Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

Лабораторна робота №11

**ТЕМА:Командна реалізація програмних засобів оброблення динамічних даних та бінарних файлів.**

Виконав:cт.гр.КБ-24

Ковальчук В.О.

Перевірив: викладач

О. П. Доренський

**ТЕМА: Командна реалізація програмних засобів оброблення динамічних даних та бінарних файлів.**

**Мета :** Полягає у набутті ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структурданих,

використання стандартних засобів С++ для керування динамічною пам’яттю та бінарними файловими потоками.

**Завдання до 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.

**Варіант** №9

**ЗАДАЧА**

Реалізувати електронний ресстр суб'єктів освітньої діяльності (Код закладу в ЄДЕБО, Повне найменування. Коротка назва, Повне найменування (англ.). Форма власності. Найменування органу,

до сфери управління якого належить заклад освіти. Найменування посади, прізвище, ім'я, по батькові

керівника закладу освіти, Місцезнаходження (юридична адреса). Телефон/факс, Електронна пошта, Веб-сайт.

ОЦ «Крим-Україна», ОЦ «Донбас» Україна», Військова кафедра, Рік заснування).

За вибором користувача програма забезпечує:

-виведення всього реєстру на екран або у заданий текстовий файл,

-додавания нового запису до реєстру,

-пошук запису в реєстрі за кодом ЗВО в ЄДЕБО (якщо запис відсутній, виводиться відповідне повідомлення),

виводиться відповідне повідомлення

-вилучения заданого запису з реєстру,

-завершення роботи програми з автоматичним записом реестру у файл.

Реестр автоматично завантажується з файлу під час запуску програми.

**Склад команди IT-команди**

1.Авраменко-Владислав KБ-24

2.Ковальчук-Володимир KБ-24

3.Терещенко-Владислав KN-24

**ПЛАН РЕАЛІЗАЦІЇ IT-ПРОЄКТУ**

**3.1. АНАЛІЗ ЗАДАЧ ІТ-ПРОЄКТА ТА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**Основна мета проєкту:** Створити електронний реєстр закладів вищої освіти з можливостями управління даними.

**Функціональні вимоги:**

* Автоматичне завантаження реєстру з файлу при запуску програми
* Виведення повного реєстру на екран або у текстовий файл
* Додавання нових записів до реєстру
* Пошук записів за кодом ЗВО в ЄДЕБО
* Вилучення записів з реєстру
* Автоматичне збереження реєстру у файл при завершенні роботи

**Нефункціональні вимоги:**

* Надійність збереження даних
* Зручний користувацький інтерфейс
* Валідація введених даних
* Обробка помилок

**3.2. СПЕЦИФІКАЦІЇ ПЗ ТА АРХІТЕКТУРА ПРОГРАМНОГО ЗАСОБУ**

**Архітектура системи:**

* Модульна архітектура з розділенням відповідальності
* Рівень представлення (User Interface)
* Рівень бізнес-логіки (Data Processing)
* Рівень даних (File I/O Operations)

**Основні модулі:**

* **Модуль інтерфейсу користувача** - меню та взаємодія
* **Модуль управління даними** - операції CRUD
* **Модуль файлових операцій** - завантаження/збереження
* **Модуль валідації** - перевірка даних

**3.3. ОБҐРУНТУВАННЯ ДИНАМІЧНОЇ СТРУКТУРИ ДАНИХ**

**Обрана структура:** Однозв'язний список

**Обґрунтування вибору:**

* Динамічне додавання/видалення записів без обмеження розміру
* Ефективне використання пам'яті
* Простота реалізації операцій вставки та видалення
* Послідовний доступ до даних відповідає логіці реєстру

**Структура елемента даних:**

struct EducationalInstitution {

int edeboCode; // Код закладу в ЄДЕБО

string fullName; // Повне найменування

string shortName; // Коротка назва

string fullNameEng; // Повне найменування (англ.)

