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 ПЗ 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 КН-24

Безкровна В. В.

ПЕРЕВІРИЛА

викладачка кафедри кібербезпеки   
та програмного забезпечення

Анастасія КОВАЛЕНКО

Кропивницький – 2025

**ТЕМА: КОМАНДНА РЕАЛІЗАЦІЯ ПЗ ОБРОБЛЕННЯ ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ**

**МЕТА: набуття 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 - 1**

**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 IT-команди:**

* Безкровна Вероніка (https://github.com/odorenskyi/Bezkrovna-Veronika-KN24.git)
* Дмитренко Ярослава (https://github.com/odorenskyi/Dmytrenko-Yaroslava-KN24.git)
* Середа Марина (https://github.com/odorenskyi/Sereda-Maryna-KN24.git)

**Аналіз задач ІТ‑проєкту та вимог до ПЗ**

1. Виведення всього реєстру на екран.
2. Виведення всього реєстру в заданий текстовий файл.
3. Додавання нового запису до реєстру.
4. Пошук запису в реєстрі за заданим державним номером (якщо запис відсутній, виводиться відповідне повідомлення).
5. Вилучення заданого запису з реєстру.
6. Завершення роботи програми з автоматичним записом реєстру у файл.
7. Реєстр автоматично завантажується з файлу під час запуску програми.

Під час лабораторної роботи ми зібралися на мітинг і обговорили специфікацію ПЗ, концептуальні проєктні рішення, сформулювали й узгодили архітектуру програмного засобу, загальні алгоритми функціонування та інтерфейси модулів.

**Вибір динамічної структури даних**

Тип: список.

Обґрунтування:

* Оптимізація швидкості пошуку та простота виведення і видалення даних.
* Покращення впорядкування та доступної інформації.

**Розподіл підзадач**

**Дмитренко Ярослава**

1. Виведення всього реєстру на екран.
2. Виведення всього реєстру в заданий текстовий файл.
3. Додавання нового запису до реєстру.

**Середа Марина**

1. Пошук запису в реєстрі за заданим державним номером (якщо запис відсутній, виводиться відповідне повідомлення).
2. Вилучення заданого запису з реєстру.

**Безкровна Вероніка**

1. Завершення роботи програми з автоматичним записом реєстру у файл.
2. Автоматичне завантаження реєстру з файлу під час запуску програми.

**План роботи за стандартом ISO/IEC 12207**

1. Аналіз задач.
2. Розподіл функцій.
3. Планування роботи.
4. Реалізація модулів.
5. Інтеграція модулів.
6. Верифікація відповідності вимогам.

**Реалізація програмного модуля ModulesBezkrovna**

Проектування архітектури функції для автоматичного завантаження реєстру з файлу:

1. Відбувається відкриття файлу, який використовується як реєстр, для читання в бінарному режимі.
2. Якщо файл відкрити не вдається, виводиться повідомлення про помилку, програма завершує роботу.
3. Створюється порожній двозв'язний список типу List.
4. Ініціалізується локальна змінна current, яка вказує на поточний елемент списку.
5. Визначається розмір файлу для читання.
6. Починається цикл, який повторюється поки не буде досягнуто кінець файлу.
7. Якщо список уже містить елементи – додається новий вузол після current. Якщо список порожній – створюється head.
8. Відбувається читання даних для кожного об'єкта структури реєстра. Спочатку визначається довжина рядка, потім зчитуються самі дані.
9. Після того, як всі поля структури реєстру були прочитані, ці дані зберігаються у відповідний вузол.
10. Після завершення циклу відкритий файл закривається.
11. Якщо отриманий список не порожній, вказівник на наступний вузол поточного встановлюється в nullptr, що позначає кінець списку.
12. Функція повертає вказівник на створений список. Відбувається завершення роботи функції.

