Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: БМТП

Лабораторна робота №11

РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ

Виконав: студент групи КН-22
Фідря М.О.
Перевірив: викладач
Собінов Олександр Георгійович

Кропивницький 2023

МЕТА полягає у набутті ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів C++ для керування динамічною пам’яттю та бінарними файловими потоками

ЗАВДАННЯ:

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

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

Варіант 14

— Завдання на розроблення програмного забезпечення —

Реалізувати **електронний реєстр суб'єктів оціночної діяльності** (Область, Назва, Керівник, Юридична адреса, Сертифікат, Напрямки діяльності).

За вибором користувача програма забезпечує:

- *виведення* всього реєстру на екран або у заданий текстовий файл;
- *додавання* нового запису до реєстру;
- *пошук* запису в реєстрі за *назвою юридичної особи* (якщо запис відсутній, виводиться відповідне повідомлення);
- *вилучення* заданого запису з реєстру;
- *завершення* роботи програми з автоматичним записом реєстру у файл.

Е-реєстр автоматично *завантажується з файлу під час запуску програми*.

Алгоритм виконання цієї програми:

1. Створити об'єкт класу Registry з використанням імені файлу для збереження реєстру.
2. Вивести меню опцій для користувача.
3. Зчитати вибір користувача.
4. За вибором користувача виконати наступні дії:
   - Якщо вибрано опцію "Вивести реєстр":
   - Викликати функцію printRegistry об'єкту Registry для виведення реєстру на екран або у заданий текстовий файл.
   - Якщо вибрано опцію "Додати запис":

- Запросити в користувача необхідну інформацію про новий запис (область, назва, керівник, юридична адреса, сертифікат, напрямки діяльності).
- Створити об'єкт типу Subject зі зчитаною інформацією.
- Викликати функцію addSubject об'єкту Registry для додавання нового запису до реєстру.

● Якщо вибрано опцію "Пошук за назвою юридичної особи":

- Запросити в користувача назву юридичної особи, яку потрібно знайти.
- Викликати функцію searchByName об'єкту Registry для пошуку запису за назвою юридичної особи.

● Якщо вибрано опцію "Вилучити запис":

- Запросити в користувача назву юридичної особи, яку потрібно вилучити.
- Викликати функцію removeSubject об'єкту Registry для вилучення запису з реєстру.

● Якщо вибрано опцію "Завершити роботу":

- Викликати функцію saveRegistry об'єкту Registry для збереження реєстру в файл.
- Вивести повідомлення про завершення роботи програми та збереження реєстру.
- Завершити виконання програми.

● Якщо вибрано невірну опцію:

- Вивести повідомлення про невірний вибір та пропонувати спробувати ще раз.
- Повернутися до кроку 2 та продовжити взаємодію з користувачем до завершення роботи програми.

Лістинг struct_type_project_14.h

```
#ifndef STRUCT_TYPE_PROJECT_14_H
#define STRUCT_TYPE_PROJECT_14_H

#include <iostream>
#include <fstream>
#include <vector>
#include <string>
```

```
// структура що описує суб'єкт оціночної діяльності
struct Subject {
    std::string region;
    std::string name;
    std::string manager;
    std::string address;
    std::string certificate;
    std::vector<std::string> activities;
};

class Registry {
private:
    std::vector<Subject> subjects;
    std::string filename;

public:
    // конструктор
    Registry(const std::string& filename);

    // функція для завантаження реєстру з файлу
    void loadRegistry();

    // функція для збереження реєстру у файл
    void saveRegistry();

     // функція для виведення реєстру на екран або у заданий текстовий файл
         void printRegistry(std::ostream& output = std::cout) const;

       // функція для додавання нового запису до реєстру
    void addSubject(const Subject& subject);

     // функція для пошуку запису в реєстрі за назвою юридичної особи
```

```
        void searchByName(const std::string& name) const;

       // функція для вилучення заданого запису з реєстру
    void removeSubject(const std::string& name);
};

#endif // STRUCT_TYPE_PROJECT_14_H
```

