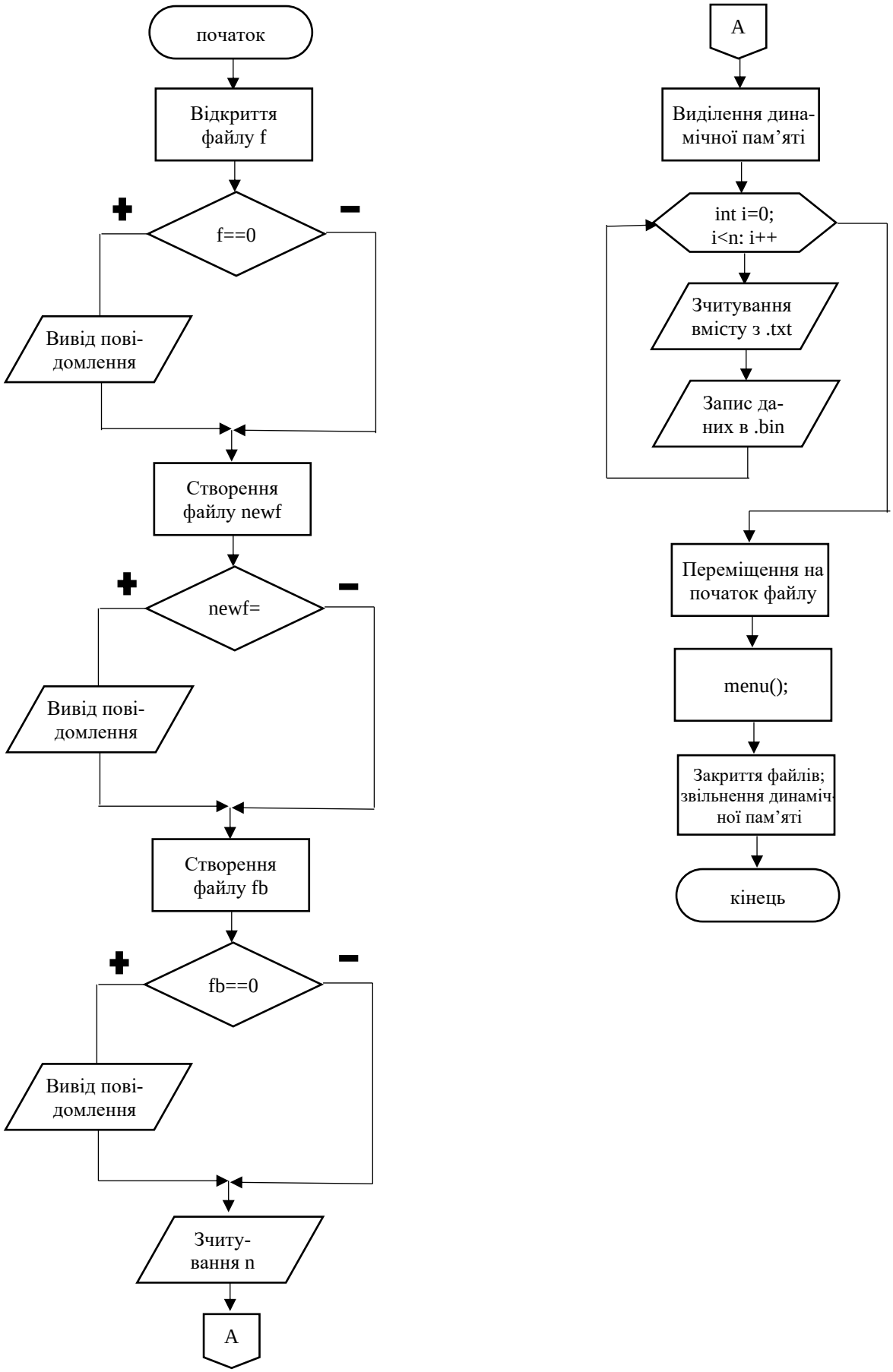


Рисунок 5.1 – Блок-схема функції main()

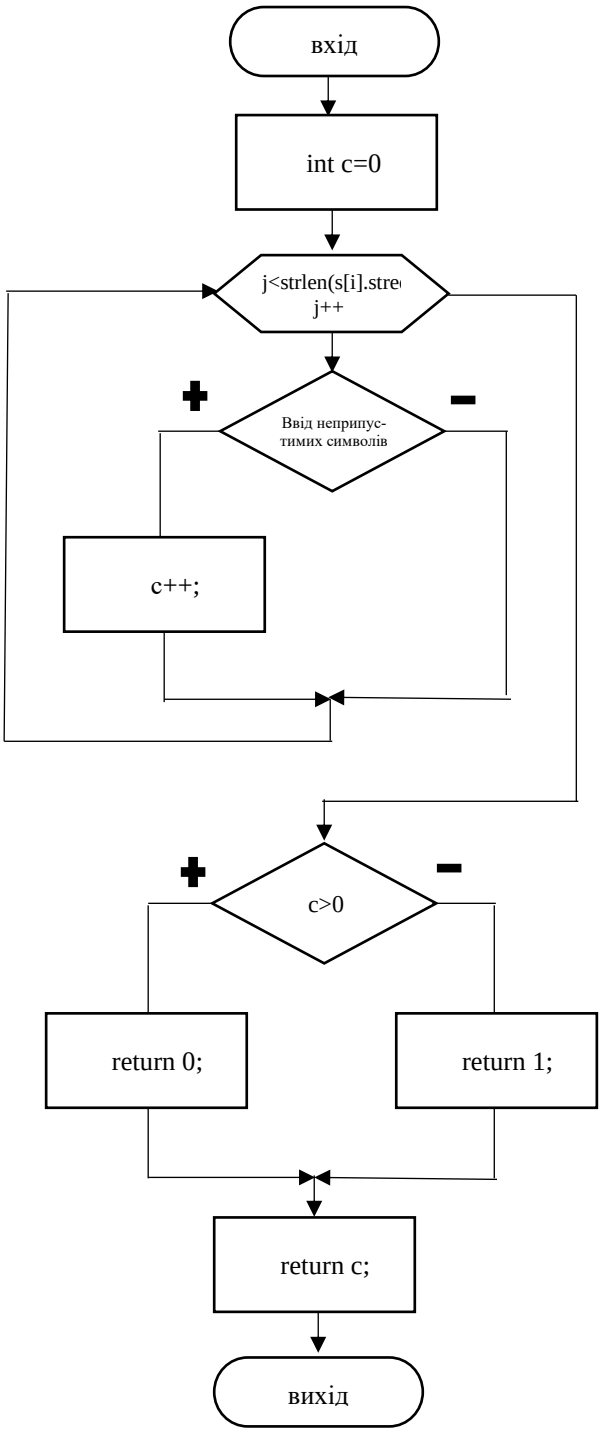


Рисунок 5.2 – Блок-схема функції valid_street(houses *s, int i)