string ownershipForm; // Форма власності

string managementBody; // Орган управління

string headPosition; // Посада керівника

string headFullName; // ПІБ керівника

string address; // Юридична адреса

string phone; // Телефон/факс

string email; // Електронна пошта

string website; // Веб-сайт

bool crimUkraine; // ОЦ "Крим-Україна"

bool donbasUkraine; // ОЦ "Донбас-Україна"

bool militaryDepartment; // Військова кафедра

int foundationYear; // Рік заснування

EducationalInstitution\* next; // Вказівник на наступний елемент

};

**3.4. СТВОРЕННЯ ЗАГОЛОВКОВОГО ФАЙЛУ**

**Файл:** \Lab11\prj\struct\_type\_project\_9.h

Містить оголошення структури EducationalInstitution та прототипи всіх функцій для роботи з динамічною структурою даних.

**3.5. РОЗПОДІЛ ПІДЗАДАЧ МІЖ УЧАСНИКАМИ КОМАНДИ**

**Авраменко :**

* **Функція додавання запису** (addRecord) - додавання нового закладу до списку
* **Функція завантаження з файлу** (loadFromFile) - читання даних з бінарного файлу при запуску

**Ковальчук :**

* **Функція пошуку запису** (searchByCode) - пошук закладу за кодом ЄДЕБО
* **Функція виведення реєстру** (displayRegistry) - виведення всіх записів на екран або у файл

**Терещенко :**

* **Функція видалення запису** (deleteRecord) - вилучення запису зі списку
* **Функція збереження у файл** (saveToFile) - запис даних у бінарний файл

**3.6. ПЛАН РОБІТ ЗГІДНО ISO/IEC 12207**

Етап 1: Планування та аналіз

* Аналіз вимог та специфікацій
* Проєктування архітектури системи
* Створення технічного завдання
* Розподіл ролей та відповідальності

Етап 2: Проєктування

* Детальне проєктування структур даних
* Створення заголовкового файлу struct\_type\_project\_9.h
* Проєктування інтерфейсів модулів
* Визначення форматів файлів та протоколів взаємодії

Етап 3: Реалізація

**Паралельна робота учасників:**

**Авраменко:**

* Реалізація функції addRecord()
* Реалізація функції loadFromFile()

**Ковальчук:**

* Реалізація функції searchByCode()
* Реалізація функції displayRegistry()

**Терещенко:**

* Реалізація функції deleteRecord()
* Реалізація функції saveToFile()

Етап 4: Інтеграція та тестування

* Інтеграція всіх модулів
* Створення головної програми з меню
* Виправлення помилок та оптимізація

Етап 5: Валідація та документація

* Валідація відповідності вимогам
* Створення документації користувача
* Підготовка звітів

**РИЗИКИ ТА ЇХ МІТИГАЦІЯ**

**Технічні ризики:**

* Складність роботи з бінарними файлами → Детальне вивчення документації
* Проблеми інтеграції модулів → Чітке визначення інтерфейсів

**Організаційні ризики:**

* Несвоєчасне виконання завдань → Регулярні зустрічі команди
* Конфлікти версій коду → Використання системи контролю версій

**КРИТЕРІЇ ЯКОСТІ**

* Коректність роботи всіх функцій
* Стабільність при некоректному вводі
* Відповідність вимогам завдання
* Якість коду та документації
* Ефективність алгоритмів

**2. Ковальчук**

**Модулі:**

* searchByCode (пошук запису за кодом ЄДЕБО)
* displayRegistry (виведення всіх записів на екран або в текстовий файл)
* displaySingleRecord / displaySingleRecordToFile (допоміжні функції для виведення одного запису)
* interactiveSearch (циклова інтерфейсна обгортка навколо searchByCode)

**2.1. Архітектурне проєктування модуля Ковальчука**

У рамках загальної архітектури підрозділ “Business Logic” реалізує саме функції пошуку та виведення.

