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\_5.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\_5.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\_5.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:

101;Кропивницький — Дніпро;щоденно;до 29.10;06:50;06:50;16:45;16:45

102;Кропивницький — Допинська;щодня, окрім ср;до 29.10;13:50;13:50;;

103;Кропивницький — Допинська;щодня, окрім ср;з 18.04 до 29.10;08:30;08:30;;

104;Кропивницький — Допинська;щоденно;до 29.10;07:30;07:30;09:50,11:30,16:20;09:50,11:30,16:20

105;Кропивницький — Жовті Води;щоденно;до 29.10;12:45;12:45;;

106;Кропивницький — Завалля;щоденно;з 17.04 до 29.10;15:30;15:30;;

107;Кропивницький — Знам'янка;щоденно;до 29.10;05:30,06:00,06:35,07:00,08:00,09:55,12:00;05:30,06:00,06:35,07:00,08:00,09:55,12:00;13:00,15:10,16:30,17:00,18:30,18:55;13:00,15:10,16:30,17:00,18:30,18:55

108;Кропивницький — Кам'янське;щоденно;з 17.04 до 29.10;14:30;14:30;;

109;Кропивницький — Київ-1, Центральний автовокзал;щоденно;до 29.10;08:35;08:35;19:25;19:25

110;Кропивницький — Київ, автостанція «Видубичі»;щоденно;з 17.04 до 29.10;07:00,10:00,23:00;07:00,10:00,23:00;;;

111;Кропивницький — Київ, автостанція «Видубичі»;щоденно;до 29.10;00:01,14:00,17:10;00:01,14:00,17:10;;

112;Кропивницький — Кременчук;щоденно;з 17.04 до 29.10;06:20;06:20;13:20,15:45;13:20,15:45

113;Кропивницький — Кривий Ріг;щоденно;до 29.10;12:40;12:40;09:10,11:00;09:10,11:00

114;Кропивницький — Медведівка;щоденно;до 29.10;13:00;13:00;;

Тест-сьюти:

Модуль: saveToFile

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Назва тесту** | **Стан системи** | **Очікуваний результат** |
| 1 | Збереження 1 маршруту | 1 запис у пам'яті | Файл містить 1 рядок |
| 2 | Збереження 0 маршрутів | Порожній список | Файл порожній |
| 3 | Перезапис існуючого файлу | Існує файл з інфо | Старий вміст замінено |
| 4 | Дані з нестандартними символами | Юнікод в назвах | Збереження збережено коректно |
| 5 | Відсутній дозвіл на запис | Тест з /readonly/... | Помилка при збереженні (в ручну перевірити) |
| 6 | Перевірка формату рядків | Дані з ручним вводом | Формат: id;route;... збережено правильно |

Модуль: removeRoute

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Назва тесту** | **Введене ID** | **Очікуваний результат** |
| 1 | Видалення існуючого маршруту | ID=101 | Рейс видалено |
| 2 | Видалення неіснуючого | ID=9999 | Вивід: "Рейс не знайдено" |
| 3 | Введено не число | "abc" | Помилка (ручна перевірка) |
| 4 | Порожній ввід | Enter | Ігнорується або помилка |
| 5 | Два однакових ID | ID=5 x2 | Видалені обидва |
| 6 | Видалення останнього рейсу | Лише один в списку | Після видалення — розклад порожній |

Вихідний код проєкту prj\_5\_Naumenko

#include "constructor.h"

#include "add\_route.h"

#include "remove\_route.h"

#include "load\_from\_file.h"

#include "save\_to\_file.h"

#include "print\_schedule.h"

#include "search\_by\_station.h"

int main() {

ScheduleSystem system("routes.txt");

int choice;

do {

std::cout << "\n1. Додати рейс\n"

<< "2. Видалити рейс\n"

<< "3. Завантажити з файлу\n"

<< "4. Зберегти у файл\n"

<< "5. Вивести розклад\n"

<< "6. Пошук за станцією\n"

<< "0. Вихід\n"

<< "Виберіть опцію: ";

std::cin >> choice;

std::cin.ignore();

switch (choice) {

case 1: system.addRoute(); break;

case 2: system.removeRoute(); break;

case 3: system.loadFromFile(); break;

case 4: system.saveToFile(); break;

case 5: system.printSchedule(); break;

case 6: system.searchByStation(); break;

}

} while (choice != 0);

return 0;

