Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

**Лабораторна робота №1**

**Тема: «ОСНОВНІ ПОНЯТТЯ ООП. КЛАСИ ТА ОБ’ЄКТИ. ФУНКЦІЇ ДОСТУПУ. ВКАЗІВНИК THIS.»**

|  |
| --- |
| Виконав: ст. гр. КI-23 |
| Карпець О. Р. |
| Перевірив викладач:  Козірова Н. Л. |
|  |

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++.

**Варіант 7**

**Завдання:**

**Завдання 1**

1. Створіть клас "Movie" (фільм), який має наступні властивості та функціональність: Приватні поля класу:

• title (назва фільму)

• director (режисер фільму)

• year (рік випуску фільму)

• duration (тривалість фільму в хвилинах)

* 1. Публічні методи класу:

• Метод setTitle(), який дозволяє задати назву фільму.

• Метод getTitle(), який повертає назву фільму.

• Метод setDirector(), який дозволяє задати режисера фільму.

• Метод getDirector(), який повертає режисера фільму.

• Метод setYear(), який дозволяє задати рік випуску фільму.

• Метод getYear(), який повертає рік випуску фільму.

• Метод setDuration(), який дозволяє задати тривалість фільму.

• Метод getDuration(), який повертає тривалість фільму.

2. Створіть об'єкт класу "Movie".

3. Використайте метод setTitle() для задання назви фільму.

4. Використайте метод getTitle() для отримання назви фільму та виведіть її на екран.

5. Використайте метод setDirector() для задання режисера фільму.

6. Використайте метод getDirector() для отримання режисера фільму та виведіть його на екран.

7. Використайте метод setYear() для задання року випуску фільму.

8. Використайте метод getYear() для отримання року випуску фільму та виведіть його на екран.

9. Використайте метод setDuration() для задання тривалості фільму.

10. Використайте метод getDuration() для отримання тривалості фільму та виведіть її на екран.

11. 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.

**Лістинг movie.h:**

#ifndef MOVIE\_H

#define MOVIE\_H

#include <string>

#include <iostream>

using namespace std;

class Movie

{

public:

Movie();

bool setTitle(string newTitle);

string getTitle() { return title; }

bool setDirector(string newDirector);

string getDirector() { return director; }

bool setYear(int newYear);

int getYear() { return year; }

bool setDuration(int newDuration);

int getDuration() { return duration; }

private:

string title;

string director;

int year;

int duration;

bool checkStrings(string checkString);

bool checkInts(int checkInt);

};

#endif // MOVIE\_H

**Лістинг structmovie.h:**

#ifndef STRUCTMOVIE\_H

#define STRUCTMOVIE\_H

#include <string>

using namespace std;

struct MovieStruct {

string title;

string director;

int year;

int duration;

};

bool setTitle(MovieStruct& movie, const string& newTitle);

string getTitle(const MovieStruct& movie);

bool setDirector(MovieStruct& movie, const string& newDirector);

string getDirector(const MovieStruct& movie);

bool setYear(MovieStruct& movie, int newYear);

int getYear(const MovieStruct& movie);

bool setDuration(MovieStruct& movie, int newDuration);

int getDuration(const MovieStruct& movie);

bool checkStrings(const string& checkString);

bool checkInts(int checkInt);

#endif // STRUCTMOVIE\_H

**Лістинг structmovie.cpp:**

#include "structmovie.h"

bool setTitle(MovieStruct& movie, const string& newTitle) {

if (checkStrings(newTitle)) {

movie.title = newTitle;

return true;

}

return false;

}

string getTitle(const MovieStruct& movie) {

return movie.title;

}

bool setDirector(MovieStruct& movie, const string& newDirector) {

if (checkStrings(newDirector)) {

movie.director = newDirector;

return true;

}

return false;

}

string getDirector(const MovieStruct& movie) {

return movie.director;

}

bool setYear(MovieStruct& movie, int newYear) {

if (checkInts(newYear)) {

movie.year = newYear;

return true;

}

return false;

}

int getYear(const MovieStruct& movie) {

return movie.year;

}

bool setDuration(MovieStruct& movie, int newDuration) {

if (checkInts(newDuration)) {

movie.duration = newDuration;

return true;

}

return false;

}

int getDuration(const MovieStruct& movie) {

return movie.duration;