Детальне проектування функції для автоматичного завантаження реєстру з файлу:

1. Відбувається відкриття файлу filename для читання в бінарному режимі.
2. Якщо файл не відкривається, виводиться повідомлення про помилку, програма завершує роботу.
3. Виділяється пам'ять для списку list та ініціалізуються його head і tail. Потім вказівник current використовується для навігації по списку.
4. Змінна size\_of\_file зберігає загальний розмір файлу в байтах.
5. Оголошується size\_of\_string, яка буде використана для зберігання довжин рядків.
6. Починається цикл, який виконується до тих пір, поки позиція зчитування у файлі менша за size\_of\_file.
7. Для кожного запису створюється новий вузол, причому для кожного вузла налаштовуються зв'язки з попереднім та наступним вузлом.
8. Відбувається оголошення змінної data типу CarRegister
9. Для кожного об'єкта структури реєстра size\_of\_string зчитує розмір рядка, для кожного поля типу string встановлюється точний розмір, відбувається читання кожного рядка.
10. Після зчитування всіх даних для поточного елемента, ці дані записуються в data елемента списку.
11. Після завершення циклу відкритий файл закривається.
12. Якщо в списку є елементи, вказівник на наступний елемент останнього вузла встановлюється в nullptr, що позначає кінець списку, tail списку встановлюється на останній елемент.
13. Функція повертає list. Відбувається завершення роботи функції.

Проектування архітектури функції для завершення роботи програми з автоматичним записом реєстру у файл:

1. Відбувається відкриття файлу, який використовується як реєстр, для письма в бінарному режимі.
2. Якщо файл відкрити не вдається, виводиться повідомлення про помилку, програма завершує роботу.
3. Ініціалізується локальна змінна current, яка вказує на поточний елемент списку.
4. Починається цикл, який повторюється поки не буде досягнуто кінця списку.
5. Відбувається запис даних для кожного об'єкта структури реєстра. Спочатку визначається довжина рядка, потім у файл записуються самі дані.
6. Після того, як дані з поточного об'єкта структури реєстра записані, вказівник current переміщується на наступний елемент списку.
7. Після завершення циклу відкритий файл закривається. Відбувається завершення роботи функції.

Детальне проектування функції для завершення роботи програми з автоматичним записом реєстру у файл:

1. Відбувається відкриття файлу filename для письма в бінарному режимі.
2. Якщо файл не відкривається, виводиться повідомлення про помилку, програма завершує роботу.
3. Ініціалізується локальна змінна current, яка вказує на поточний елемент списку.
4. Оголошується size\_of\_string, яка буде використана для зберігання довжин рядків.
5. Починається цикл, який повторюється поки current не стане рівною nullptr.
6. Для кожного об'єкта структури реєстра size\_of\_string зчитує розмір рядка, для кожного поля типу string встановлюється точний розмір, відбувається запис кожного рядка у файл.
7. Після запису даних, вказівник current оновлюється, щоб перейти до наступного елемента списку.
8. Після завершення циклу відкритий файл закривається. Відбувається завершення роботи функції.

**Лістинг prj\_1\_Bezkrovna:**

#include <iostream>

#include <locale>

#include <string>

#include <fstream>

#include <windows.h>

#include "ModulesBezkrovna.h"

#include "ModulesDmytrenko.h"

#include "ModulesSereda.h"

using namespace std;

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

char option;

string filename;

string state\_number;

ofstream file("Register.txt", ios::binary|ios::app);

file.close();

List \*list = load\_register("Register.txt");

if (list == nullptr)

{

list = new List;

list->head = nullptr;

list->tail = nullptr;

}

cout << "Електронний реєстр автомобілів регіонального сервісного центру МВС України" << endl;

while (true)

{

cout << "\nСписок команд:" << endl;

cout << "1. Вивести реєстр на екран;\n2. Вивести реєстр у текстовий файл;\n3. Додати новий запис до реєстру;\n4. Пошук запису у реєстрі за державним номером;\n5. Видалення запису з реєстра за державним номером;\n6. Збереження реєстру;\nq. Вихід із програми.\n" << endl;

cout << "Введіть номер команди яку ви обрали і натисніть Enter: " << endl;

cin >> option;

switch(option)