}

Вихідний код модулю remove\_route:

remove\_route.cpp:

#include "remove\_route.h"

#include <iostream>

#include <algorithm>

void ScheduleSystem::removeRoute() {

int id;

std::cout << "Введіть номер рейсу для видалення: ";

std::cin >> id;

std::cin.ignore();

auto it = std::remove\_if(routes.begin(), routes.end(), [id](const BusRoute& r) {

return r.id == id;

});

if (it != routes.end()) {

routes.erase(it, routes.end());

std::cout << "Рейс видалено.\n";

}

else {

std::cout << "Рейс не знайдено.\n";

}

}

remove\_route.h:

#ifndef REMOVE\_ROUTE\_H

#define REMOVE\_ROUTE\_H

#include "struct\_type\_project\_5.h"

class ScheduleSystem {

public:

void removeRoute();

};

#endif // REMOVE\_ROUTE\_H

Вихідний код модулю save\_to\_file:

save\_to\_file.cpp:

#include "save\_to\_file.h"

#include <iostream>

#include <fstream>

void ScheduleSystem::saveToFile() const {

std::ofstream fout(filename);

if (!fout) {

std::cerr << "Не вдалося зберегти файл: " << filename << "\n";

return;

}

for (const auto& r : routes) {

fout << r.id << ";" << r.route << ";" << r.frequency << ";" << r.period << ";"

<< r.startArrive << ";" << r.startDepart << ";" << r.endDepart << ";" << r.endArrive << "\n";

}

fout.close();

std::cout << "Дані збережено у файл: " << filename << "\n";

}

save\_to\_file.h:

#ifndef SAVE\_TO\_FILE\_H

#define SAVE\_TO\_FILE\_H

#include "struct\_type\_project\_5.h"

class ScheduleSystem {

public:

void saveToFile() const;

};

#endif // SAVE\_TO\_FILE\_H

Вихідний код модулю constructor:

constructor.cpp:

#include "constructor.h"

ScheduleSystem::ScheduleSystem(const std::string& file) : filename(file) {

loadFromFile();

}

constructor.h:

#ifndef CONSTRUCTOR\_H

#define CONSTRUCTOR\_H

#include "struct\_type\_project\_5.h"

#include <string>

class ScheduleSystem {

public:

ScheduleSystem(const std::string& file);

};

#endif // CONSTRUCTOR\_H

Вихідний код модулю load\_from\_file:

load\_from\_file.cpp:

#include "load\_from\_file.h"

#include <iostream>

#include <fstream>

#include <sstream>

void ScheduleSystem::loadFromFile() {

std::cout << "Спроба відкрити файл: " << filename << "\n";

std::ifstream fin(filename);

if (!fin) {

std::cerr << "Не вдалося відкрити файл: " << filename << "\n";

return;

}

routes.clear();

std::string line;

int lineNumber = 0;

while (std::getline(fin, line)) {

std::cout << line << std::endl;

lineNumber++;

if (line.empty()) continue;

std::stringstream ss(line);

std::string field;

BusRoute r;

try {

if (!std::getline(ss, field, ';') || field.empty()) throw std::runtime\_error("Неправильний ID");

r.id = std::stoi(field);

if (!std::getline(ss, r.route, ';') || r.route.empty()) throw std::runtime\_error("Неправильний маршрут");

if (!std::getline(ss, r.frequency, ';') || r.frequency.empty()) throw std::runtime\_error("Неправильна періодичність");

if (!std::getline(ss, r.period, ';') || r.period.empty()) throw std::runtime\_error("Неправильний період");

if (!std::getline(ss, r.startArrive, ';') || r.startArrive.empty()) throw std::runtime\_error("Неправильне прибуття (початкова)");

if (!std::getline(ss, r.startDepart, ';') || r.startDepart.empty()) throw std::runtime\_error("Неправильне відправлення (початкова)");

if (!std::getline(ss, r.endDepart, ';') || r.endDepart.empty()) throw std::runtime\_error("Неправильне відправлення (кінцева)");

if (!std::getline(ss, r.endArrive, ';') || r.endArrive.empty()) throw std::runtime\_error("Неправильне прибуття (кінцева)");

routes.push\_back(r);

}

catch (const std::exception& e) {

std::cerr << "Помилка в рядку " << lineNumber << ": " << e.what() << "\n";

}

}

fin.close();

std::cout << "Зчитано " << routes.size() << " рейсів.\n";

