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

Чубенко Ігор

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Дрєєва Г. М.

Кропивницький – 2024

**Лабораторна робота №11**

***Тема*:** Реалізація програмних засобів оброблення динамічних структур даних та бінарних файлів

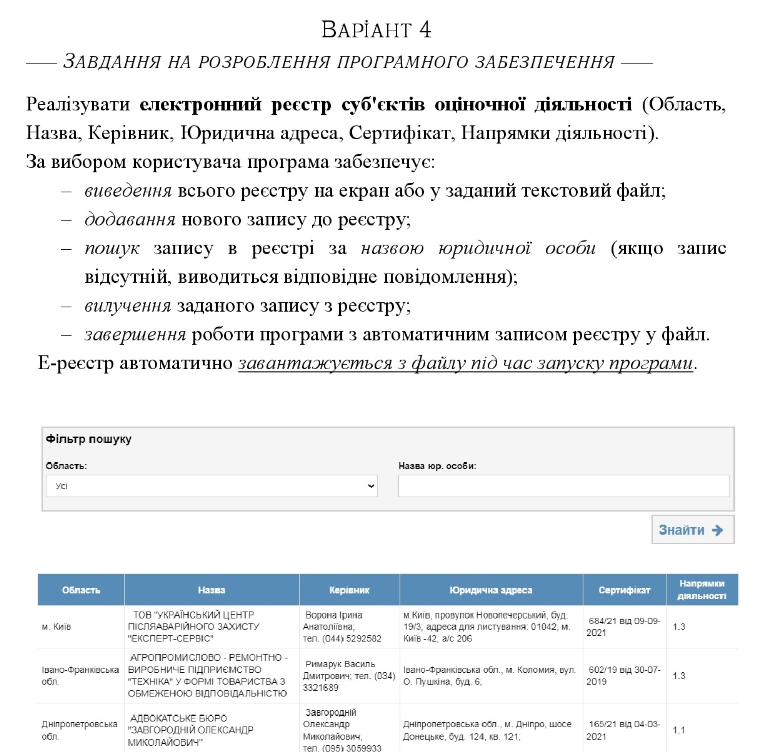
***Мета роботи:*** Набуття ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів С++ для керування динамічною пам’яттю та бінарними файловими потоками.

**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.

**Варіант 4**



**Склад команди IT-проекта:** Чубенко Ігор, Горбенко Катерина, Кобзар Миколa.

**Результати обговорення**

*1.*    *Аналіз задач лабораторної роботи №11 та вимог до програмного забезпечення:*

·       Програма повинна реалізовувати електронний реєстр суб'єктів оціночної діяльності з полями: Область, Назва, Керівник, Юридична адреса, Сертифікат, Напрямки діяльності.

·       Програма має забезпечувати функції: виведення реєстру, додавання запису, пошук запису за назвою, вилучення запису, автоматичне збереження і завантаження реєстру з файлу.

·       Програма має читатися українською мовою та вміти користуватися нею

*2.*    *Обговорення специфікацій ПЗ, архітектури, алгоритмів та інтерфейсів:*

·       Програма буде реалізована на C++ з використанням динамічних структур даних.

·       Основні модулі: структура даних для запису реєстру, функції для операцій над реєстром (додавання, пошук, вилучення, виведення тощо), інтерфейс користувача з меню вибору дій.

·       Дані будуть зберігатися у текстовому файлі.

*3.*    *Визначення динамічної структури даних для реалізації бази даних ПЗ:*

·       Для реалізації е-реєстру доцільно використати векторну динамічну структуру даних.Ця структура визначена в стандартній бібліотеці C++ (<vector>), і вона містить об'єкти типу SubjActivity. Вектор дозволяє динамічно змінювати розмір інформаційного контейнера, що зберігає об'єкти SubjActivity, і автоматично керує пам'яттю для цих об'єктів.

·       Кожен елемент списку буде представляти запис реєстру з полями: Область (рядок), Назва (рядок), Керівник (рядок), Юридична адреса (рядок), Сертифікат (рядок), Напрямки діяльності (масив рядків).