}

bool checkStrings(const string& checkString) {

return !checkString.empty();

}

bool checkInts(int checkInt) {

return checkInt > 0;

}

**Лістинг movie.cpp:**

#include "movie.h"

Movie::Movie() {}

bool Movie::setTitle(string newTitle)

{

if (checkStrings(newTitle))

{

this->title = newTitle;

return true;

}

return false;

}

bool Movie::setDirector(string newDirector)

{

if (checkStrings(newDirector))

{

this->director = newDirector;

return true;

}

return false;

}

bool Movie::setYear(int newYear)

{

if (checkInts(newYear))

{

this->year = newYear;

return true;

}

return false;

}

bool Movie::setDuration(int newDuration)

{

if (checkInts(newDuration))

{

this->duration = newDuration;

return true;

}

return false;

}

bool Movie::checkStrings(string checkString)

{

if (checkString.length() > 0)

{

return true;

}

return false;

}

bool Movie::checkInts(int checkInt)

{

if (checkInt > 0)

{

return true;

}

return false;

}

**Лістинг main.cpp:**

#include <QCoreApplication>

#include "movie.h"

#include "structmovie.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

char order = '\0';

int userInputInt;

string userInputStr;

Movie \*movie = new Movie;

MovieStruct myMovie;

cout << "OOP/Struct (O/S)?" << endl;

cin >> order;

cin.ignore();

switch (order) {

case 'o':

case 'O':

cout << "Enter Title: " << endl;

getline(cin, userInputStr);

if (movie->setTitle(userInputStr))

{

cout << "Title " << movie->getTitle() << endl;

}

else

{

cout << "Invalid Title" << endl;

}

//////////////////

cout << "Enter Director: " << endl;

getline(cin, userInputStr);

if (movie->setDirector(userInputStr))

{

cout << "Director " << movie->getDirector() << endl;

}

else

{

cout << "Invalid Director" << endl;

}

//////////////////

cout << "Enter Duration: " << endl;

cin >> userInputInt;

if (movie->setDuration(userInputInt))

{

cout << "Duration in seconds: " << movie->getDuration() << endl;

}

else

{

cout << "Invalid Duration" << endl;

}

//////////////////

cout << "Enter Year: " << endl;

cin >> userInputInt;

if (movie->setYear(userInputInt))

{

cout << "Year: " << movie->getYear() << endl;

}

else

{

cout << "Invalid Year" << endl;

}

return 0;

case 's':

case 'S':

cout << "Enter Title: " << endl;

getline(cin, userInputStr);

if (setTitle(myMovie, userInputStr)) {

cout << "Title: " << getTitle(myMovie) << endl;

} else {

cout << "Invalid Title" << endl;

}

cout << "Enter Director: " << endl;

getline(cin, userInputStr);

if (setDirector(myMovie, userInputStr)) {

cout << "Director: " << getDirector(myMovie) << endl;

} else {

cout << "Invalid Director" << endl;

}

cout << "Enter Duration: " << endl;

cin >> userInputInt;

if (setDuration(myMovie, userInputInt)) {

cout << "Duration in seconds: " << getDuration(myMovie) << endl;

} else {

cout << "Invalid Duration" << endl;

}

cout << "Enter Year: " << endl;

cin >> userInputInt;

if (setYear(myMovie, userInputInt)) {

cout << "Year: " << getYear(myMovie) << endl;

} else {

cout << "Invalid Year" << endl;

}

return 0;

default:

cout << "Invalid order" << endl;

delete movie;

return 0;

}

delete movie;

return a.exec();

}

**Висновок:**

У використанні ООП дані, захищені від прямого доступу і можуть бути змінені лише через спеціальні методи. У структурному підході дані доступні напряму, що може призвести до помилок.

Використання класу дозволяє легко додавати нові функціональні можливості (наприклад, додавання нових методів або зміна існуючих), тоді як у структурному підході це зробити складніше.

Також код, що був написаний з використанням класів, часто легше читати і підтримувати, особливо у великих проектах. Структурний підхід може бути простішим для невеликих програм, але складнішим у масштабуванні.

Вибір між цими підходами залежить від задачі: для простих програм структурний підхід може бути достатнім, але для великих і складних проектів ООП забезпечує кращу організацію, повторне використання і підтримку коду.