}

load\_from\_file.h:

#ifndef LOAD\_FROM\_FILE\_H

#define LOAD\_FROM\_FILE\_H

#include "struct\_type\_project\_5.h"

class ScheduleSystem {

public:

void loadFromFile();

};

#endif // LOAD\_FROM\_FILE\_H

Вихідний код модулю print\_schedule:

print\_schedule.cpp:

#include "print\_schedule.h"

#include <iostream>

void ScheduleSystem::printSchedule() const {

std::cout << "Кількість рейсів: " << routes.size() << "\n";

if (routes.empty()) {

std::cout << "Розклад порожній або не завантажено.\n";

return;

}

for (const auto& r : routes) {

std::cout << "ID: " << r.id << "\n"

<< "Маршрут: " << r.route << "\n"

<< "Періодичність: " << r.frequency << "\n"

<< "Період дії: " << r.period << "\n"

<< "Прибуття (початкова): " << r.startArrive << "\n"

<< "Відправлення (початкова): " << r.startDepart << "\n"

<< "Відправлення (кінцева): " << r.endDepart << "\n"

<< "Прибуття (кінцева): " << r.endArrive << "\n"

<< "------------------------------------\n";

}

}

print\_schedule.h:

#ifndef PRINT\_SCHEDULE\_H

#define PRINT\_SCHEDULE\_H

#include "struct\_type\_project\_5.h"

class ScheduleSystem {

public:

void printSchedule() const;

};

#endif // PRINT\_SCHEDULE\_H

Вихідний код модулю search\_by\_station:

search\_by\_station.cpp:

#include "search\_by\_station.h"

#include <iostream>

#include <algorithm>

void ScheduleSystem::searchByStation() const {

std::string name;

std::cout << "Введіть назву станції (початкову або кінцеву): ";

std::getline(std::cin, name);

std::string lowered;

std::transform(name.begin(), name.end(), std::back\_inserter(lowered), ::tolower);

bool found = false;

for (const auto& r : routes) {

std::string route\_lower = r.route;

std::transform(route\_lower.begin(), route\_lower.end(), route\_lower.begin(), ::tolower);

if (route\_lower.find(lowered) != std::string::npos) {

std::cout << "ID: " << r.id << " — " << r.route << " (" << r.frequency << ", " << r.period << ")\n";

found = true;

}

}

if (!found) {

std::cout << "Рейсів за вказаною станцією не знайдено.\n";

}

}

search\_by\_station.h:

#ifndef SEARCH\_BY\_STATION\_H

#define SEARCH\_BY\_STATION\_H

#include "struct\_type\_project\_5.h"

class ScheduleSystem {

public:

void searchByStation() const;

};

#endif // SEARCH\_BY\_STATION\_H

Вихідний код модулю add\_route:

add\_route.cpp:

#include "add\_route.h"

#include <iostream>

void ScheduleSystem::addRoute() {

BusRoute r;

std::cout << "Номер рейсу (ID): ";

std::cin >> r.id;

std::cin.ignore();

for (const auto& route : routes) {

if (route.id == r.id) {

std::cout << "Рейс з таким ID уже існує!\n";

return;

}

}

std::cout << "Маршрут: "; std::getline(std::cin, r.route);

std::cout << "Періодичність: "; std::getline(std::cin, r.frequency);

std::cout << "Період дії: "; std::getline(std::cin, r.period);

std::cout << "Прибуття (початкова): "; std::getline(std::cin, r.startArrive);

std::cout << "Відправлення (початкова): "; std::getline(std::cin, r.startDepart);

std::cout << "Відправлення (кінцева): "; std::getline(std::cin, r.endDepart);

std::cout << "Прибуття (кінцева): "; std::getline(std::cin, r.endArrive);

routes.push\_back(r);

std::cout << "Рейс додано.\n";

}

add\_route.h:

#ifndef ADD\_ROUTE\_H

#define ADD\_ROUTE\_H

#include "struct\_type\_project\_5.h"

class ScheduleSystem {

public:

void addRoute();

};

#endif // ADD\_ROUTE\_H

Аргументи:

