Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

**Лабораторна робота №1**

**Тема:** «Основні поняття ООП. Класи та об’єкти. Функції доступу. Вказівник this»

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Кобзар М. В. |
| Перевірив: Козірова Наталія Леонідівна |
|  |

Кропивницький

2024

**Тема:** Основні поняття ООП. Класи та об’єкти. Функції доступу. Вказівник this

**Мета:**ознайомитись з основними поняттями ООП. Вивчити поняття

клас, об’єкт, сеттер, геттер та навчитись їх програмно реалізовувати мовою С++.

**Завдання:** Завдання 1

1. Створіть клас "TravelDestination" (туристичний напрямок), який має

наступні властивості та функціональність:

Приватні поля класу:

• name (назва туристичного напрямку)

• country (країна, в якій знаходиться напрямок)

• rating (рейтинг популярності напрямку)

Публічні методи класу:

• Метод setName(), який дозволяє задати назву туристичного напрямку.

• Метод getName(), який повертає назву туристичного напрямку.

• Метод setCountry(), який дозволяє задати країну, в якій знаходиться

напрямок.

• Метод getCountry(), який повертає країну, в якій знаходиться напрямок.

• Метод setRating(), який дозволяє задати рейтинг популярності

напрямку.

• Метод getRating(), який повертає рейтинг популярності напрямку.

2. Створіть об'єкт класу "TravelDestination".

3. Задайте значення полів об'єкту за допомогою відповідних методів.

4. Виведіть інформацію про туристичний напрямок на екран,

використовуючи методи getName(), getCountry() та getRating().

5. Змініть значення рейтингу популярності напрямку за допомогою

методу setRating().

6. Виведіть оновлену інформацію про туристичний напрямок на екран,

використовуючи знову методи getName(), getCountry() та getRating().

7. Реалізувати програму за допомогою роздільної компіляції.

У вашому рішенні можуть бути додаткові методи та поля, якщо ви

вважаєте їх необхідними.

Завдання 2

Реалізувати вище наведену задачу за допомогою структурного

програмування. У висновку описати різницю цих методів.

**Варіант 8**

**Лістинг :**

1 Завдання

main.cpp

#include <iostream>

#include "TravelDestination.h"

using namespace std;

int main() {

TravelDestination destination("Kyiv", "Ukraine", 9);

cout << "Destination: " << destination.getName() << endl;

cout << "Country: " << destination.getCountry() << endl;

cout << "Rating: " << destination.getRating() << endl;

// Оновимо рейтинг

destination.setRating(10);

cout << "\nUpdated Rating: " << destination.getRating() << endl;

return 0;

}

TravelDestination.cpp

#include "TravelDestination.h"

using namespace std;

// Ініціалізація конструктора

TravelDestination::TravelDestination(const string& destinationName, const string& destinationCountry, int destinationRating)

: name(destinationName), country(destinationCountry), rating(destinationRating) {}

void TravelDestination::setName(const string& destinationName) {

name = destinationName;

}

string TravelDestination::getName() const {

return name;

}

void TravelDestination::setCountry(const string& destinationCountry) {

country = destinationCountry;

}

string TravelDestination::getCountry() const {

return country;

}

void TravelDestination::setRating(int destinationRating) {

rating = destinationRating;

}

int TravelDestination::getRating() const {

return rating;

}

TravelDestination.h

#pragma once

#include <iostream>

namespace std {

class TravelDestination {

private:

string name;

string country;

int rating;

public:

// Конструктор

TravelDestination(const string& destinationName = "", const string& destinationCountry = "", int destinationRating = 0);

void setName(const string& destinationName);

string getName() const;

void setCountry(const string& destinationCountry);

string getCountry() const;

void setRating(int destinationRating);

int getRating() const;

};

}

2 Завдання

main.cpp

#include <iostream>

#include "TravelDestination2.h"

using namespace std;

int main() {

TravelDestination destination;

// Ініціалізація значень

setName(destination, "Kyiv");

setCountry(destination, "Ukraine");

setRating(destination, 9);

cout << "Destination: " << getName(destination) << endl;

cout << "Country: " << getCountry(destination) << endl;

cout << "Rating: " << getRating(destination) << endl;