{

case '1':

printRegister(list);

break;

case '2':

cout << "Введіть назву текстового файлу для виведення (не забудьте вказати тип файлу): " << endl;

cin >> filename;

printRegistryToFile(list, filename);

break;

case '3':

addNewCarRecord(list);

break;

case '4':

cout << "Введіть державний номер: " << endl;

cin >> state\_number;

search\_by\_state\_number("Register.txt", state\_number);

break;

case '5':

cout << "Введіть державний номер: " << endl;

cin >> state\_number;

if (removeByStateNumber(\*list, state\_number))

{

cout << "Запис успішно вилучено." << endl;

}

else

{

cout << "Запис з таким номером не знайдено" << endl;

}

break;

case '6':

save\_to\_file(list, "Register.txt");

cout << "Файл збережено." << endl;

break;

case 'q':

save\_to\_file(list, "Register.txt");

deleteRegistry(list);

return 0;

default:

cout << "Такої команди не існує!" << endl;

}

}

}

**ВИСНОВОК:** в цій лабораторній роботі ми набули 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.

Ми досягли мети роботи:

1. Виконавши самостійну підготовку до виконання лабораторної роботи №11 для успішного виконання завдань.
2. Охайно оформивши роботу.
3. Аналізувавши мету роботи для правильного виконання лабораторної роботи.
4. Завантаживши копію власного Git-репозиторію.
5. Повторивши застосування команд git fetch та git merge на практиці.
6. Пригадавши використання команд git add, git commit, git merge для Git.
7. Аналізувавши задачі ІТ-проєкта.
8. Аналізувавши вимоги до програмного забезпечення.
9. Визначивши найкращий вид динамічної структури для реалізації бази даних ПЗ.
10. Обґрунтувавши обраний вид динамічної структури для реалізації бази даних ПЗ.
11. Пригадавши процес створення заголовкових файлів.
12. Пригадавши синтаксис запису типу структура.
13. Пригадавши оголошення об’єкта типу структура.
14. Реалізувавши двозв’язний список.
15. Пригадавши виділення пам’яті для елементів списку.
16. Пригадавши, як відбувається доступ до елементів списку.
17. Пригадавши, як відбувається опис вказівника на елемент списку.
18. Пригадавши вказівник previous на попередній вузол у списку.
19. Пригадавши вказівник next на наступний вузол у списку.
20. Пригадавши вказівник head на перший елемент у списку.
21. Пригадавши вказівник tail на останній елемент у списку.
22. Розподіливши між учасниками команди підзадачі з реалізації операцій над динамічною структурою даних відповідно до розробленої архітектури програмного засобу.
23. Склавши план робіт з виконання ІТ-проєкта.
24. Пригадавши міжнародний стандарт ISO/IEC 12207.
25. Пригадавши процес створення статичної бібліотеки в C++.
26. Здійснивши проектування архітектури функцій розробленого модуля.
27. Здійснивши детальне проєктування функцій розробленого модуля.
28. Здійснивши конструювання функцій розробленого модуля.
29. Здійснивши тестування функцій розробленого модуля.
30. Не знайшовши в функціях помилок.
31. Пригадавши процес підключення до проекту власних бібліотек.
32. Пригадавши процес підключення простору імен std.
33. Пригадавши синтаксис функцій в мові програмування C++.
34. Пригадавши види функцій в мові програмування C++.
35. Пригадавши виведення інформації на екран за допомогою std::cout.
36. Пригадавши введення інформації на екран за допомогою std::cin.
37. Пригадавши використання бібліотеки <fstream>.
38. Пригадавши використання бібліотеки <iostream>.
39. Пригадавши роботу з файлами в C++.
40. Пригадавши використання бінарних файлових потоків в C++.
41. Пригадавши режим бінарного відкриття файлового потоку ios::binary.
42. Пригадавши режим відкриття файлу для додавання ios::app.
43. Пригадавши функцію read() для читання бінарних даних.
44. Пригадавши функцію write() для запису бінарних даних.
45. Пригадавши метод close(), який явно закриває файл.
46. Пригадавши використання вказівників в C++.
47. Навчившись використовувати tellg() для визначення, на якій позиції знаходиться курсор в потоці файлу, тобто кількість байтів від початку файлу до поточної позиції читання.
48. Навчившись використовувати seekg() для переміщення курсору в потоці введення, щоб змінити його поточну позицію для подальшого зчитування.
49. Пригадавши використання циклу while.
50. Пригадавши використання маніпуляторів в C++.
51. Пригадавши тип string в C++.
52. Пригадавши використання бібліотеки <string>.
53. Пригадавши, як здійснюється робота з рядками в C++.
54. Пригадавши використання типу char в C++.
55. Пригадавши логічне заперечення в C++.
56. Пригадавши використання оператора розгалуження if.
57. Пригадавши оператори порівняння в C++
58. Дізнавшись про тип size\_t, який використовується для представлення розмірів об'єктів у байтах.
59. Пригадавши використання умовного оператора розгалуження switch.
60. Пригадавши використання ключового слова break.
61. Пригадавши використання ключового слова nullptr.
62. Пригадавши необхідність валідації для правильної роботи коду.
63. Пригадавши підключення Windows-1251 всередині проекту.
64. Пригадавши процес зміни кодування проекту.
65. Пригадавши використання бібліотеки <locale>.
66. Пригадавши використання бібліотеки <windows.h>.
67. Пригадавши використання оператору sizeof() для визначення розміру типів даних або змінних.
68. Дізнавшись про метод resize() для зміни розміру рядка.
69. Пригадавши процес створення проєкту консольного додатка в C++.
70. Реалізувавши програмний засіб згідно з завданням.
71. Створювавши коміти на протязі всієї роботи.
72. Відправлявши коміти до репозиторію.
73. Гарно організувавши свою роботу для ефективного засвоєння інформації.
74. Аналізувавши результати своєї роботи.
75. Підсумувавши виконану роботу.