Лістинг struct_type_project_14.cpp

```
#include "struct_type_project_14.h"

Registry::Registry(const std::string& filename)
    : filename(filename)
{
    loadRegistry();
}

void Registry::loadRegistry()
{
    std::ifstream file(filename);
    if (file.is_open()) {
        subjects.clear();
        std::string line;
        while (std::getline(file, line)) {
            Subject subject;
            subject.region = line;
            std::getline(file, subject.name);
            std::getline(file, subject.manager);
            std::getline(file, subject.address);
            std::getline(file, subject.certificate);
            std::string activities;
            std::getline(file, activities);
                // розділити напрямки діяльності за комами і зберегти у вектор
            size_t pos = 0;
```

```
            while ((pos = activities.find(',')) !=
std::string::npos) {
                                std::string activity =
activities.substr(0, pos);

subject.activities.push_back(activity);
                activities.erase(0, pos + 1);
            }

subject.activities.push_back(activities);  //  додати
останній напрямок діяльності
            subjects.push_back(subject);
        }
        file.close();
    }
}

void Registry::saveRegistry()
{
    std::ofstream file(filename);
    if (file.is_open()) {
        for (const Subject& subject : subjects) {
            file << subject.region << "\n";
            file << subject.name << "\n";
            file << subject.manager << "\n";
            file << subject.address << "\n";
            file << subject.certificate << "\n";
                        for  (size_t  i  =  0;  i  <
subject.activities.size() - 1; ++i) {
                    file << subject.activities[i] <<
",";
            }
                file << subject.activities.back() <<
"\n";
        }
        file.close();
    }
}
```

```
void  Registry::printRegistry(std::ostream&  output)
const
{
    for (const Subject& subject : subjects) {
          output << "Region: " << subject.region <<
"\n";
        output << "Name: " << subject.name << "\n";
         output << "Manager: " << subject.manager <<
"\n";
         output << "Address: " << subject.address <<
"\n";
                    output  <<  "Certificate:  "  <<
subject.certificate << "\n";
        output << "Activities: ";
                      for  (size_t  i  =  0;  i  <
subject.activities.size() - 1; ++i) {
            output << subject.activities[i] << ", ";
        }
        output << subject.activities.back() << "\n";
        output << "-------------------------\n";
    }
}

void Registry::addSubject(const Subject& subject)
{
    subjects.push_back(subject);
}

void Registry::searchByName(const std::string& name)
const
{
    bool found = false;
    for (const Subject& subject : subjects) {
        if (subject.name == name) {
            std::cout << "Subject found:\n";
                       std::cout  <<  "Region:  "  <<
subject.region << "\n";
```

```
            std::cout << "Name: " << subject.name <<
"\n";
                        std::cout  <<  "Manager:  "  <<
subject.manager << "\n";
                        std::cout  <<  "Address:  "  <<
subject.address << "\n";
                    std::cout  <<  "Certificate:  "  <<
subject.certificate << "\n";
            std::cout << "Activities: ";
                            for  (size_t  i  =  0;  i  <
subject.activities.size() - 1; ++i) {
                    std::cout << subject.activities[i]
<< ", ";
            }
              std::cout << subject.activities.back()
<< "\n";
                                          std::cout   <<
"--------------------------\n";
            found = true;
        }
    }
    if (!found) {
        std::cout << "Subject not found.\n";
    }
}

void    Registry::removeSubject(const    std::string&
name)
{
        for  (auto  it  =  subjects.begin();  it  !=
subjects.end(); ++it) {
        if ((*it).name == name) {
            subjects.erase(it);
            std::cout << "Subject removed:\n";
             std::cout << "Region: " << (*it).region
<< "\n";
              std::cout << "Name: " << (*it).name <<
"\n";
```

```
                        std::cout << "Manager: " <<
(*it).manager << "\n";
                        std::cout << "Address: " <<
(*it).address << "\n";
                    std::cout << "Certificate: " <<
(*it).certificate << "\n";
            std::cout << "Activities: ";
                          for (size_t i = 0; i <
(*it).activities.size() - 1; ++i) {
                  std::cout << (*it).activities[i] <<
", ";
            }
              std::cout << (*it).activities.back() <<
"\n";
                                          std::cout <<
"--------------------------\n";
            return;
        }
    }
    std::cout << "Subject not found.\n";
}
```

