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”

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 КН-23

Гончаренко В.В.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Ганна Дрєєва.

Кропивницький – 2024

ЛАБОРАТОРНА РОБОТА № 11

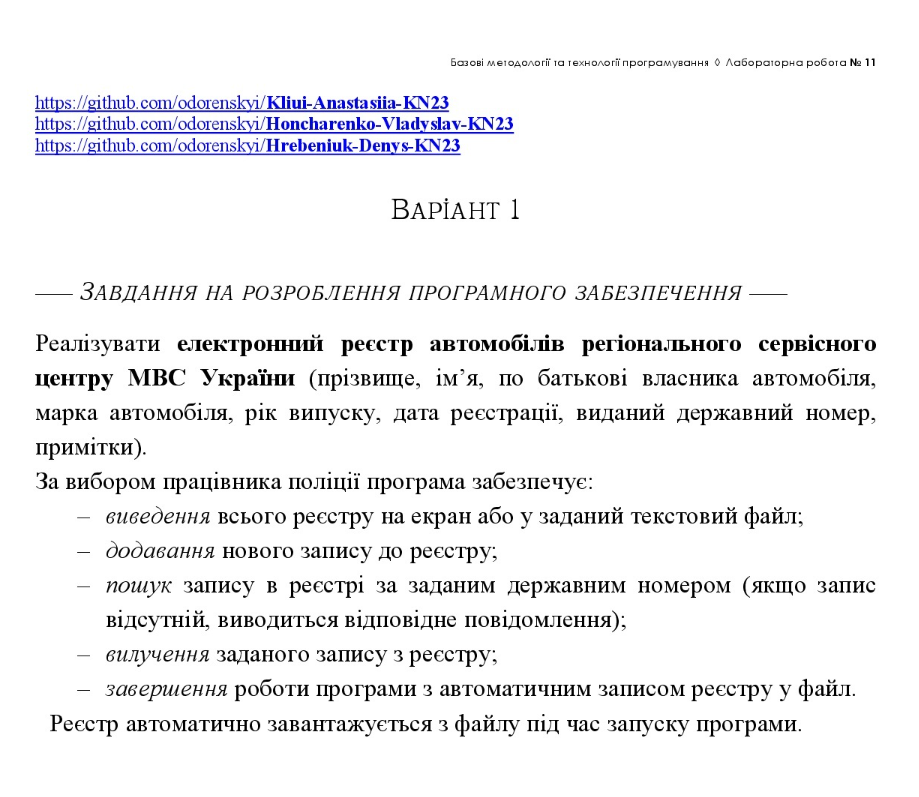
**РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ**

Мета: Набути 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 ДО ЛАБОРАТОРНОЇ РОБОТИ**

1. У складі команди ІТ-проекта розробити програмні модулі оброблення динамічної структури даних.

2. Реалізувати програмний засіб на основі розроблених командою ІТ-проекта модулів.



ОЗРОБЛЕНИЙ ПЛАН ВИКОНАННЯ ІТ-ПРОЕКТА «ЕЛЕКТРОННИЙ РЕЄСТР АВТОМОБІЛІВ»

**1. Аналіз задач ІТ-проекта та вимог до програмного забезпечення**

***Аналіз задачі проекту***

**● Ціль проекту:**

* розробка програми для автоматизації процесу ведення  електроного реєстру автомобілів у регіональному центрі МВС України.

**● Функціональні вимоги:**

1. ***Виведення*** всього реєстру на екран або у заданий текстовий файл

2. ***Додавання*** нового запису до реєстру автомобілів з такою інформацією: прізвище, ім'я, по батькові власника, марка автомобіля, рік випуску, дата реєстрації, виданий державний номер та примітки;

3. ***Пошук***запису в реєстрі за даним державним номером;

4. ***Вилучення*** заданого запису з реєстру.

5. ***Завершення***роботи програми з автоматичним записом реєстру у файл.

6.  ***Автоматичне завантаження*** з файлу під час запуску програми

**● Нефункціональні вимоги:**

1. програма повинна бути ефективною та швидкою у роботі;

2. надійність та стійкість програми до помилок та відмов.

**● Дані:**

- прізвище, ім'я, по батькові власника автомобіля;

- марка автомобіля, рік випуску, дата реєстрації;

- виданий державний номер;

- примітки.

***Вимоги до програмного забезпечення***

**●Інтерфейс користувача:**

- інтуїтивно зрозумілий та зручний для введення та виведення інформації.

**● Продуктивність:**

- швидке виконання запитів, оптимізація для великих обсягів даних.

**● Сумісність:**

- підтримка різних операційних систем.