*4.*    *Створення заголовкового файлу з описом елемента структури даних:*

файл struct\_type\_project\_4.h

5. Архітектура ПЗ

* modulka
* ModulesHorbenko.h (Головний файл  Горбенко)
* ModulesKobzar.h(Головний файл Кобзаря)
* ModulesChubenko.h(Головний файл Чубенко)
* struct\_type\_project\_4.h(Заголовковий файл який містить структуру проекту)
* libModulesHorbenko.a
* libModulesKobzar.a
* libModulesChubenko.a
* ModulesHorbenko (Модулі розроблені Горбенко )
* ModulesKobzar (Модулі розроблені Кобзарем)
* ModulesChubenko (Модулі розроблені Чубенко)
* Test (Наша программа в якій відбувається виклик модулів)

6.    Розподіл підзадач з реалізації операцій над структурою даних. Кожен учасник команди реалізує дві функції для роботи зі списком :

·       Додавання нового запису в список (Чубенко Ігор)

·       Пошук запису за назвою(Горбенко)

·       Вилучення запису зі списку(Чубенко)

·       Виведення списку на екран (Кобзар Микола)

·       Збереження списку у файл (Горбенко)

7.    Складання плану робіт з виконання лабораторної :

·       Розробка структури даних, функцій для роботи з нею

·       Реалізація інтерфейсу користувача(меню с викликом функцій)

·       Тестування і налагодження програми

·       Оформлення документації

Тестування:

1. searchRecord

Позитивні тести

1. Пошук існуючого запису
   * Вхідні дані: Ввести існуючий запис (наприклад, ім'я або ID).
   * Очікуваний результат: Успішний пошук, запис знайдений та відображений.
2. Пошук з особливими символами
   * Вхідні дані: Ввести запис, що містить особливі символи (наприклад, "J@hn D\*e").
   * Очікуваний результат: Успішний пошук, запис знайдений та відображений.

Негативні тести

1. Пошук неіснуючого запису
   * Вхідні дані: Ввести запис, який не існує в системі.
   * Очікуваний результат: Повідомлення про те, що запис не знайдено.
2. Пошук з порожнім запитом
   * Вхідні дані: Ввести порожній запит.
   * Очікуваний результат: Повідомлення про те, що запит не може бути порожнім.

2. saveRegistry

Позитивні тести

1. Збереження з кількома записами
   * Вхідні дані: Реєстр з кількома записами.
   * Очікуваний результат: Успішне збереження реєстру, всі записи збережені коректно.
2. Збереження порожнього реєстру
   * Вхідні дані: Порожній реєстр.
   * Очікуваний результат: Успішне збереження порожнього реєстру без помилок.

Негативні тести

1. Збереження з недоступною назвою файлу
   * Вхідні дані: Назва файлу, яка недоступна або використовується системою (наприклад, "C:/System/registry.txt").
   * Очікуваний результат: Повідомлення про помилку збереження через недоступність назви файлу.
2. Збереження з некоректними даними
   * Вхідні дані: Реєстр, що містить некоректні або неповні дані (наприклад, відсутність обов'язкових полів).
   * Очікуваний результат: Повідомлення про помилку збереження через некоректні дані.

**Лістинг ModulesChubenko**

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <algorithm>

#include "ModulesChubenko.h"

using namespace std;

extern std::vector<SubjActivity> registry;

const string defaultFilename = "registry.txt";

void addRecord() {

    // Завантаження реєстру з файлу перед додаванням нового запису

    ifstream fileIn(defaultFilename);

    if (fileIn.is\_open()) {

        registry.clear(); // Очищуємо вектор перед завантаженням

        SubjActivity subj;

        while (getline(fileIn, subj.region, ',')) {

            getline(fileIn, subj.name, ',');

            getline(fileIn, subj.manager, ',');

            getline(fileIn, subj.address, ',');

            getline(fileIn, subj.certificate, ',');

            getline(fileIn, subj.activities);

            registry.push\_back(subj);

        }

        fileIn.close();

    } else {

        cerr << "Файл не знайдено: " << defaultFilename << endl;

        return;