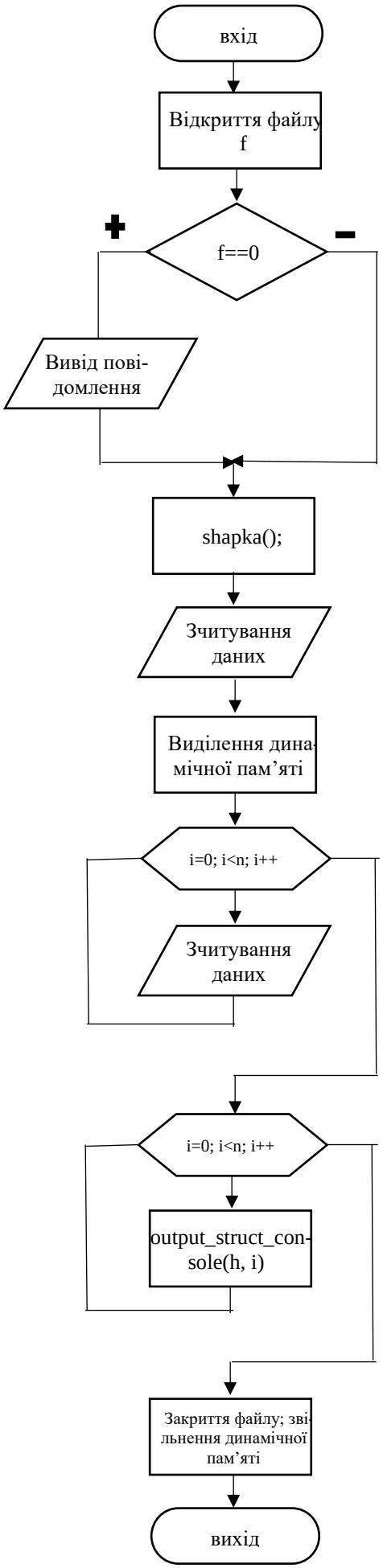


Рисунок 5.3 – Блок-схема функції output1()

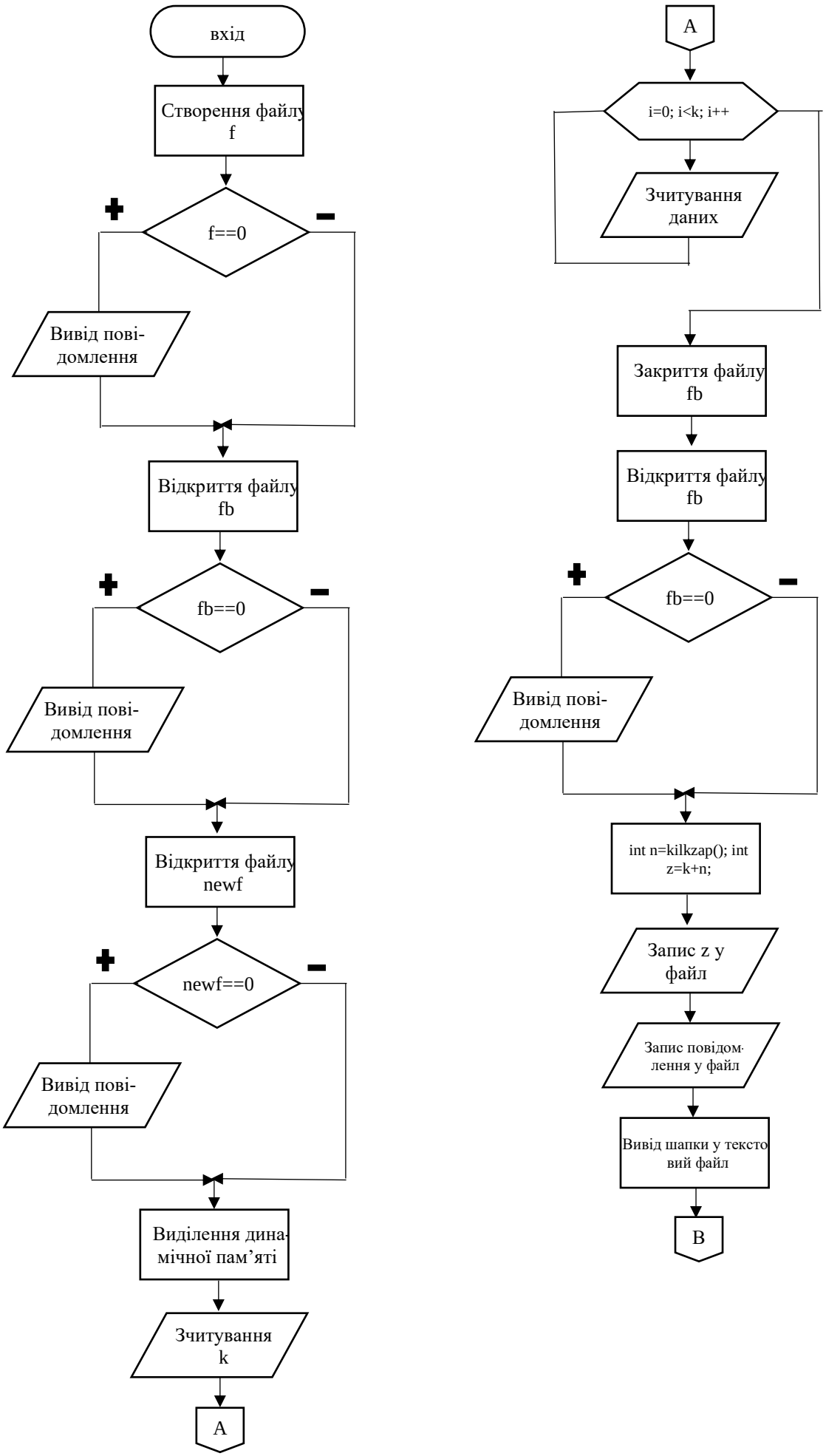
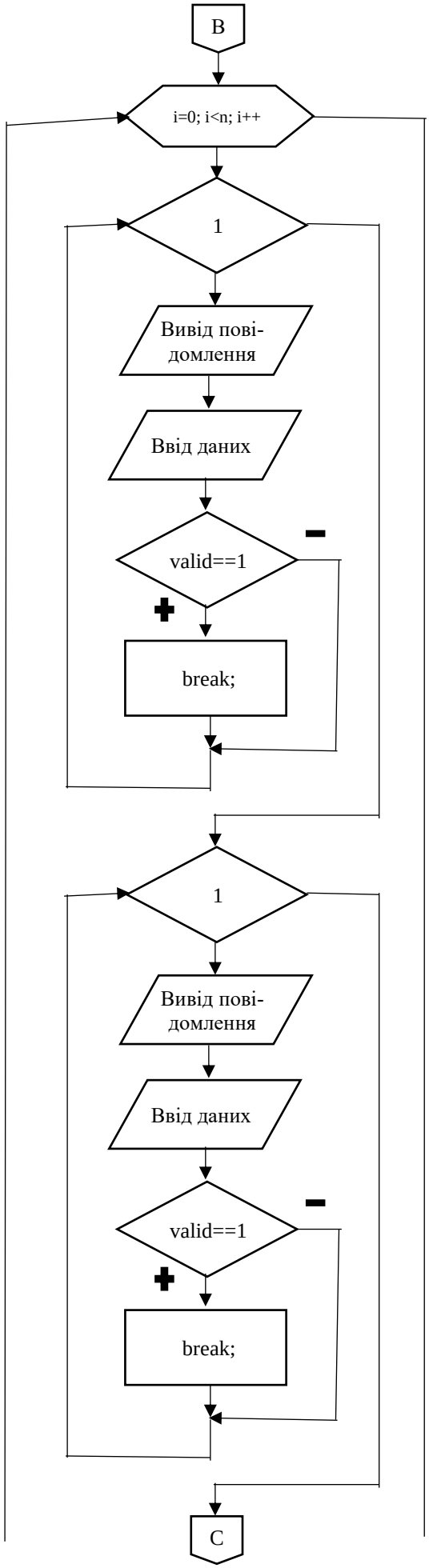