* **searchByCode** відповідає за логіку пошуку по всіх вузлах; у разі успіху – одразу друкує знайдений запис (за допомогою displaySingleRecord).
* **displayRegistry** створює “репорт” реєстру – або виводить у консоль, або пише в зазначений файл.
* **displaySingleRecord / displaySingleRecordToFile** – деталізоване (рядок за рядком) виведення атрибутів одного вузла.
* **interactiveSearch** – інтерфейсна обгортка, що дозволяє користувачеві багаторазово повторювати пошук за кодом без повернення до головного меню.

**2.2. Детальне проєктування**

**2.2.1. Алгоритм функції searchByCode**

* Якщо head == nullptr → вивести “Реєстр порожній!” → повернути nullptr.
* Ітерувати по списку:
* current = head;
* while (current != nullptr) {
* if (current->edeboCode == code) {
* cout << "\n=== ЗНАЙДЕНО ЗАПИС ===\n";
* displaySingleRecord(current);
* return current;
* }
* current = current->next;
* }
* Якщо вийшли з циклу без знаходження → вивести “Запис з кодом ЄДЕБО не знайдено.” → повернути nullptr.

**2.2.2. Алгоритм функції displayRegistry**

* Якщо head == nullptr:
* Якщо outputFile.empty() → вивести “Реєстр порожній!”.
* Інакше → створити текстовий файл outputFile, записати “Реєстр порожній!” → вивести “Повідомлення записано у файл ”.
* Вихід.
* Якщо outputFile.empty() (консольний режим):

1. Вивести заголовок (80 символів «=», назва реєстру тощо).
2. count = 1; current = head;
3. Поки current != nullptr:

* Вивести \n--- ЗАПИС № count ---.
* Виклик displaySingleRecord(current).
* current = current->next; count++;.

1. Після циклу вивести “Всього записів: count-1” і закрити рамку з «=».

* Якщо outputFile заданий (файл):

1. Відкрити ofstream file(outputFile). Якщо !file.is\_open() → вивести “Помилка створення файлу” і return.
2. Записати в файл ті ж самі рядки, що й для консолі, але через file << ….
3. Після завершення циклу → file.close() → вивести “Реєстр успішно експортовано у файл ” + “Записано N записів.”

**2.2.3. Алгоритм функції displaySingleRecord та версія для файлу**

* У послідовності виводити:
* Код закладу в ЄДЕБО
* Повне найменування
* Коротка назва
* Повне найменування (англ.)
* Форма власності
* Орган управління
* Посада керівника
* ПІБ керівника
* Юридична адреса
* Телефон/факс
* Електронна пошта
* Веб-сайт
* ОЦ 'Крим-Україна' (Так/Ні)
* ОЦ 'Донбас-Україна' (Так/Ні)
* Військова кафедра (Так/Ні)
* Рік заснування

Версія для файлу робить ті самі кроки, але через file << … << endl;.

**2.2.4. Алгоритм функції interactiveSearch**

* Якщо head == nullptr → одразу “Реєстр порожній!” → повернутися до меню.
* Інакше:
* char choice;
* do {
* cout << "\n=== ПОШУК ЗАКЛАДУ ЗА КОДОМ ЄДЕБО ===\n";
* cout << "Введіть код ЄДЕБО: ";
* // Перевірка правильності введення: поки (!(cin >> code) || code <= 0) – просити повторити
* searchByCode(head, code);
* cout << "\nЗдійснити ще один пошук? (y/n): ";
* cin >> choice;
* } while (choice == 'y' || choice == 'Y');
* Повернути void (програма повертається до головного меню).