Лістинг main.cpp

```
#include "struct_type_project_14.h"

int main()
{
     std::string filename = "registry.txt"; // ім'я
файлу для збереження реєстру
        Registry registry(filename); // створення
об'єкту класу Registry

    int choice;
    while (true) {
        std::cout << "--------------------------\n";
        std::cout << "1. Display the register\n";
        std::cout << "2. Add a record\n";
```

```
           std::cout << "3. Search by name of legal
entity\n";
        std::cout << "4. Remove entry\n";
        std::cout << "5. Finish the job\n";
        std::cout << "Select an option: ";
        std::cin >> choice;

        switch (choice) {
        case 1:
               std::cout << "Register of subjects of
evaluation activity:\n";
            registry.printRegistry();
            break;
        case 2: {
            Subject subject;

std::cin.ignore(std::numeric_limits<std::streamsize>
::max(), '\n'); // очистка буфера
            std::cout << "Region: ";
            std::getline(std::cin, subject.region);
            std::cout << "Name: ";
            std::getline(std::cin, subject.name);
            std::cout << "Head: ";
            std::getline(std::cin, subject.manager);
            std::cout << "Legal address: ";
            std::getline(std::cin, subject.address);
            std::cout << "Certificate: ";
                                std::getline(std::cin,
subject.certificate);
             std::cout << "Areas of activity (through
whom): ";
            std::string activities;
            std::getline(std::cin, activities);
            size_t pos = 0;
              while ((pos = activities.find(',')) !=
std::string::npos) {
                               std::string activity =
activities.substr(0, pos);
```

```
subject.activities.push_back(activity);
                activities.erase(0, pos + 1);
            }

subject.activities.push_back(activities);
            registry.addSubject(subject);
              std::cout << "The entry has been added
to the register.\n";
            break;
        }
        case 3: {

std::cin.ignore(std::numeric_limits<std::streamsize>
::max(), '\n');
            std::string searchName;
                std::cout << "Enter the name of the
legal entity: ";
            std::getline(std::cin, searchName);
            registry.searchByName(searchName);
            break;
        }
        case 4: {

std::cin.ignore(std::numeric_limits<std::streamsize>
::max(), '\n');
            std::string removeName;
                std::cout << "Enter the name of the
legal entity to remove: ";
            std::getline(std::cin, removeName);
            registry.removeSubject(removeName);
            break;
        }
        case 5:
            registry.saveRegistry();
                 std::cout << "The program has been
completed. The register is saved in a file " <<
filename << ".\n";
```

```
            return 0;
        default:
                    std::cout << "Wrong choice. Try
again.\n";
        }
    }

    return 0;
}
```

Додатково створюється файл registry.txt у який записується потрібна інформація.

Висновок

В процесі роботи над цим завданням ви отримали досвід розробки програмного забезпечення для електронного реєстру суб'єктів оціночної діяльності. Ви використали мову програмування C++ та створили структуру даних для представлення записів в реєстрі. Ви також вивчили основи роботи з файлами для збереження та завантаження даних.

Це завдання дозволило вам отримати практичний досвід роботи зі структурами даних, класами та функціями. Ви навчилися взаємодіяти з користувачем, обробляти та валідувати введені дані. Також ви ознайомилися з основними операціями, такими як додавання, пошук та вилучення записів з реєстру.

Цей досвід допоможе вам у подальшій роботі з розробкою програмного забезпечення. Ви набули навичок розробки програм з використанням класів, функцій та роботи з файлами, що є важливими аспектами реальних проектів. Крім того, ви розширили свої знання з мови програмування C++.

Загалом, це було корисне завдання для вас, яке дало вам практичний досвід у розробці програмного забезпечення та розширило ваші навички програмування.