Рисунок 5.4 – Блок-схема функції record_start()



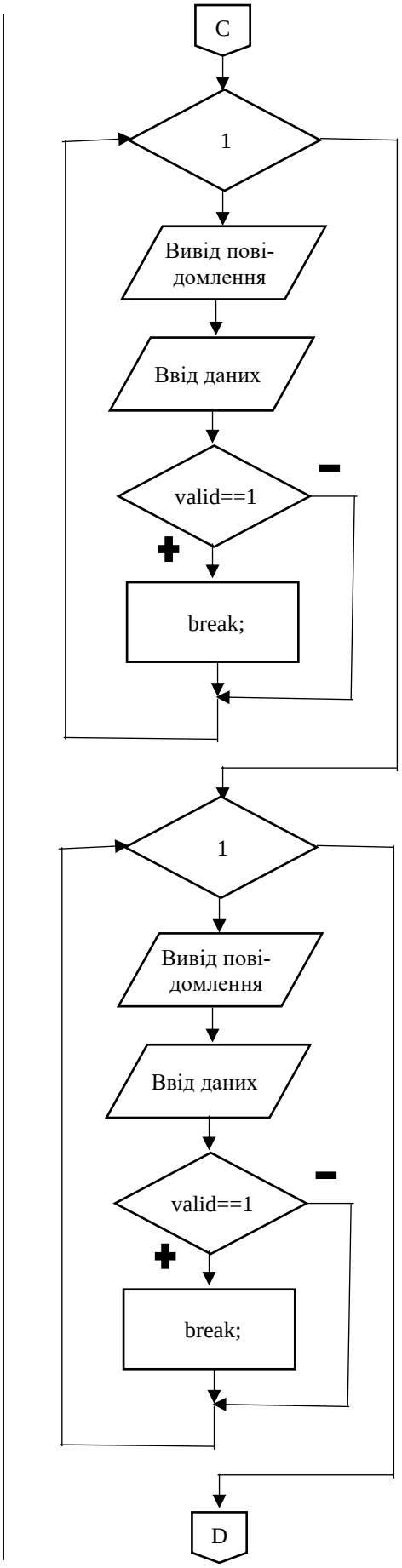


Рисунок 5.5 – Продовження блок-схеми функції record_start()

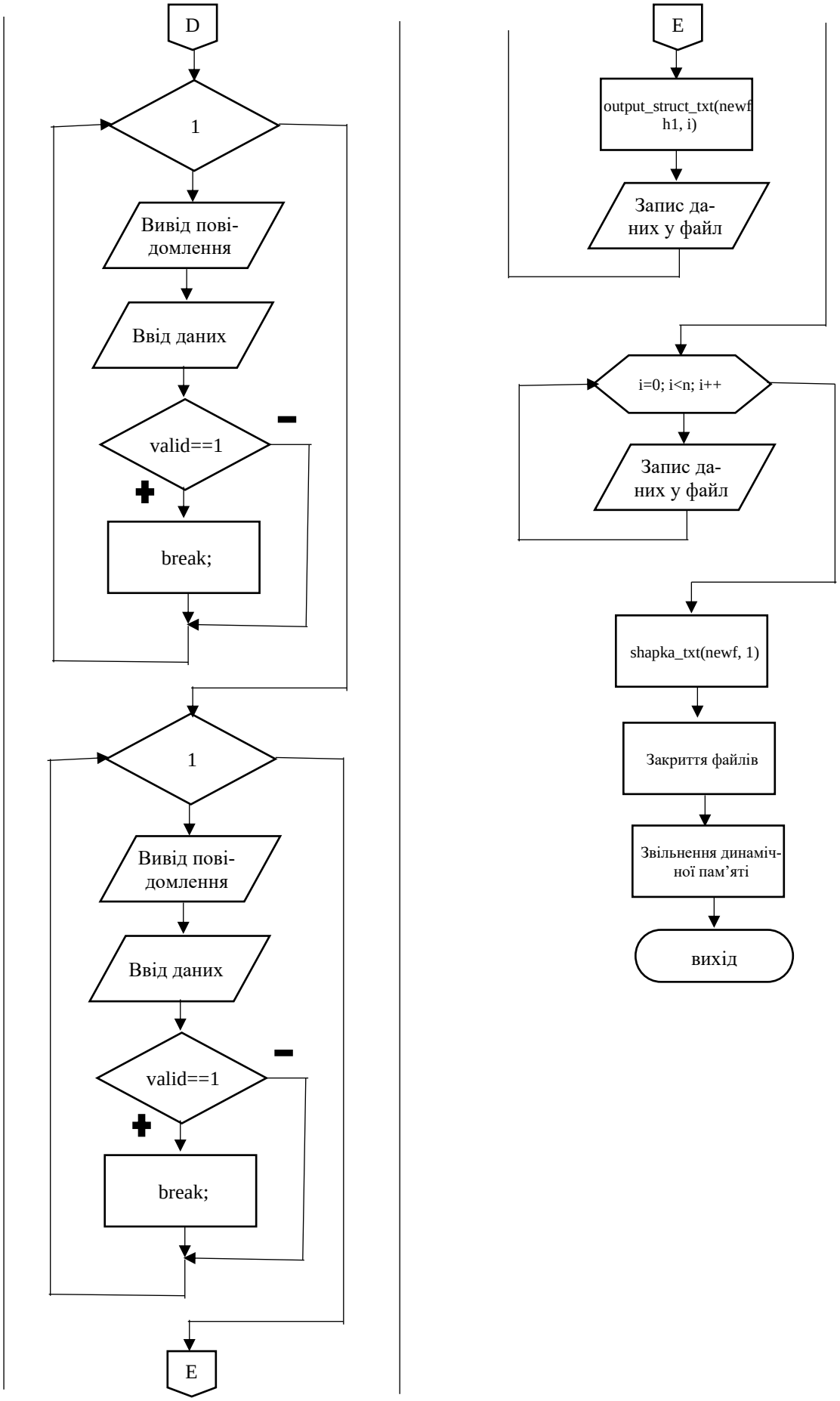


Рисунок 5.6 – Продовження блок-схеми функції record_start()

