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

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КБ-24

Науменко О. В.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

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. 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.

**Варіант №5**

**Склад команди:**

Науменко Олександр, КБ-24 (<https://github.com/Devyadnaumenko>)  
Бовкун Дмитро, КБ-24  (<https://github.com/sttaxs>)

Сарваді Роман, КІ-24 (<https://github.com/xneverdiedx>)

Розподіл завдань між учасниками команди:

Sarvadi Roman

1. Завантаження даних з бінарного файлу при запуску програми.

-Реалізація функції loadFromFile(), яка читає дані з бінарного файлу та ініціалізує динамічну структуру даних.

-Використання ifstream та методів read().

1. Пошук рейсів за назвою станції (початкової або кінцевої).

-Функція searchByStation(const string& stationName), яка обходить структуру даних і виводить всі рейси, де зустрічається задана назва (без урахування регістру).

Bovkun Dmytro

1. Виведення всього розкладу на екран.

-Функція printAllSchedules(), яка ітерується по структурі даних і форматує вивід у зручному вигляді.

1. Додавання нового рейсу.

-Функція addNewRoute(), яка запитує користувача про дані нового рейсу, додає його до структури та перевіряє на унікальність номера.

Naumenko Oleksandr

1. Видалення рейсу за номером.

-Функція deleteRoute(int routeNumber), яка знаходить рейс у структурі за номером, видаляє його та звільняє пам’ять.

1. Збереження даних у файл при завершенні роботи.

-Функція saveToFile(), яка серіалізує динамічну структуру даних у бінарний файл за допомогою ofstream та write().

Аналіз задач ІТ-проєкту та вимог до ПЗ

Завдання системи:

1. Завантаження бази з файлу при старті програми. Програма має при

запуску сканувати файл з даними розкладу автобусів і завантажувати в пам’ять існуючий розклад.

1. Відображення всієї бази на екрані або вивід у текстовий файл (за

вибором користувача). Користувач може за потреби переглянути весь розклад прямо в консолі або експортувати його у текстовий файл для подальшого аналізу.

1. Додавання нового запису (новий рейс). Оператор може додати новий

запис про рейс автобуса з вказанням номеру, станцій відправлення та прибуття, частоти рейсу, часу відправлення та часу прибуття.

1. Пошук запису за назвою зупинки або маршрутом. Програма дозволяє

шукати записи, у яких одна з станцій (відправлення або прибуття) містить вказаний шаблон (без врахування великих/малих літер).

1. Видалення запису за номером рейсу. Забезпечується можливість

видалення конкретного запису (рейсу) за його унікальним номером (аналог табельного номера).

1. Збереження бази у файл при завершенні роботи. Після завершення

роботи дані розкладу автоматично зберігаються у файл для подальшого використання.

Специфікації ПЗ, концептуальні рішення та архітектура

-Головний файл (main.cpp) Єдиний «оркестратор» системи. Він містить консольне меню, забезпечує виклик усіх операцій та інтегрує роботу модулів.

-Три незалежні модулі (по 2 функції в кожному), які залежать лише від спільного заголовку struct\_type\_project\_2.h:

-Module\_File\_Naumenko – відповідає за завантаження бази з файлу та її збереження: функції loadDatabase() та saveDatabase().

-Module\_CRUD\_Bovkun – відповідає за редагування даних, а саме за додавання нового запису (нового рейсу) та відображення всієї бази: функції addRecord() та printAllRecords().

-Module\_SearchDel\_Sarvadi – відповідає за пошук запису та видалення: функції findRecordByName() (пошук за назвою зупинки чи маршрутом) і deleteRecord() (видалення за номером рейсу).

-Спільний заголовковий файл (struct\_type\_project\_2.h) містить опис структури даних для розкладу (у нашому випадку ­– запис про рейс автобуса) та визначення вузла однозв’язного списку.

Схема взаємодії модулів

-main.cpp → викликає модулі:

-Module\_File\_Naumenko (loadDatabase, saveDatabase)

-Module\_CRUD\_Bovkun (addRecord, printAllRecords)

-Module\_SearchDel\_Sarvadi (findRecordByName, deleteRecord)

-Всі ці модулі включають заголовковий файл struct\_type\_project\_2.h для спільного визначення структури даних.

Алгоритм роботи з програми:

1. При старті викликається loadDatabase(head, filename) для завантаження розкладу.
2. В головному меню користувач обирає одну з операцій: додавання, перегляд, пошук чи видалення.
3. Після завершення роботи викликається saveDatabase(head, filename) для збереження актуальної бази.

Розподіл підзадач (Таблиця 1)

|  |  |  |
| --- | --- | --- |
| **Модуль** | **Студент** | **Функції** |
| **Module\_File\_Naumenko** | Naumenko | loadDatabase(), saveDatabase() |
| **Module\_CRUD\_Bovkun** | Bovkun | addRecord(), printAllRecords() |
| **Module\_SearchDel\_Sarvadi** | Sarvadi | findRecordByName(), deleteRecord() |

План робіт за ISO/IEC 12207 (Таблиця 2)

|  |  |
| --- | --- |
| **Етап** | **Відповідальні** |
| 1. Планування | Усі |
| 2. Реалізація модулів | За модулями |
| 3. Інтеграція та тестування | Усі |
| 4. Верифікація відповідності вимогам | Усі |
| 5. Валідація, демонстрація викладачу | Усі + викладач |

Аналіз модуля Module\_SearchDel\_Sarvadi

Ціль модуля: Модуль Module\_SearchDel\_Sarvadi реалізує дві ключові операції:

-Пошук запису за назвою зупинки або маршрутом:

-Користувач викликає функцію findRecordByName(head, "Кропивницький").

-Функція перевіряє, чи список не є порожнім.

-Задану назву (шаблон) приводять до нижнього регістру.

-Програма проходитиме по кожному елементу списку, перетворюючи значення поля departureStop і arrivalStop у нижній регістр для пошуку заданого шаблону.

-При знаходженні збігу виводиться повна інформація про рейс, а лічильник збільшується.

-Якщо записи знайдені, в кінці виводиться їх кількість, інакше – повідомлення про відсутність співпадінь.

-Видалення запису за номером рейсу:

-Користувач викликає функцію deleteRecord(head, 123).

-Програма перевіряє наявність елементів у списку.

-Якщо перший елемент містить заданий routeNumber, він видаляється, а покажчик head оновлюється.

-Якщо ні – програма проходить список, шукаючи елемент, у якого routeNumber співпадає зі заданим.

-При успішному видаленні виводиться відповідне повідомлення, якщо запис не знайдено – повідомляється про помилку.

-Функція повертає логічний результат (true або false).

Приклад бази даних розкладу автобусів

Приклад записів:

--- Запис #1 --- Номер рейсу: 1 Станція відправлення: Кропивницький Станція прибуття: Заріччя Частота: Щодня Час відправлення: 08:00 Час прибуття: 08:45

--- Запис #2 --- Номер рейсу: 2 Станція відправлення: Кропивницький Станція прибуття: Левківка Частота: Парні дні Час відправлення: 09:15 Час прибуття: 10:00

--- Запис #3 --- Номер рейсу: 3 Станція відправлення: Кропивницький Станція прибуття: Комсомольськ Частота: Непарні дні Час відправлення: 10:30 Час прибуття: 11:20