1. Ознайомився з принципами побудови багатофайлових C++-проєктів.
2. Навчився створювати окремі модулі для кожної функції.
3. Створив 7 окремих .h-файлів для кожного методу класу.
4. Реалізував відповідні .cpp-файли для кожного модуля.
5. Дотримався принципу "одна функція — один модуль".
6. Уніфікував доступ до даних через структуру BusRoute.
7. Розробив архітектуру класу ScheduleSystem для обробки маршрутів.
8. Реалізував функцію додавання маршруту з валідацією.
9. Реалізував функцію видалення маршруту за ID.
10. Реалізував функцію друку всіх маршрутів.
11. Реалізував пошук маршруту за назвою станції.
12. Забезпечив збереження маршрутів у файл.
13. Забезпечив зчитування маршрутів з файлу.
14. Опрацював обробку помилок при зчитуванні файлів.
15. Впровадив обробку неправильного формату CSV.
16. Забезпечив читабельний консольний інтерфейс.
17. Реалізував інтерфейс вибору опцій меню.
18. Реалізував цикл виконання з обробкою некоректного введення.
19. Розробив окрему функцію для пошуку по маршруту.
20. Забезпечив логічну взаємодію всіх функцій через меню.
21. Вивчив техніку інкапсуляції в класах.
22. Впровадив принцип інформаційного приховування.
23. Навчився використовувати вектори STL для зберігання об'єктів.
24. Застосував стандартні бібліотеки для роботи з рядками.
25. Використовував потоки ifstream/ofstream для роботи з файлами.
26. Вивчив формат CSV і його структуру.
27. Навчився обробляти роздільники CSV (символ ;).
28. Навчився працювати з рядками за допомогою stringstream.
29. Використав засоби винятків для захисту від некоректного вводу.
30. Реалізував механізм перевірки унікальності ID.
31. Використав erase-remove idiom для видалення з вектору.
32. Реалізував перевірку, чи знайдено маршрут під час пошуку.
33. Розділив проєкт на логічні частини для зручності підтримки.
34. Вивчив переваги модульності при розробці.
35. Описав структуру BusRoute в окремому заголовковому файлі.
36. Застосував директиви препроцесора для запобігання дублям.
37. Впровадив include guards у всіх заголовкових файлах.
38. Забезпечив повторне використання заголовків без конфліктів.
39. Вивчив основи повторного використання коду.
40. Реалізував зручне головне меню через main.cpp.
41. Забезпечив узгоджене форматування виводу.
42. Створив окремий заголовковий файл для конструктора.
43. Створив тестовий файл з маршрутами для перевірки.
44. Побудував тестові кейси для всіх функцій системи.
45. Створив таблиці з описом очікуваних результатів (тест-сюїти).
46. Сформував 7 тест-сюїтів по кожному модулю.
47. Запропонував 45+ окремих тест-кейсів для перевірки.
48. Реалізував інтерактивне тестування через термінал.
49. Проводив модульне тестування окремих функцій.
50. Вивчив принципи написання інтеграційних тестів.
51. Узагальнив власні висновки щодо структури коду.
52. Визначив переваги модульного підходу над монолітним.
53. Дотримався єдиного стилю іменування та форматування.
54. Описав призначення кожної функції в коментарях.
55. Всі функції мають зрозумілі імена, відповідні дії.
56. Проєкт легко модифікувати й розширювати.
57. Легко додати нові функції в майбутньому (наприклад, фільтри).
58. Здобув практичні навички проєктування класів.
59. Зрозумів переваги роботи з вектором замість масиву.
60. Навчився тестувати вводи з невалідними значеннями.
61. Вивчив механізм повторного виклику при помилковому введенні.
62. Реалізував тестування для сценарію з порожнім файлом.
63. Реалізував тестування для файлу з некоректними рядками.
64. Сформував шаблон меню, який можна використовувати в інших ЛР.
65. Вивчив обробку порожніх, дубльованих та помилкових ID.
66. Продемонстрував самостійність у реалізації всіх частин.
67. Впровадив підключення кількох заголовків у main.cpp.
68. Забезпечив логічну ізоляцію кожної функції.
69. Навчився дотримуватися вимог POSIX до консольних програм.
70. Забезпечив повну автономність кожного модуля.
71. Забезпечив повну відсутність залежностей між модулями.
72. Оформлення коду відповідає рекомендаціям Google C++ Style.
73. Отримав навички структурованого підходу до роботи над проєктом.
74. Продемонстрував вміння організовувати великі проєкти.
75. Досяг усіх цілей лабораторної роботи та здобув ключові знання з модульної архітектури та тестування в C++.