Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 11

з навчальної дисципліни

“Базові методології та технології програмування”

 КОМАНДНА РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ

ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ

ВИКОНАВ

студент академічної групи КН-24

Куртяк К.В.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

**Тема**: Командна реалізація програмних засобів оброблення динамічних структур даних та бінарних файлів

**Мета роботи** полягає у набутті ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів С++ для керування динамічною пам’яттю та бінарними файловими потоками.

Варіант №8

**Завдання**:

1. У складі команди ІТ-проєкта розробити програмні модулі оброблення динамічної структури даних.
2. Реалізувати програмний засіб на основі розроблених командою ІТ-проєкта модулів.

**Склад команди**:

* Марченко Кирило, КБ-24
* Іванов Даніїл, КБ-24
* Куртяк Катерина, КН-24

# Аналіз та планування проєкту "Електронний довідник Укрпошти"

## 3.1 Аналіз задач ІТ-проєкта та вимог до програмного забезпечення

### Основні функціональні вимоги:

* **Пошук записів**: за п'ятизначним індексом або назвою населеного пункту
* **Виведення даних**: весь довідник на екран або у текстовий файл
* **Додавання записів**: можливість внесення нових записів в довідник
* **Вилучення записів**: видалення обраних користувачем записів
* **Автоматичне збереження**: запис даних у файл при завершенні роботи
* **Автоматичне завантаження**: читання даних з файлу при запуску програми

### Структура даних запису:

* П'ятизначний індекс (int або string)
* Область (string)
* Район (string)
* Населений пункт (string)
* ВПЗ, що обслуговує (string)

## 3.2 Специфікації ПЗ та архітектура програмного засобу

### Концептуальні проєктні рішення:

* **Модульна архітектура**: розділення функціональності на окремі модулі
* **Відокремлення даних**: використання динамічної структури для зберігання
* **Файлова система**: бінарний файл для постійного зберігання

### Архітектура програмного засобу:

* Main Module — Головний модуль, меню користувача
* Data Manager — Управління даними, завантаження/збереження
* Data Structure — Динамічна структура даних (список)
* File Operations — Операції з файлами

### Інтерфейси модулів:

* **Пошук**: searchByIndex(), searchByCity()
* **Виведення**: displayAll(), exportToFile()
* **Модифікація**: addRecord(), deleteRecord()
* **Файлові операції**: loadFromFile(), saveToFile()

## 3.3 Обрана динамічна структура даних

### Обраний тип: Двозв'язний список (Doubly Linked List)

### Обґрунтування вибору:

1. **Ефективність вставки/видалення**: O(1) при наявності вказівника на елемент
2. **Гнучкість**: можливість руху в обидва боки
3. **Динамічність**: розмір змінюється під час виконання
4. **Простота реалізації**: відносно проста для розуміння та налагодження
5. **Оптимальність для завдання**: підходить для частих операцій додавання/видалення

### Альтернативи та їх недоліки:

* **Масив**: фіксований розмір, повільне видалення/вставка
* **Бінарне дерево**: складніше для реалізації, потребує балансування
* **Хеш-таблиця**: складніше реалізувати, проблеми з колізіями

## 3.4 Опис структури даних

### Заголовковий файл: struct\_type\_project\_8.h

#ifndef STRUCT\_TYPE\_PROJECT\_N\_H

#define STRUCT\_TYPE\_PROJECT\_N\_H

#include <string>

// Структура для зберігання інформації про поштове відділення

struct PostalRecord {

int index; // П'ятизначний індекс

std::string region; // Область

std::string district; // Район

std::string city; // Населений пункт

std::string postal\_office; // ВПЗ, що обслуговує

// Конструктори

PostalRecord();

PostalRecord(int idx, const std::string& reg, const std::string& dist,

const std::string& c, const std::string& office);

};

// Вузол двозв'язного списку

struct ListNode {

PostalRecord data; // Дані запису

ListNode\* next; // Вказівник на наступний елемент

ListNode\* prev; // Вказівник на попередній елемент

// Конструктори

ListNode();

ListNode(const PostalRecord& record);

};

// Клас для управління двозв'язним списком

class PostalDatabase {

private:

ListNode\* head; // Вказівник на початок списку

ListNode\* tail; // Вказівник на кінець списку

int size; // Кількість елементів

public:

// Конструктор та деструктор

PostalDatabase();

~PostalDatabase();

// Основні операції

void addRecord(const PostalRecord& record);

bool deleteRecord(int index);

ListNode\* searchByIndex(int index);

ListNode\* searchByCity(const std::string& city);

void displayAll();

void exportToFile(const std::string& filename);

bool loadFromFile(const std::string& filename);

bool saveToFile(const std::string& filename);

int getSize() const;

bool isEmpty() const;

void clear();