    }

    // Додавання нового запису до реєстру

    SubjActivity newSubj;

    cout << "Введіть область: ";

    getline(cin, newSubj.region);

    cout << "Введіть назву: ";

    getline(cin, newSubj.name);

    // Переконайтеся, що назва унікальна

    auto it = find\_if(registry.begin(), registry.end(), [&](const SubjActivity& s) {

        return s.name == newSubj.name;

    });

    if (it != registry.end()) {

        cout << "Запис з такою назвою вже існує!" << endl;

        return;

    }

    cout << "Введіть керівника: ";

    getline(cin, newSubj.manager);

    cout << "Введіть юридичну адресу: ";

    getline(cin, newSubj.address);

    cout << "Введіть сертифікат: ";

    getline(cin, newSubj.certificate);

    cout << "Введіть напрямки діяльності: ";

    getline(cin, newSubj.activities);

    registry.push\_back(newSubj);

    cout << "Запис успішно додано!" << endl;

    // Збереження оновленого реєстру у файл після додавання нового запису

    ofstream fileOut(defaultFilename);

    if (!fileOut.is\_open()) {

        cerr << "Помилка при відкритті файлу для запису: " << defaultFilename << endl;

        return;

    }

    // Запис даних у файл

    for (const auto& subj : registry) {

        fileOut << subj.region << ","

               << subj.name << ","

               << subj.manager << ","

               << subj.address << ","

               << subj.certificate << ","

               << subj.activities << "\n";

    }

    fileOut.close();

}

void removeRecord() {

    std::string name;

    std::cout << "Введіть назву юридичної особи для видалення: ";

    std::getline(std::cin, name);

    auto it = std::find\_if(registry.begin(), registry.end(), [&](const SubjActivity& subj) {

        return subj.name == name;

    });

    if (it != registry.end()) {

        registry.erase(it);

        std::cout << "Запис успішно видалено!" << std::endl;

        saveRegistry("registry.txt");

    } else {

        std::cout << "Запис не знайдено." << std::endl;

    }

}

**Лістинг ModulesChubenko.h**

#ifndef MODULESCHUBENKO\_H\_INCLUDED

#define MODULESCHUBENKO\_H\_INCLUDED

#include <string>

#include <vector>

#include "struct\_type\_project\_4.h"

void loadRegistry(const std::string& filename);

void saveRegistry(const std::string& filename);

void addRecord();

void removeRecord();

#endif // MODULESCHUBENKO\_H\_INCLUDED

**Висновок**

У даній лабораторній роботі ми створили програму для управління реєстром юридичних осіб. Вона базується на динамічній структурі даних std::vector, що дозволяє зручно та ефективно керувати реєстром незалежно від його розміру.

Використання std::vector дозволяє нам гнучко маніпулювати реєстром: додавати нові записи, видаляти існуючі, змінювати інформацію вже наявних записів. Це особливо корисно у випадках, коли реєстр постійно оновлюється або використовується в динамічному середовищі.

Конкретно мною, були реалізовані такі завдання:

Додавання нового запису в список (Чубенко Ігор): Завдяки функції addRecord() користувач може зручно ввести дані для нового запису юридичної особи та додати їх до реєстру. Це дозволяє розширити можливості програми, забезпечуючи можливість додавання нових записів без необхідності втручання вихідного коду.

Вилучення запису зі списку (Чубенко): Функція removeRecord() дозволяє користувачу видаляти записи з реєстру за їх назвою. Це полегшує управління реєстром, забезпечуючи швидке та ефективне видалення непотрібних записів.

Додавання цих функцій розширило можливості програми, забезпечивши користувачеві більше контролю над реєстром та зробивши його керування більш зручним і ефективним.Крім того, наша програма має інтуїтивно зрозумілий інтерфейс для користувача, який включає в себе меню з різними опціями взаємодії з реєстром. Це робить її зручною у використанні для широкого кола користувачів.

Програма також надає можливість збереження змін до файлу, що забезпечує збереження даних між різними сеансами використання програми.

У результаті цієї лабораторної роботи ми отримали не лише технічний досвід у роботі з динамічними структурами даних, але й збільшили нашу розуміння того, як ефективно управляти даними в програмах, що мають змінний обсяг інформації.