5.2 Таблиця ідентифікації

В програмі	В завданні
street	Поле запису, символьного типу, назва вулиці
number_house	Поле запису, цілочисельного типу, номер будинку
kol_meshk	Поле запису, цілочисельного типу, кількість мешканців
year_construct	Поле запису, цілочисельного типу, рік побудови будинку
date	Поле запису, символьного типу, дата постановки на облік
repair	Поле запису, символьного типу, наявність ремонту за останні 20 років
int main()	Головна функція
void line(int n)	Функція виводу ліній, використовуючи елементи кодування 866
void linee(int n)	Функція виводу ліній, використовуючи елементи кодування 866
void line1(int n), void line2(int n), void line3(int n)	Функції виводу ліній, що використовує елементи кодування 866
int valid_street(houses *s, int i)	Функція валідації даних поля вулиці
int valid_number(houses* s, int i)	Функція валідації даних поля номера будинку
int valid_meshk(houses* s, int i)	Функція валідації даних поля кількості мешканців
int valid_year(houses* s, int i)	Функція валідації даних поля року спорудження аварійного будинку

int valid_date(houses* s, int i)	Функція валідації даних поля дати постановки на облік
int valid_repair(houses *s, int i)	Функція валідації даних поля наявності ремонту за останні 20 років
int valid_n(int n)	Функція валідації даних значення змінної
void gotoxy(int x, int y)	Функція визначення позиції для можливості керування клавішами
void fullConsole()	Функція для відкриття консолі в розширеному режимі
void output_struct_txt(FILE* newf, houses h[], int i)	Функція, що виводить структуру у вихідний текстовий файл
void output_struct_console (houses h[], int i)	Функція, яка виводить структуру на екран
int edit()	Функція, яка дозволяє редагувати записи
int kilkzap()	Функція, яка повертає значення кількості записів
void shapka()	Функція, яка виводить шапку в консоль
void shapka_txt(FILE* newf, int n)	Функція, яка виводить шапку у вихідний текстовий файл
int output1()	Функція, яка виводить дані з текстового файлу, що зберігає базу даних
int output2()	Функція, яка виводить вміст бінарного файлу
int record_start()	Функція, яка дозволяє записувати дані на початок файлів
int record_end();	Функція, яка дозволяє вносити записи в кінець файлів

int sort_street(int x);
int sort_date(int x);
int sort_repair(int x);
int sort_year_constructed(int x);
int sort_kol_meshk(int x)
	Функції, призначені для сортування даних по різних полях за зростанням чи спаданням

void del_street()	Функція, яка видаляє всі дані, пов’язані з вулицею Пішонівською
void del()	Функція, що забезпечує можливість видалення записів за їхнім порядковим номером
int oblik_current_year()	Функція, що виводить дані про будинки, поставлені на облік поточного року
int oblik_year2000()	Функція, що виводить дані про аварійні будинки, поставлені на облік протягом 2000-х років
int oblik_year_10ago()	Функція, яка знаходить записи про будинки, поставлені на облік понад 10 років тому
int oblik_year_recently()	Функція, що виводить дані про будинки, поставлені на облік порівняно недавно (починаючи від 2010 року включно)
int repair_yes()	Функція, що виводить дані про будинки, в яких проводився ремонт протягом останніх 20 років
int with_repair()	Функція, що надає інформацію про аварійні будинки, які взагалі не потребують ремонту
int repair()	Функція, яка виводить дані про будинки, які потребують термінового ремонту або взагалі повного зносу
int more_kol_meshk()	Вивід даних про будинки, в яких кількість мешканців більша за середню
int smaller_kol_meshk()	Вивід даних про будинки, в яких кількість мешканців менша за середню
int more_houses_100year()	Функція, яка надає користувачу можливість дізнатися, яким аварійним будинкам від дня побудови виповниться через 10 років більше, ніж 100 років

int year_created90();
int year_created_1920_1950();
int year_created2000()
	Функція, яка виводить дані про будинки, споруджені протягом певного періоду залежно від умови

int limit_kol_meshk10(int x)	Функція, яка виводить лімітовану кількість даних залежно від кількості жителів у будинку за зростанням або спаданням
int kol_oblik_current()	Функція, яка виводить дані про будинки, поставлені на облік поточного року, та рахує їхню кількість
void menuK(), void menuK2(), void menuK3(), void menuK31(), void menuK4(), void menuK41(), void menuK42(), void menuK43(), void menu5(), void menuK51(), void menuK52(), void menuK53(), void menuK54(), void menuK55(), void menuK6()	Функції, що викликаються у функціях меню для повної взаємодії з програмою та підтримання головного меню програми, а також для визначення необхідних позицій для стрілки керування у програмі
void choice(), void choice2(), void choice3(), void choice31(), void choice4(), void choice41(), void choice42(), void choice43(), void choice5(), void choice51(), void choice52(), void choice53(), void choice54(), void choice55(), void choice6()	Функції, пов’язані з menu, які визначають, яку клавішу натиснуто, після чого виконується перехід до інших функцій для забезпечення цілісності програми

5.3 Текст програми та її опис

Вміст файлу main.cpp:

```
// Косак Аліна, ПІ-18-01
// Балабаник О.К.
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "Header.h"
#include "Windows.h"

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	FILE *f, *fb, *newf;
	int n;
	fullConsole();
	f = fopen("streets.txt", "r");
	if (f == NULL)
	{
		cout << "Текстовий файл не відкрито!" << endl;
		cin.get(); // затримка екрану
		exit (1);
	}
	newf = fopen("str_operations.txt", "w+");
	if (newf == NULL)
	{
		cout << "Текстовий файл не створено!" << endl;
		cin.get(); // затримка екрану
		exit(1);
	}
	fb = fopen("street.dat", "w+b");
	if (fb == NULL)
	{
		cout << "Бінарний файл не створено!" << endl;
		cin.get(); // затримка екрану
		exit(1);
	}
	fscanf(f, "%i ", &n);
	fwrite(&n, sizeof(int), 1, fb);
	houses* h = new houses[n];
	for(int i=0; i<n; i++)
	{
		fscanf(f, "%s", &h[i].street);
		fwrite(&h[i].street, sizeof(h[i].street), 1, fb);
		fscanf(f, "%i", &h[i].number_house);
		fwrite(&h[i].number_house, sizeof(h[i].number_house), 1, fb);
		fscanf(f, "%i", &h[i].kol_meshk);
		fwrite(&h[i].kol_meshk, sizeof(h[i].kol_meshk), 1, fb);
		fscanf(f, "%i ", &h[i].year_construct);
		fwrite(&h[i].year_construct, sizeof(h[i].year_construct), 1, fb);
		fscanf(f, "%s ", &h[i].date);
		fwrite(&h[i].date, sizeof(h[i].date), 1, fb);
		fscanf(f, "%s", &h[i].repair);
		fwrite(&h[i].repair, sizeof(h[i].repair), 1, fb);
	}
	fseek(fb, 0, SEEK_SET);
	menu();
	fclose(f);
```

```
	fclose(fb);
	fclose(newf);
	delete[]h;
	system("pause");
	return 0;
}
```