**Додаток А**

Лістинг struct\_type\_project\_1.h:

#ifndef STRUCT\_TYPE\_PROJECT\_1\_H\_INCLUDED

#define STRUCT\_TYPE\_PROJECT\_1\_H\_INCLUDED

#include <string>

struct CarRegister {

std::string last\_name; //Прізвище

std::string first\_name; //Ім'я

std::string middle\_name; //По батькові

std::string car\_brand; // Марка автомобіля

int year\_of\_manufacture; //Рік випуску

std::string date\_of\_registration; //Дата реєстрації

std::string state\_number; // Виданий державний номер

std::string notes; //Примітки

};

// Вузол списку

struct Node {

CarRegister data;

Node\* previous;

Node\* next;

};

// Двобічний список

struct List {

Node\* head = nullptr;

Node\* tail = nullptr;

};

#endif // STRUCT\_TYPE\_PROJECT\_1\_H\_INCLUDED

**Додаток Б**

Лістинг ModulesBezkrovna:

#include <iostream>

#include <string>

#include <fstream>

#include "struct\_type\_project\_1.h"

using namespace std;

List\* load\_register(string filename)

{

ifstream in(filename, ios::binary|ios::in);

if (!in)

{

cout << "Не вдалося відкрити файл: " << filename << endl;

return nullptr;

}

List\* list = new List;

list->head = nullptr;

list->tail = nullptr;

Node \*current = list->head;

in.seekg(0, ios::end);

streamsize size\_of\_file = in.tellg();

in.seekg(0, ios::beg);

size\_t size\_of\_string;

while(in.tellg() < size\_of\_file)

{

if (current != nullptr)

{

current->next = new Node;

current->next->previous = current;

current = current->next;

}

else

{

list->head = new Node;

list->head->previous = nullptr;

current = list->head;

}

CarRegister data;

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.last\_name.resize(size\_of\_string);

in.read((char\*) &data.last\_name[0], size\_of\_string);

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.first\_name.resize(size\_of\_string);

in.read((char\*) &data.first\_name[0], size\_of\_string);

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.middle\_name.resize(size\_of\_string);

in.read((char\*) &data.middle\_name[0], size\_of\_string);

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.car\_brand.resize(size\_of\_string);

in.read((char\*) &data.car\_brand[0], size\_of\_string);

in.read((char\*) &data.year\_of\_manufacture, sizeof(int));