// Оновлення рейтингу

setRating(destination, 10);

cout << "\nUpdated Rating: " << getRating(destination) << endl;

return 0;

}

TravelDestination.cpp

#include "TravelDestination2.h"

void setName(TravelDestination& destination, const string& destinationName) {

destination.name = destinationName;

}

string getName(const TravelDestination& destination) {

return destination.name;

}

void setCountry(TravelDestination& destination, const string& destinationCountry) {

destination.country = destinationCountry;

}

string getCountry(const TravelDestination& destination) {

return destination.country;

}

void setRating(TravelDestination& destination, int destinationRating) {

destination.rating = destinationRating;

}

int getRating(const TravelDestination& destination) {

return destination.rating;

}

TravelDestination.h

#pragma once

#include <string>

using namespace std;

struct TravelDestination {

string name;

string country;

int rating;

};

// Функції для роботи з TravelDestination

void setName(TravelDestination& destination, const string& destinationName);

string getName(const TravelDestination& destination);

void setCountry(TravelDestination& destination, const string& destinationCountry);

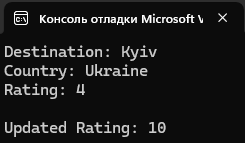
string getCountry(const TravelDestination& destination);

void setRating(TravelDestination& destination, int destinationRating);

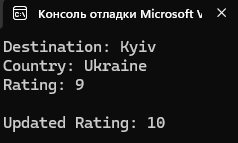
int getRating(const TravelDestination& destination);

**Результат:**

1 завдання



2 завдання



**Висновок :**

Об'єктно-орієнтоване програмування (OOP)

Код:

- Файли: `TravelDestination.h`, `TravelDestination.cpp`, `main.cpp`

- Опис: Використовується клас `TravelDestination` для інкапсуляції даних (назва, країна, рейтинг) і функцій, які працюють з цими даними (геттери та сеттери). Конструктор класу ініціалізує значення при створенні об'єкта.

Особливості:

- Клас: Об'єднує дані і методи в одному блоці, що дозволяє краще управляти зв'язками між ними.

- Капсулювання: Дані захищені від прямих змін через використання `private` модифікатора доступу, що знижує ризик помилок.

- Гнучкість: Легше додавати нові функції або змінювати існуючі, не порушуючи цілісність програми.

Переваги

- Зручність у розширенні і модифікації коду.

- Краще управління даними і їх доступом.

Структурне програмування

Код:

-Файли: `TravelDestination.h`, `TravelDestination.cpp`, `main.cpp`

- Опис: Використовується структура `TravelDestination` для зберігання даних. Окремі функції реалізують доступ до цих даних та їх модифікацію.

Особливості:

- Структура\*: Зберігає дані, але функції для роботи з даними окремі, що може ускладнити управління кодом.

- Функції: Не забезпечують захисту даних від зовнішніх змін, так як дані доступні безпосередньо.

Переваги:

- Простота реалізації для невеликих програм.

- Менш складна структура, яка може бути зручнішою для деяких простих завдань.

Різниця між OOP і Структурним Програмуванням

1. Організація даних:

- OOP: Дані і методи організовані в класах, що забезпечує кращу інкапсуляцію і захист даних.

- Структурне: Дані і функції розділені, що може ускладнити управління даними.

2. Капсулювання

- OOP: Дані можуть бути захищені від змін через модифікатори доступу.

- Структурне: Немає вбудованого захисту даних; функції мають доступ до даних.

3. Розширюваність:

- OOP: Легше додавати нові функції і класи.

- Структурне: Зміни можуть вимагати переписування багатьох частин коду.

4. Управління кодом:

- OOP: Краща організація і легше управляти великими програмами.

- Структурне: Може бути менш організованим, особливо в більших програмах.

Вибір підходу

Вибір між OOP і структурним програмуванням залежить від конкретних вимог проекту, його складності і необхідності у розширюваності. OOP забезпечує більш високий рівень абстракції і організації, що робить його придатним для великих і складних систем, тоді як структурне програмування може бути простішим для менших проектів.