Вміст файлу Header.h:

```
// Косак Аліна, ПІ-18-01
// Балабаник О.К.
#ifndef HeaderH
#define HeaderH
#include <iostream>
#include <conio.h>
#include <iomanip>
#include <time.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <Windows.h>
using namespace std;

struct houses
{
	char street[30];
	int number_house;
	int kol_meshk;
	int year_construct;
	char date[11];
	char repair[4];
};

void menu();
void fullConsole();
void output_struct_txt(FILE* newf, houses h[], int i);
void output_struct_console(houses h[], int i);
int valid_street(houses *s, int i);
int valid_number(houses *s, int i);
int valid_date(houses *s, int i);
int valid_meshk(houses *s, int i);
int valid_year(houses *s, int i);
int valid_repair(houses *s, int i);
int valid_n(int n);
int edit();
int repair();
int with_repair();
int repair_yes();
int oblik_current_year();
int oblik_year2000();
int oblik_year_10ago();
int oblik_year_recently();
int more_kol_meshk();
int smaller_kol_meshk();
int more_houses_100year();
int year_created90();
int year_created_1920_1950();
int year_created2000();
int limit_kol_meshk10(int x);
int kol_oblik_current();
int kilkzap();
void shapka();
void shapka_txt(FILE* newf, int n);
```

```
int output1();
int output2();
int record_start();
int record_end();
int sort_street(int x);
int sort_date(int x);
int sort_repair(int x);
int sort_year_constructed(int x);
int sort_kol_meshk(int x);
void del();
void del_street();
void line(int n);
void linee(int n);
void line1(int n);
void line2(int n);
void line3(int n);
void gotoxy(int x, int y);
#endif
```

Вміст інших файлів наведено у додатку А.

5.4 Контрольний приклад та результат його виконання на ПК

Скріншот головного меню зображено на рисунку 5.7

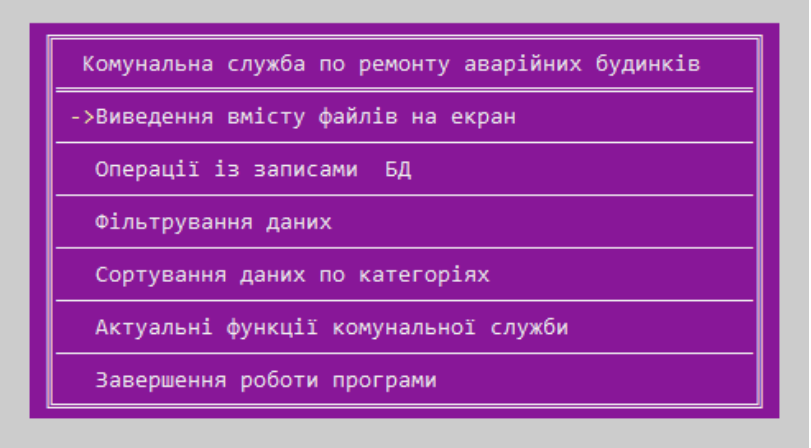


Рисунок 5.7 - Меню програми

Скріншот виведення даних з бази даних, записаної в текстовий файл зображено на рисунку 5.8

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
1	Микитинецька	36	81	1937	29.01.1998	так
2	Тисменицька	82	18	1926	20.02.1972	так
3	Вовчинецька	219	98	1991	20.01.2011	ні
4	Пішонівська	21	82	2002	20.03.2020	ні
5	Бельведерська	27	20	1999	02.09.2018	ні
6	Василіянок	213	52	1927	18.12.1989	так
7	Дністровська	211	89	1992	20.09.2018	ні
8	Калуська	18	36	2017	11.11.2020	ні
9	Незалежності	111	92	1971	19.11.2000	так
10	Автоливмашівська	219	82	1961	20.02.1992	так
11	Франка	19	20	1992	12.09.2011	ні
12	Карпатська	219	81	1992	20.04.2009	ні
13	Пішонівська	241	19	1990	29.09.2015	ні
14	Молодіжна	23	19	2017	12.01.2020	ні
15	Чорновола	78	21	1997	20.10.2015	ні
16	Галицька	72	18	2013	22.11.2020	ні
17	Пішонівська	219	25	1943	12.09.2001	ні
18	Коновальця	89	62	1999	12.01.2019	ні
19	Микитинецька	112	27	1997	29.09.2020	так
20	Дністровська	271	72	1923	27.01.2018	так
21	Незалежності	12	57	1968	21.03.1993	так
22	Сагайдачного	178	89	2009	16.03.2018	ні
23	Симоненка	26	37	1975	27.07.2003	так

Рисунок 5.8 – Вміст текстового файлу

Скріншот виведення даних з бінарного файлу зображено на рисунку 5.9

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
1	Микитинецька	36	81	1937	29.01.1998	так
2	Тисменицька	82	18	1926	20.02.1972	так
3	Вовчинецька	219	98	1991	20.01.2011	ні
4	Пішонівська	21	82	2002	20.03.2020	ні
5	Бельведерська	27	20	1999	02.09.2018	ні
6	Василіянок	213	52	1927	18.12.1989	так
7	Дністровська	211	89	1992	20.09.2018	ні
8	Калуська	18	36	2017	11.11.2020	ні
9	Незалежності	111	92	1971	19.11.2000	так
10	Автоливмашівська	219	82	1961	20.02.1992	так
11	Франка	19	20	1992	12.09.2011	ні
12	Карпатська	219	81	1992	20.04.2009	ні
13	Пішонівська	241	19	1990	29.09.2015	ні
14	Молодіжна	23	19	2017	12.01.2020	ні
15	Чорновола	78	21	1997	20.10.2015	ні
16	Галицька	72	18	2013	22.11.2020	ні
17	Пішонівська	219	25	1943	12.09.2001	ні
18	Коновальця	89	62	1999	12.01.2019	ні
19	Микитинецька	112	27	1997	29.09.2020	так
20	Дністровська	271	72	1923	27.01.2018	так
21	Незалежності	12	57	1968	21.03.1993	так
22	Сагайдачного	178	89	2009	16.03.2018	ні
23	Симоненка	26	37	1975	27.07.2003	так