**● Масштабованість:**

- легкість оновлення та розширення функціоналу.

**● Технічні обмеження:**

- врахування можливостей використовуваного обладнання та програмного середовища.

**● Документація:**

- повна документація коду

**2. Специфікації ПЗ, концептуальні проектні рішення, архітектура програмного засобу, загальні алгоритми функціонування та інтерфейси модулів**

***Специфікація ПЗ***

**Обговорення щодо:**

1) вимог до програмного забезпечення;

2) вибрання типу динамічної структури даних (список, стек, черга, дерево) для реалізації бази даних ПЗ;

3) створення заголовочного файлу;

4) розподілити підзадачі реалізації операції над динамічною структурою даних;

5) складання плану робіт проекту з урахуванням стандарту ISO/IEC 12207

***Концептуальні проектні рішення***

**1) Модульна архітектура**

• розподіл системи на незалежні модулі для спрощення розробки та тестування:

• модуль виведення реєстру - відповідає за виведення всього реєстру на екран або у заданий текстовий файл.

• модуль додавання нового запису - відповідає за додавання нового запису про автомобіль до реєстру.

• модуль пошуку за державним номером - відповідає за пошук запису в реєстрі за даним державним номером. Якщо запис не буде знайдено, модуль виведе відповідне повідомлення.

• модуль вилучення запису - відповідає за вилучення заданого запису з реєстру.

• модуль автоматичного запису реєстру у файл - відповідає за автоматичний запис у файл під час завершення роботи програми.

**2) Використання списку для реалізації електронного реєстру автомобілів**

• оптимізація швидкості пошуку, вставки та видалення даних

• поліпшення впорядкувння та доступної інформації

***Архітектура програмного засобу***

**1)Модуль управління даними**

**●** Основною структурою даних, є структура **CarRecord**

**●** Операції:

- виведення реєстру: функція для виведення всього реєстру на екран або у заданий текстовий файл. Ця операція дозволяє користувачу переглядати всі дані, що зберігаються у реєстрі.

- додавання нового запису: функція для додавання нового запису про автомобіль до реєстру. Ця операція дозволяє користувачу ввести дані про новий автомобіль та додати його до реєстру.

- пошук за державним номером: функція для пошуку запису в реєстрі за заданим державним номером. Ця операція дозволяє знайти інформацію про автомобіль за його номером.

- вилучення  запису: функція для вилучення заданого запису з реєстру. Ця операція дозволяє користувачу вилучити інформацію про автомобіль з реєстру.

- завершення: функція для завершення роботи програми з автоматичним записом реєстру у файл. Під час завершення роботи програми всі дані про автомобілі автоматично записуються у файл, щоб забезпечити збереження інформації.

- автоматичне завантаження: під час запуску програми, автоматично завантажує реєстр автомобілів з файлу. Ця операція дозволяє користувачу продовжувати роботу з програмою без необхідності вручного завантаження даних.

**2) *Інтерфейс користувача***

• Консольний інтерфейс: простий текстовий інтерфейс, що дозволяє користувачу вводити команди для управліня записами;

• Взаємодія з користувачем: чіткі інструкції та запити для забезпечення правильного введення даних;

***Загальні алгоритми функціонування***

1.**Виведення всього реєстру на екран або у заданий текстовий файл:**

- проходження через всі записи в реєстрі;

- виведення кожного запису на екран або збереження у файл.

**2. Додавання нового запису до реєстру:**

- перевірка наявності даних про автомобіль;

- введення нових даних;

- збереження нового запису в реєстрі.

**3. Пошук запису в реєстрі за даним державним номером:**

- введення державного номера для пошуку;

- пошук в реєстрі за державним номером;

- виведення результату пошуку або відповідного повідомлення, якщо запис не знайдено.

**4. Вилучення заданого запису з реєстру:**

- введення даних для ідентифікації запису, який потрібно вилучити;

- пошук запису в реєстрі;

- вилучення знайденого запису.

**5. Завершення роботи програми з автоматичним записом реєстру у файл:**

- збереження всіх змін у реєстрі у файл.

-завершення виконання програми.

**6. Автоматичне завантаження реєстру з файлу під час запуску програми:**

- перевірка наявності файлу з реєстром;

- якщо файл існує, завантаження даних з файлу у пам'ять програми;

- якщо файл відсутній, створення порожнього реєстру.

**3.  Обгрунтування вид динамічної структури даних для реалізації бази даних ПЗ, складових її елементів (поля структури і їх типи).**

Для реалізації бази даних програмного забезпечення «Електронний реєстр автомобілів» у мові програмування C++ було обрано двонапрямлений (двозв’язний)  список.

