

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
БМТП

Звіт
Лабораторна робота №11

Виконав: ст. Гр. КБ-24-1
Іванов Даніїл Андрійович
Перевірив: Викладач
Анастасія Сергіївна Коваленко

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 C++ для керування

динамічною пам'яттю та бінарними файловими потоками.

ЗАВДАННЯ:

Базові інтеграції на тематичні проєкти: 3. Лабораторна робота № 11
github.com/odorenskyi/Kurtiak-Kateryna-KN24
github.com/odorenskyi/Marchenko-Kyrylo-KB24
github.com/odorenskyi/Ivanov-Danill-KB24

ВАРІАНТ 8

— ЗАВДАННЯ НА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ —

Створити електронний довідник індексів та відділень поштового зв'язку України УДППЗ “Укрпошта” (п'ятизначний індекс, область, район, населений пункт, ВПЗ, що обслуговує).

За вибором користувача (оператора) програма забезпечує:

- пошук запису в довіднику за введеним індексом або назвою населеного пункту;
- виведення всього довідника на екран або у текстовий файл;
- додавання нового запису в довідник;
- вилучення заданого оператором запису з довідника;
- завершення роботи програми з автоматичним записом даних у файл.

Дані довідника автоматично завантажуються з файлу під час запуску програми.

Поштові індекси та відділення поштового зв'язку України

Пошук за індексом

25006 Пошук здійснюється за п'ятизначним індексом

Пошук за місцем розташування

Пошук за назвою населеного пункту

Знайдено 3 записи

Індекс	Область	Район	Населений пункт	До переіменування	ВПЗ, що обслуговує
25006	Кіровоградська	Кропивницький	Кіровоград		Кропивницький.б
25006	Кіровоградська	Кропивницький	Кіровоград		Центр м.Кропивницький
25006	Кіровоградська	Кропивницький	Кіровоград		ЦОС № 5 м. Кропивницький

Функції: addRecord(), deleteRecord()

Заголовок: struct_type_project_Ivanov.h

Бібліотека: libIvanov.a 2.1

Проектування архітектури

Модуль Data (Ivanov): додає та вилучає елементи з однозв'язного списку

Інтерфейс: // struct_type_project_Ivanov.h #pragma once #include
"struct_type_project_8.h"

```
namespace Ivanov { bool addRecord(PostalRecord*& head, const  
PostalRecord& newRec); bool deleteRecord(PostalRecord*& head, int index,  
const std::string& city); }
```

2.2 Детальне проєктування Алгоритм addRecord: Створити new PostalRecord з копією newRec Додати на початок списку (або в кінець за домовленістю) Повернути true Алгоритм deleteRecord: Якщо head == nullptr → false Ітерувати, шукати перший вузол, що відповідає критеріям (index та city) Переважно зв'язати попередній із наступним, видалити вузол, повернути true Якщо не знайдено → false

2.3 Конструювання (код) // struct_type_project_Ivanov.cpp #include
"struct_type_project_Ivanov.h"

```
bool Ivanov::addRecord(PostalRecord*& head, const PostalRecord& newRec) {  
    auto* node = new PostalRecord(newRec); node->next = head; head = node;  
    return true; }
```

```
bool Ivanov::deleteRecord(PostalRecord*& head, int index, const std::string&  
city) { PostalRecord* prev = nullptr; for (auto* cur = head; cur; prev = cur, cur  
= cur->next) { if (cur->index == index && cur->city == city) { if (prev) prev-  
>next = cur->next; else head = cur->next; delete cur; return true; } } return  
false; }
```

2.4 Тестування

Test Case ID	Action	Expected Result	Test Result
I001	addRecord(head, sampleRec) на порожній структурі head != nullptr і head->index == sampleRec.index		
I002	Додавання кількох записів, потім пошук за першим	Перший елемент списку має дані sampleRec	
I003	deleteRecord(head, existingIndex, existingCity)	Повернути true, елемент вилучено із списку	
I004	deleteRecord(head, nonExistIndex, "");	Повернути false, список без змін	

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: Ефективність вставки/видалення: $O(1)$ при наявності вказівника на елемент Гнучкість: можливість руху в обидва боки Динамічність: розмір змінюється під час виконання Простота реалізації: відносно проста для розуміння та налагодження 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
```