Рисунок 5.9 – Вміст бінарного файлу

Скріншот меню для введення нових записів зображено на рисунку 5.10

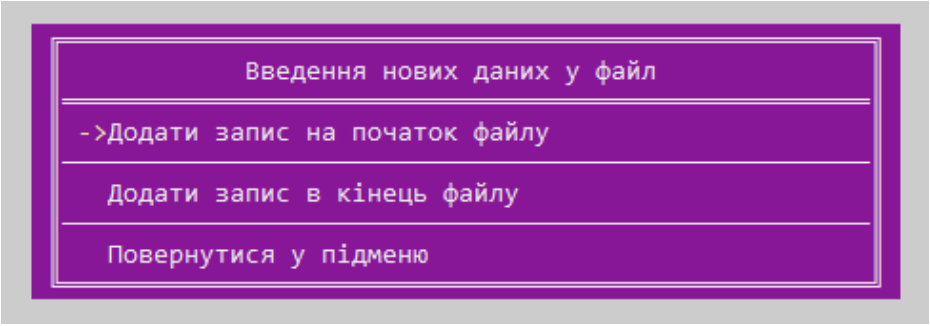


Рисунок 5.10 – Меню для введення запису у базу даних

Скріншот алгоритму введення нових записів зображено на рисунку 5.11

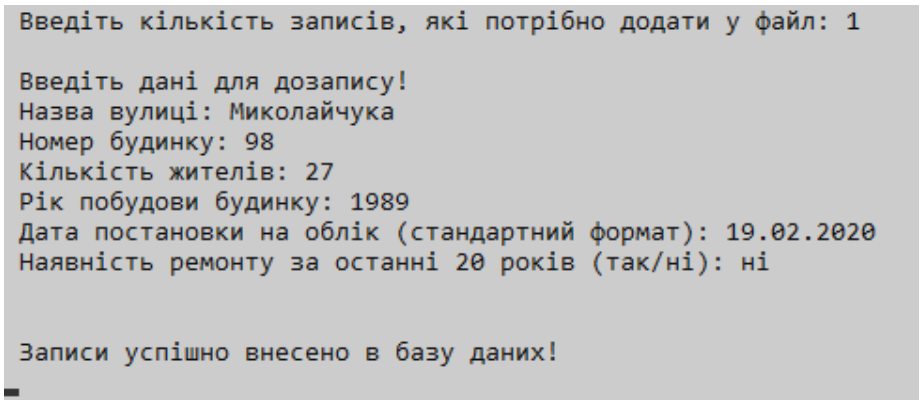


Рисунок 5.11 – Алгоритм введення нового запису у базу даних

Скріншот алгоритму редагування записів зображено на рисунку 5.12

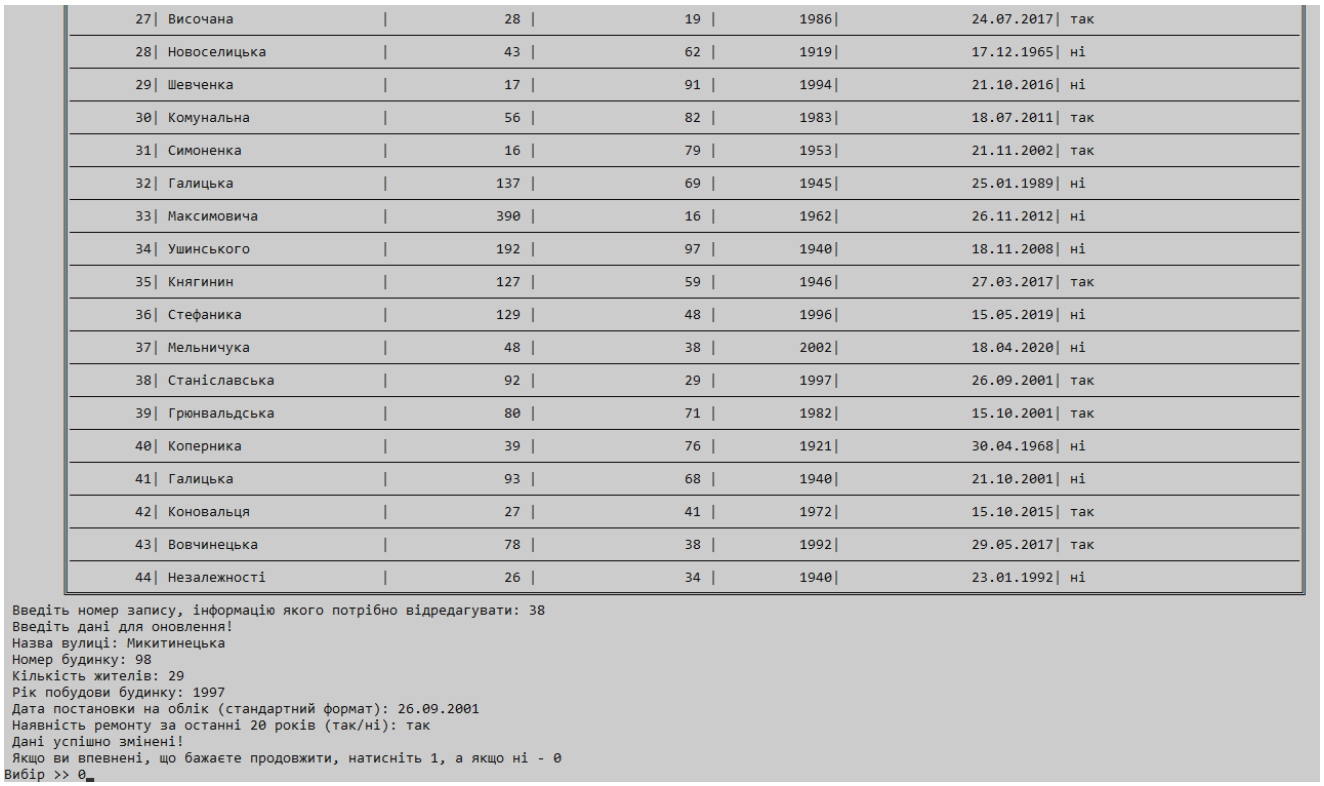


Рисунок 5.12 – Видалення запису з бази даних

За допомогою пункту головного меню «Видалення запису» здійснено виклик запиту та його опрацювання для видалення, що зображено на рисунку 5.13