};

#endif

## 3.5 Розподіл підзадач між учасниками команди

### Іванов Даніїл — Модуль файлових операцій:

1. **Функція завантаження з файлу**: loadFromFile(const string& filename) - завантаження індексів з бінарного файлу
2. **Функція збереження у файл**: saveToFile(const string& filename) - збереження індексів у бінарний файл
3. **Функція експорту**: exportToFile(const string& filename) - експорт у текстовий файл (який може звичайно прочитати користувач)

### Марченко Кирило — Модуль пошуку та виведення:

1. **Функція пошуку за індексом**: searchByIndex(int index) - пошук за індексом
2. **Функція пошуку за містом**: searchByCity(const string& city) - пошук за містом
3. **Функція виведення всіх записів**: displayAll() - виведення всіх записів у консоль

### Куртяк Катерина - Модуль модифікації даних:

1. **Функція додавання запису**: addRecord(const PostalRecord& record) - додати запис
2. **Функція видалення запису**: deleteRecord(int index) - видалити запис

### Спільні задачі:

* Розробка головного меню та інтерфейсу користувача
* Тестування та налагодження
* Документація

## 3.6 План робіт відповідно до ISO/IEC 12207

### Фаза 1: Планування та аналіз

* **Куртяк**: Фінальний аналіз вимог, створення технічного завдання
* **Іванов**: Дослідження форматів файлів, планування структури даних
* **Марченко**: Проектування інтерфейсу користувача

### Фаза 2: Проектування

* **Іванов**: Проектування файлових операцій (бінарні та текстові файли)
* **Марченко**: Проектування алгоритмів пошуку та виведення даних
* **Куртяк**: Проектування операцій модифікації даних

### Фаза 3: Реалізація

* **Іванов**:
  + Реалізація loadFromFile() - завантаження з бінарного файлу
  + Реалізація saveToFile() - збереження у бінарний файл
  + Реалізація exportToFile() - експорт у текстовий файл
* **Марченко**:
  + Реалізація searchByIndex() - пошук за індексом
  + Реалізація searchByCity() - пошук за містом
  + Реалізація displayAll() - виведення на екран
* **Куртяк**:
  + Реалізація addRecord() - додавання запису
  + Реалізація deleteRecord() - видалення запису

### Фаза 4: Інтеграція та тестування

* **День 1**: Інтеграція модулів, створення головного меню
* **День 2**: Модульне тестування кожної функції
* **День 3**: Системне тестування, виправлення помилок