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.date\_of\_registration.resize(size\_of\_string);

in.read((char\*) &data.date\_of\_registration[0], size\_of\_string);

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.state\_number.resize(size\_of\_string);

in.read((char\*) &data.state\_number[0], size\_of\_string);

in.read((char\*) &size\_of\_string, sizeof(size\_t));

data.notes.resize(size\_of\_string);

in.read((char\*) &data.notes[0], size\_of\_string);

current->data = data;

}

in.close();

if (list->head != nullptr)

{

current->next = nullptr;

list->tail = current;

}

return list;

}

void save\_to\_file(List\* list, string filename)

{

ofstream out(filename, ios::binary|ios::out);

if (!out)

{

cout << "Не вдалося відкрити файл: " << filename << endl; // якщо не відкрилось

return;

}

Node \*current = list->head;

size\_t size\_of\_string;

while(current != nullptr)

{

size\_of\_string = current->data.last\_name.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.last\_name[0], size\_of\_string);

size\_of\_string = current->data.first\_name.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.first\_name[0], size\_of\_string);

size\_of\_string = current->data.middle\_name.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.middle\_name[0], size\_of\_string);

size\_of\_string = current->data.car\_brand.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.car\_brand[0], size\_of\_string);

out.write((char\*) &current->data.year\_of\_manufacture, sizeof(int));

size\_of\_string = current->data.date\_of\_registration.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.date\_of\_registration[0], size\_of\_string);

size\_of\_string = current->data.state\_number.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.state\_number[0], size\_of\_string);

size\_of\_string = current->data.notes.size();

out.write((char\*) &size\_of\_string, sizeof(size\_t));

out.write((char\*) &current->data.notes[0], size\_of\_string);

current = current->next;

}

out.close();

}

Лістинг ModulesBezkrovna.h:

#ifndef MODULESBEZKROVNA\_H\_INCLUDED

#define MODULESBEZKROVNA\_H\_INCLUDED

#include <string>

#include "struct\_type\_project\_1.h"

List\* load\_register(std::string filename);

void save\_to\_file(List\* list, std::string filename);

#endif // MODULESBEZKROVNA\_H\_INCLUDED

**Додаток В**

Таблиця 1

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_11\_ load\_register |
| Назва проекта / ПЗ  Name of Project / Software | libModulesBezkrovna.а |
| Рівень тестування  Level of Testing | Модульний / Unit Testing |
| Автор тест-сьюта  Test Suite Author | Вероніка Безкровна |
| Виконавець  Implementer | Вероніка Безкровна |

Таблиця 2

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Action (test steps) | Expected Result | Test Result (passed/failed) |
| TC-1 | Виклик функції load\_register(filename), де filename – існуючий файл, який вже містить дані. | Реєстр завантажується з файлу | passed |
| TC-2 | Виклик функції load\_register(filename), де filename – неіснуючий файл. | На консоль виводиться попередження: «Не вдалося відкрити файл: Register.txt»  Реєстр не завантажується | passed |
| TC-3 | Виклик функції load\_register(filename), де filename – існуючий файл, який є пустим. | Реєстр залишається пустим | passed |

**Додаток Г**

Таблица 1

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_11\_ save\_to\_file |
| Назва проекта / ПЗ  Name of Project / Software | libModulesBezkrovna.а |
| Рівень тестування  Level of Testing | Модульний / Unit Testing |
| Автор тест-сьюта  Test Suite Author | Вероніка Безкровна |
| Виконавець  Implementer | Вероніка Безкровна |

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Action (test steps) | Expected Result | Test Result (passed/failed) |
| TC-1 | Виклик функції load\_register(filename), де filename – існуючий файл, який вже містить дані. | Реєстр завантажується з файлу | passed |
| TC-2 | Виклик функції load\_register(filename), де filename – неіснуючий файл. | На консоль виводиться попередження: «Не вдалося відкрити файл: Register.txt»  Реєстр не завантажується | passed |
| TC-3 | Виклик функції load\_register(filename), де filename – існуючий файл, який є пустим. | Реєстр залишається пустим | passed |