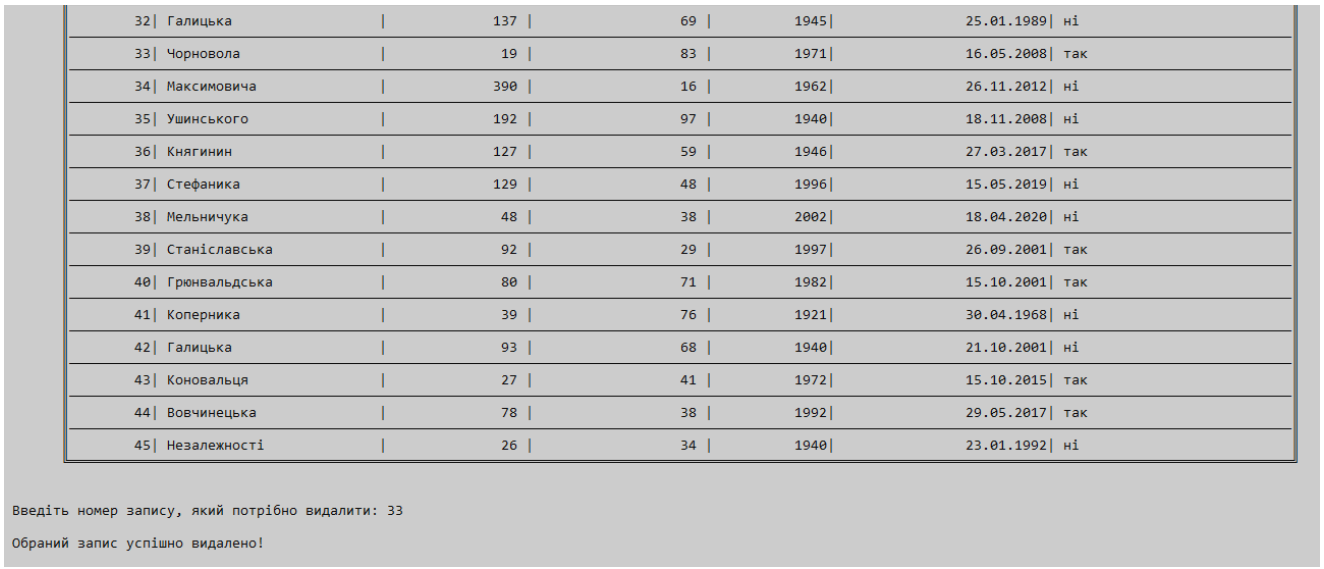


Рисунок 5.13 – Видалення запису з бази даних

Вигляд меню для вибору фільтрування даних зображено на рисунку 5.14

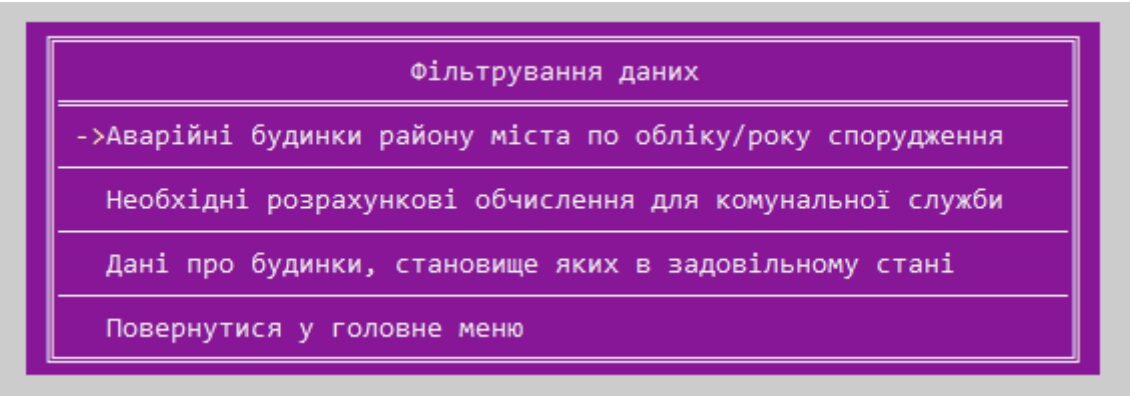


Рисунок 5.14 – Меню фільтрування функцій

За допомогою пункту меню «Фільтрування даних» «Будинки, поставлені на облік поточного року» здійснено виклик запиту та його опрацювання для виводу, що зображено на рисунку 5.15

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
5	Тролейбусна	34	47	1998	26.06.2020	ні
14	Симоненка	159	29	2018	15.09.2020	ні
21	Побутова	17	27	2001	18.12.2020	ні
25	Миколайчука	192	29	1999	27.12.2020	ні
26	Яблунева	16	16	2018	17.08.2020	ні
37	Мельничука	48	38	2002	18.04.2020	ні

Рисунок 5.15 – Будинки, поставлені на облік поточного року

За допомогою пункту меню «Фільтрування даних» «Будинки, які через 10 років святкуватимуть ювілей» здійснено виклик запиту та його опрацювання для виводу даних, що зображено на рисунку 5.16

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
20	Сагайдачного	2	19	1925	24.11.1974	ні
23	Тисменицька	37	41	1926	26.07.1973	ні
28	Новоселицька	43	62	1919	17.12.1965	ні
40	Коперника	39	76	1921	30.04.1968	ні

Рисунок 5.16 – Будинки, які через 10 років святкуватимуть ювілей

За допомогою пункту меню «Фільтрування даних» «Будинки, які взагалі не потребують ремонту» здійснено виклик запиту та його опрацювання для виводу даних, що зображено на рисунку 5.17

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
43	Вовчинецька	78	38	1992	29.05.2017	так

Рисунок 5.17 – Будинки, які взагалі не потребують ремонту

Вигляд меню для вибору сортування зображено на рисунку 5.18

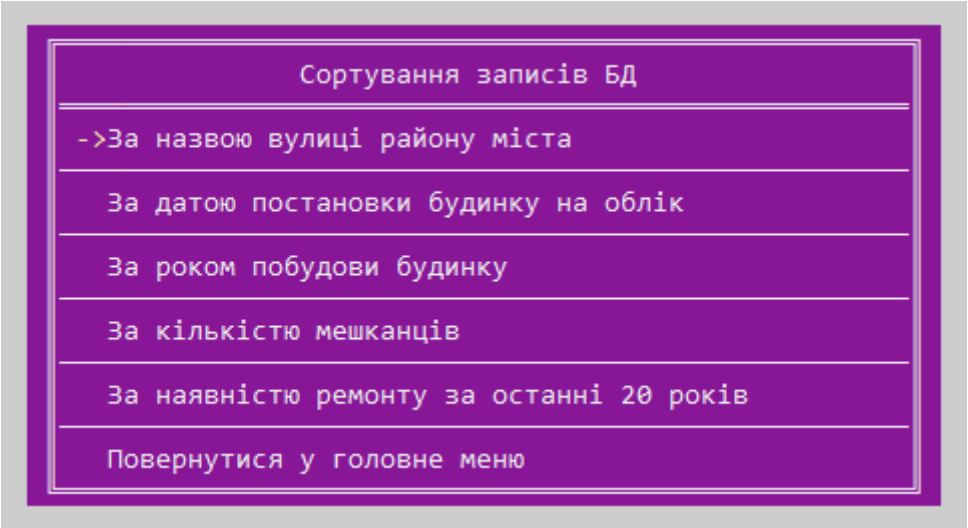


Рисунок 5.18 – Меню для вибору сортування

За допомогою пункту меню «Сортування даних по категоріях» «За датою постановки будинку на облік» «За спаданням» здійснено виклик запиту та його опрацювання для виводу даних, що зображено на рисунку 5.19

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
1	Миколайчука	192	29	1999	27.12.2020	ні
2	Побутова	17	27	2001	18.12.2020	ні
3	Симоненка	159	29	2018	15.09.2020	ні
4	Яблунева	16	16	2018	17.08.2020	ні
5	Тролейбусна	34	47	1998	26.06.2020	ні
6	Мельничука	48	38	2002	18.04.2020	ні
7	Тисменицька	89	18	2004	31.03.2020	ні
8	Галицька	145	27	2008	18.08.2019	ні
9	Стефаника	129	48	1996	15.05.2019	ні
10	Шевченка	47	23	2008	11.01.2019	ні
11	Коновальця	36	56	1996	30.08.2018	ні
12	Сагайдачного	178	89	2009	16.03.2018	ні
13	Височана	28	19	1986	24.07.2017	так
14	Вовчинецька	78	38	1992	29.05.2017	так
15	Василіянок	46	47	1989	29.03.2017	так
16	Княгинин	127	59	1946	27.03.2017	так
17	Микитинецька	25	39	1983	31.11.2016	так
18	Шевченка	17	91	1994	21.10.2016	ні
19	Коновальця	27	41	1972	15.10.2015	так
20	Максимовича	390	16	1962	26.11.2012	ні
21	Декабристів	28	47	1965	13.07.2012	так
22	Комунальна	56	82	1983	18.07.2011	так
23	Вовчинецька	192	98	1958	27.11.2010	ні