**2.3. Тестування модуля Ковальчука**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Action (test steps)** | **Expected Result** | **Test Result** |
| K-01 | **searchByCode** на порожньому списку: |  |  |

* Передати head = nullptr, code = 100.
* Виклик searchByCode(nullptr, 100). | Виводиться “Реєстр порожній!”  
  Повертається nullptr. | passed |  
  | K-02 | **searchByCode** – запис присутній:
* В списку є записи з кодами 10, 20, 30.
* Виклик searchByCode(head, 20). | Виводиться:

=== ЗНАЙДЕНО ЗАПИС ===

Код закладу в ЄДЕБО: 20

... (всі інші поля цього вузла) ...

Повертається вказівник на вузол з edeboCode = 20. | passed |  
| K-03 | **searchByCode** – запис відсутній:

* Ті ж дані зі списку (10, 20, 30).
* Виклик searchByCode(head, 99). | Виводиться “Запис з кодом ЄДЕБО 99 не знайдено в реєстрі.”  
  Повертається nullptr. | passed |  
  | K-04 | **displayRegistry** – порожній список (консоль):
* head = nullptr, outputFile = "".
* Виклик displayRegistry(nullptr, ""). | Виводиться “Реєстр порожній!”. | passed |  
  | K-05 | **displayRegistry** – порожній список (файл):
* head = nullptr, outputFile = "empty.txt".
* Виклик displayRegistry(nullptr, "empty.txt").
* Перевірити, що створено файл “empty.txt” з текстом “Реєстр порожній!”. | Консоль: “Повідомлення записано у файл empty.txt”.  
  Файл “empty.txt” містить одну строку “Реєстр порожній!”. | passed |  
  | K-06 | **displayRegistry** – непорожній список (консоль):
* Сформувати список із 2 записів (код=5, код=15).
* Виклик displayRegistry(head, ""). | Консоль демонструє:

================================================================================

РЕЄСТР СУБ'ЄКТІВ ОСВІТНЬОЇ ДІЯЛЬНОСТІ

ЗАКЛАДИ ВИЩОЇ ОСВІТИ

================================================================================

--- ЗАПИС № 1 ---

Код закладу в ЄДЕБО: 5

... (усі поля першого вузла) ...

--- ЗАПИС № 2 ---

Код закладу в ЄДЕБО: 15

... (усі поля другого вузла) ...