```
// 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 Данііл —

Модуль файлових операцій: Функція завантаження з файлу:

loadFromFile(const string& filename) - завантаження індексів з бінарного

файлу Функція збереження у файл: saveToFile(const string& filename) -

збереження індексів у бінарний файл Функція експорту: exportToFile(const string& filename) - експорт у текстовий файл (який може звичайно

прочитати користувач) Марченко Кирило — Модуль пошуку та виведення:

Функція пошуку за індексом: searchByIndex(int index) - пошук за індексом

Функція пошуку за містом: searchByCity(const string& city) - пошук за

містом Функція виведення всіх записів: displayAll() - виведення всіх

записів у консоль Куртяк Катерина - Модуль модифікації даних: Функція

додавання запису: addRecord(const PostalRecord& record) - додати запис

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

- Основам структурованого програмування на C/C++.

- Що таке модульність і як її застосовувати.
- Як працює компіляція багатофайлових проєктів.
- Що таке заголовочні файли і як їх підключати.
- Як оголошуються та використовуються структури (**struct**).
- Принципи побудови бібліотек.
- Основи компоновки (**linking**) при збиранні проєкту.
- Що таке інкапсуляція в контексті модульного коду.
- Роль документації у програмуванні.
- Призначення **Makefile** або інших засобів автоматизації збірки.
- Створення проєктної структури.
- Написання заголовочних файлів (**.h**).
- Реалізація функцій у **.c / .cpp** файлах.
- Імплементация власної бібліотеки.
- Використання бібліотек інших студентів (колективна робота).
- Робота з масивами структур.
- Пошук та сортування структур.
- Застосування циклів для обробки структур.
- Використання умовних операторів (**if, switch**) в контексті даних.
- Організація вводу/виводу з файлів.
- Спільна розробка проєкту.
- Підключення зовнішніх бібліотек від колег.
- Використання єдиного шаблону проєкту.
- Уніфікація інтерфейсів між модулями.
- Спільне тестування системи.
- Орієнтування у файловій структурі проєкту.
- Робота з IDE або текстовим редактором.
- Налаштування коду (**debugging**).
- Перевірка працездатності окремих модулів.
- Компіляція з командного рядка (за потреби).
- Розробка алгоритмів обробки даних.
- Вибір ефективних способів сортування/пошуку.
- Побудова логіки взаємодії між модулями.
- Планування структур даних під конкретну задачу.
- Аналіз складності алгоритмів.

- Проведення ручного тестування.
- Створення тестових даних.
- виправлення помилок компіляції.
- Усунення логічних помилок.
- Розуміння повідомлень компілятора.
- Структуризація проєкту за модулями.
- Чітке розмежування інтерфейсу та реалізації.
- Коментування функцій і структур.
- Використання читаємих імен змінних.
- Підготовка коду до презентації або захисту.
- Аналіз функціональності бібліотек.
- Визначення точок з'єднання модулів.
- Виявлення повторного коду.
- Оптимізація структури програми.
- Розбір чужого коду (з бібліотек інших студентів).
- Написання README-файлу.
- Опис структури проєкту.
- Пояснення роботи кожного модуля у звіті.
- Використання прикладів у документації.
- Аргументація вибраного підходу до реалізації.
- Формалізація задачі у вигляді алгоритму.
- Перетворення алгоритму в програмний код.
- Пошук та виправлення логічних помилок.
- Самостійне прийняття рішень щодо реалізації.
- Застосування теоретичних знань на практиці.
- Самоаналіз власного коду.
- Вивчення сильних сторін чужих рішень.
- Порівняння різних підходів до вирішення однієї задачі.
- Отримання зворотного зв'язку.
- Поліпшення коду після перевірки.
- Створення структурованого і масштабованого коду.
- Дотримання стандартів проєктування.
- Формування професійного підходу до задачі.
- Основи командної розробки.

- Навички рефакторингу.
- Вироблення відповідальності.
- Покращення тайм-менеджменту.
- Розвиток уваги до деталей.
- Здатність працювати самостійно.
- Зростання впевненості у власних силах.