Рисунок 5.19 – Сортування даних за датою постановки будинку на облік за спаданням

Вигляд меню для вибору актуальних операцій зображено на рисунку 5.20

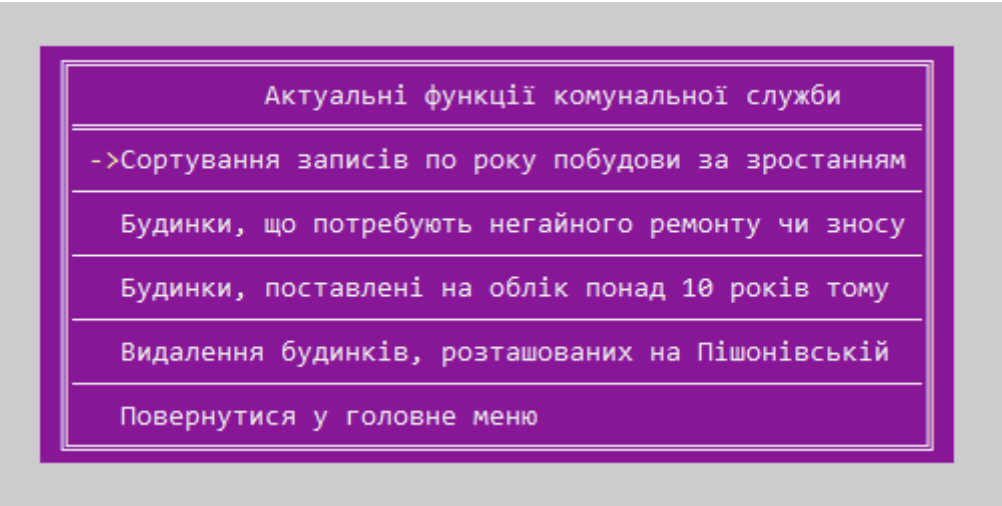


Рисунок 5.20 – Меню актуальних функцій

За допомогою пункту меню «Актуальні функції» здійснено виклик запиту та його опрацювання для виводу даних про будинки, що потребують негайного ремонту чи зносу, що зображено на рисунку 5.21

№ запису	Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за останні 20 років
7	Молодіжна	154	67	1939	27.02.1968	ні
15	Галицька	210	78	1932	23.02.1985	ні
28	Новоселицька	43	62	1919	17.12.1965	ні
34	Ушинського	192	97	1940	18.11.2008	ні
40	Коперника	39	76	1921	30.04.1968	ні
41	Галицька	93	68	1940	21.10.2001	ні

Рисунок 5.21 – Будинки, що потребують негайного ремонту чи зносу

Скріншот вмісту вихідного текстового файлу зображено на рисунку 5.22

Будинки, поставлені на облік понад 10 років тому:

Назва вулиці	Номер будинку	Кількість мешканців	Рік побудови	Дата постановки на облік	Ремонт за остан
Незалежності	12	57	1968	21.03.1993	так
Симоненка	26	37	1975	27.07.2003	так
Сахарова	18	25	1942	18.06.1987	так
Молодіжна	154	67	1939	27.02.1968	ні
Бельведерська	12	49	1935	15.02.1994	так
Незалежності	29	38	1997	18.09.2008	так
Вовчинецька	192	98	1958	27.11.2010	ні
Галицька	210	78	1932	23.02.1985	ні
Хоткевича	119	50	1956	16.04.1998	так
Надрічна	127	26	1932	21.05.1984	ні
Сагайдачного	2	19	1925	24.11.1974	ні
Тисменицька	37	41	1926	26.07.1973	ні
Новоселицька	43	62	1919	17.12.1965	ні
Симоненка	16	79	1953	21.11.2002	так
Галицька	137	69	1945	25.01.1989	ні
Ушинського	192	97	1940	18.11.2008	ні
Станіславська	92	29	1997	26.09.2001	так
Грюнвальдська	80	71	1982	15.10.2001	так
Коперника	39	76	1921	30.04.1968	ні

Рисунок 5.22 – Вміст вихідного текстового файлу

ВИСНОВКИ

Під час розробки курсового проєкту було закріплено знання з дисципліни “Основи програмування”.

В процесі проєктування програми застосовувався метод низхідного проєктування, що дозволяє розбивати задачі на менші підзадачі, чим полегшує реалізацію розробки функцій програми.

Під час написання програмного продукту було покращено навички створення і використання власних функцій для вирішення завдання. В ході роботи виникали помилки, проте було знайдено ефективні методи їх вирішення. Наприклад, усунено проблему запису нових даних на початок файлів. Проблема полягала у тому, що при записі на початок у текстовому файлі рядки накладаються один на одний. Для вирішення було прийняте рішення зберігати нові дані в окремій динамічній структурі, а потім перезаписувати файли наново. При розробці курсового проєкту було також закріплено практичні навики у роботі з файлами: як текстовими, так і бінарними, а ще покращено знання у роботі зі структурами, функціями і модульною структурою програм.

Здобуто нові теоретичні знання при розробці інтерфейсу користувача. Зокрема, засвоєно теорію з використання псевдографіки, кодування графічних символів.

Реалізовано функцію редагування, яка дозволяє користувачу вносити зміни і їх зберігати на вказаному місці. При введенні постійно йде перевірка на правильність завдяки функціям валідації даних. При цьому користувач має можливість вводити дані, доки не буде введено правильно, адже використовується безкінечний цикл. Проєкт реалізовано як багатофайловий. Загалом використано 10 файлів: 6 вихідних, 1 заголовковий, 3 ресурсних. Створено додаткові функції, які будуть практичними для комунальної служби та дозволять ефективно економити час.

Завдяки курсовому проєкту отримано практичні навики у розробці програм для створення, зберігання та обробки даних, котрі зберігаються у файлах та які можна застосовувати на практиці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Курсове та дипломне проектування. Методичні вказівки. Укладач: Л.В. Пітчук. – викладач Фахового коледжу електронних приладів ІФНТУНГу.
2. Методичні вказівки до виконання курсового проекту з дисципліни "Основи програмування" для спеціальності 121 «Інженерія програмного забезпечення». Укладачі: Балабаник О.К. Семеген Ю.В. – викладачі Фахового коледжу електронних приладів ІФНТУНГу
3. Шпак З.Я. Програмування мовою С. – Львів: Оріана-Нова, 2011. – 432
4. С++. Теорія та практика: навчальний посібник.. [Електронний ресурс]. – Режим доступу: http://www.dut.edu.ua
5. Безкоштовні курси [Електронний ресурс]. – Режим доступу: https://prometheus.org.ua/
6. С++. Теорія та практика: навчальний посібник.. [Електронний ресурс]. – Режим доступу: http://www.dut.edu.ua
7. Програмування С++ [Електронний ресурс]. Режим доступу: https://www.youtube.com/playlist?list=PLDoFqsgVaX0Z7lgvvz130eF4fcscIrrF
8. Форум українських програмістів.. – Режим доступу: https://replace.org.ua/
9. Загальний форум програмістів.. – Режим доступу: https://stackoverflow.com
10. Форум програмістів. С\С++ – Режим доступу: https://www.cyberforum