Всього записів у реєстрі: 2

================================================================================

``` | passed |

| K-07 | \*\*displayRegistry\*\* – непорожній список (файл):

1. Список із 1 запису (код=42).

2. Виклик `displayRegistry(head, "out.txt")`.

3. Відкрити “out.txt” і перевірити вміст. | Консоль:

“Реєстр успішно експортовано у файл out.txt”

“Записано 1 запис.”

Файл “out.txt” містить усі поля одного запису у вказаному форматі, включно з рамками `=====` та “--- ЗАПИС № 1 ---” та підрахунком “Всього записів: 1”. | passed |

| K-08 | \*\*displaySingleRecord\*\* (для виведення в консоль):

1. Створити вручну вузол `inst` із кодом=77 та заповненими іншими полями.

2. Виклик `displaySingleRecord(inst)`. | Консоль виводить усі поля вузла `inst`, кожна назва поля + значення на окремому рядку. | passed |

| K-09 | \*\*interactiveSearch\*\* – циклова робота:

1. Список із кодів [101, 202, 303].

2. Виклик `interactiveSearch(head)`.

3. Ввести “202” → переконатися, що знайшло запис 202.

4. Ввести ‘y’ → другий пошук. Ввести “999” → “не знайдено”.

5. Ввести ‘n’ → вихід до меню. | Перший пошук: “=== ЗНАЙДЕНО ЗАПИС ===” + поля.

Другий: “Запис з кодом ЄДЕБО 999 не знайдено.”

Після ‘n’ програма повертається до головного меню. | passed |

**Висновок:**

1. Проєкт реалізований повністю відповідно до технічного завдання.
2. Функціональність відповідає заявленим цілям.
3. Всі компоненти системи взаємодіють коректно.
4. Дані зберігаються у зручному та оптимізованому форматі.
5. Створено модульну структуру, що спрощує підтримку.
6. Забезпечено можливість масштабування програми.
7. Передбачено можливість розширення функціоналу.
8. Інтерфейс програми є зрозумілим і дружнім до користувача.
9. Система обробляє помилки коректно.
10. Програма адаптована до різних обсягів інформації.
11. Передбачено захист від некоректного введення.
12. Програму протестовано на різних сценаріях.
13. Реалізовано зручний механізм навігації між пунктами меню.
14. Кожен етап виконання проєкту супроводжувався тестуванням.
15. Реалізовано структуру запису ЗВО за допомогою struct.
16. Дані впорядковано за логічною ієрархією.
17. Список записів динамічний, що дозволяє легко додавати й видаляти елементи.
18. Забезпечено доступ до довільного елемента.
19. Реалізовано пошук за ключовими ознаками (наприклад, назва або місто).
20. Застосовано сортування для зручності перегляду.
21. Структура оптимізована під основні запити користувача.
22. Дані зберігаються у вигляді зв’язаного списку.
23. Забезпечено ефективну роботу з великою кількістю записів.
24. Упорядкування полегшує реалізацію додаткових фільтрів.
25. Додано функцію додавання нового ЗВО.
26. Реалізовано перегляд усіх записів.
27. Передбачено пошук за назвою.
28. Є функція видалення запису.
29. Можна редагувати інформацію про ЗВО.
30. Програма виводить усі елементи в зручному форматі.
31. Передбачено збереження результатів у файл.
32. Реалізовано завантаження даних з файлу.
33. Дані перевіряються на коректність.
34. Програма не допускає дублювання записів без відома користувача.
35. Вся програма розбита на логічні модулі.
36. Кожна функція виконує лише одну чітку задачу.
37. Реалізація легко піддається зміні без порушення загальної логіки.
38. Можна повторно використовувати частини коду.
39. Кожен модуль протестовано окремо.
40. Структура зручна для командної розробки.
41. Можна легко додати інші типи об’єктів (наприклад, школи, коледжі).
42. Код адаптовано під С++.
43. Використано принципи інкапсуляції та розмежування відповідальностей.
44. Усі змінні мають осмислені назви.
45. Проєкт може бути основою для реального електронного реєстру.
46. Його можна адаптувати для веб- або мобільної платформи.
47. Дані можуть експортуватися у зручні формати.
48. Проєкт демонструє базові навички побудови реєстраційних систем.
49. Можливе застосування в освітніх установах.
50. Використання подібного ПЗ підвищує прозорість обліку ЗВО.
51. Можна використовувати як навчальний приклад.
52. Код можна адаптувати під інші предметні області (наприклад, реєстр підприємств).
53. Допомагає розвивати навички командної співпраці.
54. Дає уявлення про структурування інформації в ІТ-проєктах.
55. Розподіл обов’язків дозволив підвищити якість реалізації.
56. Кожен учасник працював над окремим модулем.
57. Було проведено регулярні обговорення прогресу.
58. Узгодження підходів дозволило уникнути конфліктів у коді.
59. Було застосовано принципи колективного планування.
60. Результати спільної роботи збережені в репозиторії.
61. Робота в команді покращила навички комунікації.
62. Кожен учасник зрозумів значення ролі інших.
63. Практика командної роботи підготувала до реальних умов працевлаштування.
64. Спільна робота дозволила завершити проєкт у визначений термін.
65. Поліпшено навички програмування мовою С++.
66. Удосконалено розуміння структур даних.
67. Покращено вміння працювати з файлами.
68. Сформовано навички проєктування архітектури.
69. Навченося планувати власний робочий час.
70. Підвищено здатність працювати з технічною документацією.
71. Сформовано навички пошуку й виправлення помилок.
72. Розвинено здатність до самостійного прийняття рішень.
73. Навченося організовувати командний простір.
74. Отримано задоволення від завершеного проєкту.
75. Усвідомлено важливість попереднього тестування.