**Обґрунтування цього вибору наступне:**

**1) Швидкий доступ:**

Двонапрямлені списки дозволяють швидше переміщатися як вперед, так і назад по списку. Це важливо для операцій, таких як пошук та видалення записів за державним номером, де може знадобитися звернення до попередніх записів.

**2) Ефективність додавання та вилучення:**

Додавання та видалення елементів у двонапрямленому списку може бути ефективними, оскільки не потрібно переміщатися через весь список для зміни посилань.

**3) Простота реалізації:**

Двонапрямлені списки є достатньо простими для реалізації та керуються досить зрозумілими правилами, що полегшує їх розуміння та використання.

**4) Зручність використання:**

Для операцій, таких як виведення всього реєстру на екран або у заданий текстовий файл, двонапрямлений список також може бути зручним, оскільки його можна ітерувати у будь-якому напрямку.

**5) Навчальний аспект:**

Часто використовується у курсах зі структур даних. Відповідно, він ідеально підходить для проектів, спрямованих на вивчення студентами фундаментальних принципів роботи структур даних.

**Складові елементи структури**

**1) Прізвище власника автомобіля:**

- поле структури: lastName

- тип даних: string

**2) Ім'я власника автомобіля:**

- поле структури: firstName

- тип даних: string

**3) По батькові власника автомобіля:**

- поле структури: middleName

- тип даних: string

**4)  Марка автомобіля:**

- поле структури: carBrand

- тип даних: string

**5) Рік випуску автомобіля:**

- поле структури: year

- тип даних: int

**6) Дата реєстрації автомобіля:**

- поле структури: registrationDate

- тип даних: string

**7) Державний номер автомобіля:**

- поле структури: licensePlate

- тип даних: string

**8) Примітки:**

- поле структури: notes

- тип даних: string

**4. Створення заголовочного файлу**

**Задача:** створити заголовочний файл struct\_type\_project\_N.h (N ― номер варіанта завдання) з описом елементів динамічної структури даних мовою С++

**5. Розподіл завдань між учасниками команди підзадачі з реалізації операцій над динамічною структурою даних відповідно до розробленої архітектури програмного засобу.**

**Клюй Анастасія**

1) ***виведення*** всього реєстру на екран або у заданий текстовий файл: реалізація функції для виведення вмісту дерева на екран чи у текстовий файл.

2) ***додавання*** нового запису до реєстру: розробка методу, який дозволяє користувачу вводити новий запис, і додає цей запис до двійкового дерева пошуку. Це включає збір даних від користувача і створення нового вузла в дереві.

**Гончаренко Владислав**

1) ***пошук*** запису в реєстрі за даним державним номером: розробка функції пошуку, яка дозволяє користувачу ввести державний номер і знаходить відповідний запис у дереві, виводячи інформацію про автомобіль або повідомлення про те, що автомобіль не знайдено.

2) ***автоматичне*** завантаження реєстру з файлу під час запуску програми:  розробка механізму для читання даних з файлу і відновлення структури дерева при запуску програми, забезпечуючи коректне відновлення стану реєстру.

**Гребенюк Денис**

1) ***вилучення***заданого запису з реєстру:  реалізація методу видалення запису за введеним користувачем державним номером, що вимагає впевненого видалення вузла з дерева та балансування його після видалення.

2) ***завершення*** роботи програми з автоматичним записом реєстру у файл:  розробка методу, що забезпечує збереження всіх даних у файл при закритті програми та їх відновлення при наступному запуску.

**Лістинг модуля .h :**

// modules\_honcharenko.cpp

#include <iostream>

#include <fstream>

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

using namespace std;

void saveRegisterData(List \*list, string registerFileName) {

ofstream registerFile(registerFileName, ios::binary | ios::out);

if (!registerFile) {

cerr << "Не вдалось відкрити " << registerFileName << " для того, щоб зберегти реєстр!" << endl;

return;

}

Node \*current = list->head;

size\_t stringSize;

while (current != nullptr) {

stringSize = current->data.lastName.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.lastName[0], stringSize);

stringSize = current->data.firstName.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.firstName[0], stringSize);

stringSize = current->data.middleName.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.middleName[0], stringSize);

stringSize = current->data.carBrand.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.carBrand[0], stringSize);

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

stringSize = current->data.registrationDate.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.registrationDate[0], stringSize);

stringSize = current->data.licensePlate.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.licensePlate[0], stringSize);

stringSize = current->data.lastName.size();

registerFile.write((char\*) &stringSize, sizeof(size\_t));

registerFile.write((char\*) &current->data.lastName[0], stringSize);

current = current->next;