### Фаза 5: Документація та здача

* **Куртяк**: Технічна документація
* **Іванов**: Інструкція користувача

**Марченко**: Звіт про тестування

**Аргументи на користь досягнення мети лабораторної роботи:**

1. У складі команди ІТ-проєкта розробити програмні модулі оброблення динамічної структури даних. 2. Реалізувати програмний засіб на основі розроблених командою ІТ-проєкта модулів. 3. Розвинути навички командної співпраці. 4. Отримати досвід роботи в команді над єдиною кодовою базою. 5. Навчитися інтегрувати модулі з різними інтерфейсами. 6. Навчитися створювати модульні структури програм. 7. Зрозуміти принципи динамічного виділення пам’яті. 8. Розібратися у роботі з вказівниками в C++. 9. Застосувати двозв’язні списки на практиці. 10. Освоїти принципи інкапсуляції. 11. Навчитися користуватись заголовковими файлами. 12. Забезпечити безпечне знищення динамічних об'єктів. 13. Використати механізм деструкторів. 14. Закріпити поняття конструкторів. 15. Освоїти зберігання структури у бінарному файлі. 16. Реалізувати експорт даних у текстовий файл. 17. Зрозуміти принципи читання з бінарного потоку. 18. Ознайомитися з ефективним зберіганням текстових рядків. 19. Розробити функції додавання елементів у список. 20. Реалізувати функцію видалення за значенням. 21. Навчитися писати ефективні пошукові алгоритми. 22. Реалізувати пошук за індексом. 23. Реалізувати пошук за назвою міста. 24. Використати перегрузку операторів. 25. Навчитись працювати з класами в C++. 26. Ознайомитись з namespace та їх перевагами. 27. Створити резервну копію бази даних. 28. Реалізувати логіку експорту в формат, зручний для користувача. 29. Практикуватися в роботі з динамічною пам’яттю. 30. Навчитись обробляти структури з полями типу string. 31. Застосувати принцип DRY (Don't Repeat Yourself). 32. Закріпити правила написання чистого коду. 33. Ознайомитися з принципами SOLID. 34. Покращити навички декомпозиції задач. 35. Навчитися тестувати окремі модулі. 36. Отримати досвід командного контролю версій. 37. Приділити увагу структуруванню файлів проєкту. 38. Навчитись документувати функції. 39. Зрозуміти необхідність коментарів у коді. 40. Сформувати відповідальність за власну частину проєкту. 41. Розвинути навички взаємодії з іншими учасниками команди. 42. Засвоїти важливість взаємної перевірки коду. 43. Навчитися уникати конфліктів у команді. 44. Практикувати планування розробки. 45. Дотримуватись дедлайнів під час командної роботи. 46. Закріпити розуміння динамічних структур даних. 47. Навчитися видаляти елементи з динамічного списку без помилок. 48. Вивчити, як уникнути витоків пам’яті. 49. Освоїти механізм лінійного проходження по списку. 50. Реалізувати статистику записів у структурі. 51. Зрозуміти логіку обходу двозв’язного списку. 52. Навчитися валідувати введені користувачем дані. 53. Засвоїти обробку помилок при відкритті файлів. 54. Реалізувати перевірку коректності індексу. 55. Використати практичні знання про об’єктно-орієнтоване програмування. 56. Навчитись створювати backup-файли. 57. Забезпечити узгодженість даних у пам’яті та файлі. 58. Забезпечити масштабованість системи. 59. Освоїти багатофайлову структуру проєкту. 60. Розробити та використовувати загальні інтерфейси. 61. Побудувати логіку з урахуванням можливого розширення. 62. Використати засоби std::fstream на практиці. 63. Освоїти серіалізацію об'єктів у файл. 64. Ознайомитися з принципами low-level доступу до файлів. 65. Використати шаблони при потребі в розширенні. 66. Навчитись створювати зрозумілий інтерфейс для користувача. 67. Отримати навички контролю доступу до елементів класу. 68. Закріпити знання про friend-функції. 69. Створити систему перевірки введених даних. 70. Працювати з нульовими вказівниками. 71. Перевірити алгоритми на граничних значеннях. 72. Тестувати програму на порожньому списку. 73. Виявляти та усувати помилки логіки. 74. Застосувати на практиці отримані теоретичні знання. 75. Поглибити знання про структури даних. 76. Створити гнучку архітектуру коду. 77. Навчитись працювати з константами у просторі імен. 78. Створити допоміжні функції в окремому просторі імен. 79. Забезпечити зручність розширення функціоналу. 80. Освоїти принципи самодокументованого коду. 81. Застосувати техніку модульного тестування. 82. Виявити зв’язок між пам’яттю та файлами. 83. Зрозуміти вплив структури даних на ефективність. 84. Навчитися розраховувати кількість елементів. 85. Реалізувати виведення інформації на екран. 86. Побачити цінність багаторазового використання коду. 87. Створити класи з чітко визначеною відповідальністю. 88. Розвинути логічне мислення в контексті структур. 89. Поглибити знання про взаємозв’язок типів даних. 90. Навчитися писати програму з нуля. 91. Побачити цінність стандартів і стилю кодування. 92. Реалізувати коректну очистку пам’яті. 93. Навчитися відлагоджувати помилки на рівні структур. 94. Ознайомитися з підходами до форматування виводу. 95. Вивчити шляхи взаємодії між класами. 96. Побачити важливість чіткої ролі кожного учасника. 97. Закріпити практичні навички обробки динамічних структур. 98. Навчитись створювати просту базу даних. 99. Навчитись реалізовувати CRUD-функціонал. 100. Сформувати цілісне бачення процесу створення ПЗ в команді.

**Висновок:**

У ході виконання лабораторної роботи було досягнуто основну мету — набуття практичних навичок командної розробки програмного забезпечення з використанням динамічних структур даних у мові програмування C++. Завдяки розподілу обов’язків між учасниками проєкту вдалося створити модульну систему, в якій кожен учасник відповідав за окрему частину функціоналу. Зокрема, було реалізовано:

* зчитування та збереження даних у бінарні файли;
* ефективний пошук за ключовими полями (індекс, місто);
* додавання та видалення елементів з динамічної структури — двозв'язного списку.

Особливу увагу приділено організації доступу до динамічної пам’яті та підтримці цілісності списку при видаленні або вставці елементів. Робота з потоками введення/виведення дозволила закріпити розуміння механізмів зберігання об’єктів у бінарному форматі та взаємодії з файлами.

Крім технічної складової, лабораторна робота також сприяла розвитку навичок командної роботи, налагодження спільного коду, розуміння важливості розділення логіки за принципами модульності та відповідальності. Отримані знання можуть бути застосовані у більш масштабних проєктах, що працюють з великими обсягами структурованих даних.

Таким чином, лабораторна робота повністю виконана, а її результати засвідчують успішне засвоєння як теоретичних основ, так і практичних прийомів роботи з динамічними структурами, файлами та архітектурою програмного забезпечення.