}

registerFile.close();

}

void searchCarInRegister(List\* list, string stateLicensePlate) {

Node\* current = list->head;

while (current != nullptr) {

if (current->data.licensePlate == stateLicensePlate) {

cout << "Прізвище: " << current->data.lastName << endl;

cout << "Ім'я: " << current->data.firstName << endl;

cout << "По-батькові: " << current->data.middleName << endl;

cout << "Модель авто: " << current->data.carBrand << endl;

cout << "Рік: " << current->data.year << endl;

cout << "Дата регестрації: " << current->data.registrationDate << endl;

cout << "Номер авто: " << current->data.licensePlate << endl;

cout << "Нотатки: " << current->data.notes << endl;

return;

}

current = current->next;

}

cerr << "Автомобіль з державним номером " << stateLicensePlate << " не знайдено в реєстрі." << endl;

}

**Лістинг консольного застосунку:**

#include <iostream>

#include <ctime>

#include <fstream>

#include <conio.h>

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

#include "ModulesHrebeniuk.h"

#include "ModulesHoncharenko.h"

#include "ModulesKlui.h"

#define REGISTER\_NAME "register\_database"

using namespace std;

void displayMenu() {

system("cls");

cout << " ======================= Реєстр МВС України ========================= " << endl;

cout << "| 1. Додати автомобіль у реєстр |" << endl;

cout << "| 2. Пошук автомобіля за номером |" << endl;

cout << "| 3. Видалення автомобіля з реєстру за номерним знаком |" << endl;

cout << "| 4. Вивести усі дані з реєстру |" << endl;

cout << "| 5. Вийти з програми |" << endl;

cout << " ==================================================================== " << endl;

cout << "Виберіть опцію: ";

}

void exitApp(List \*list) {

delete list->head;

delete list->tail;

delete list;

exit(0);

}

int main()

{

system("chcp 1251 & cls");

List \*list = loadRegisterData("register\_database");

if (list == nullptr) {

list = new List;

list->head = nullptr;

list->tail = nullptr;

}

int choice;

do {

displayMenu();

cin >> choice;

cin.ignore();

system("cls");

switch (choice) {

case 1: {

addCarInRegister(list);

saveRegisterData(list, REGISTER\_NAME);

break;

}

case 2: {

string searchLicensePlate;

cout << "Введіть номер авто для пошуку: ";

cin >> searchLicensePlate;

if (searchLicensePlate.empty() || searchLicensePlate.size() > 10) {

cout << "Помилка: номерний знак не може бути порожнім та його максимальна довжина повинна бути не більше 10 символів!" << endl;

break;

}

searchCarInRegister(list, searchLicensePlate);

break;

}

case 3: {

string removeLicensePlate;

cout << "Введіть номер авто для видалення: ";

cin >> removeLicensePlate;

if (removeLicensePlate.empty() || removeLicensePlate.size() > 10) {

cout << "Помилка: номерний знак не може бути порожнім та його максимальна довжина повинна бути не більше 10 символів!" << endl;

break;

}

removeCarFromRegister(list, removeLicensePlate);

saveRegisterData(list, REGISTER\_NAME);

break;

}

case 4: {

outputRegisterData(list);

break;

}

case 5: {

cout << "До побачення!" << endl;

exitApp(list);

break;

}

default:

cout << "Невірний вибір. Спробуйте ще раз." << endl;

}

cout << "Продовжити роботу застосунку? (y/n): ";

choice = getch();

} while (choice == 'Y' || choice == 'y');

exitApp(list);

return 0;

}

**Висновок:** ми зкомандою успішно розробили програму для управління реєстром автомобілів, що включає модуль виведення реєстру, який дозволяє користувачеві переглядати інформацію про всі автомобілі на екрані або в текстовому файлі. Модуль додавання нового запису дозволяє легко додавати нові записи про автомобілі до реєстру, щоб забезпечити актуальність даних. Модуль пошуку за державним номером забезпечує швидкий доступ до інформації про конкретний автомобіль шляхом пошуку за його номерною плиткою. В разі відсутності запису з вказаним номером, користувач отримає відповідне повідомлення. Модуль вилучення запису дає можливість користувачеві видаляти конкретні записи з реєстру, що є корисним для збереження актуальності даних. Крім того, команда реалізувала модуль автоматичного запису реєстру у файл, що дозволяє автоматично зберігати дані при завершенні роботи програми, забезпечуючи надійну збереженість і доступність інформації для майбутнього використання. Загалом, програма виконує свої завдання ефективно і надійно, допомагаючи в управлінні та підтримці реєстру автомобілів.