# **Звіт про виконання лабораторної роботи № 1.1**

## «Поля та методи – дії над одним (поточним) об’єктом» з дисципліни «Об’єктно-орієнтоване програмування»

## Студента групи «**ІТ-12**» - **Степанчука Сергія**

## Мета роботи

Освоїти використання класів. Навчитися створювати багатомодульні С++-проекти

## Умова завдання

Класом-парою називається клас з двома приватними полями, які мають імена first та

second. Потрібно реалізувати такий клас. Обов’язково мають бути присутніми:

 методи доступу (константні методи зчитування та методи запису) значення кожного

поля;

 метод ініціалізації Init( ); метод має контролювати значення аргументів на

коректність;

 метод введення з клавіатури Read( );

 метод виведення на екран Display( ).

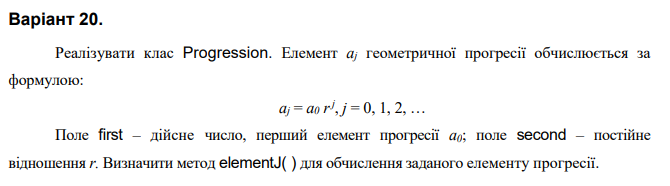
Реалізувати зовнішню функцію з ім’ям makeКлас( ), де Клас – ім’я класу, об’єкт

якого вона створює. Функція має отримувати як аргументи значення для полів класу і

повертати об’єкт необхідного класу. При передачі помилкових параметрів слід виводити

повідомлення і закінчувати роботу.

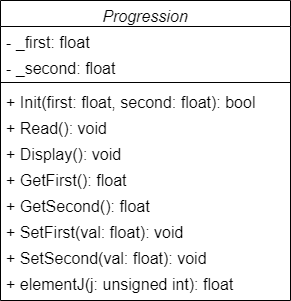
Визначення класу та реалізацію його методів слід розмістити в окремих модулях.

Варіанти завдань наступні:

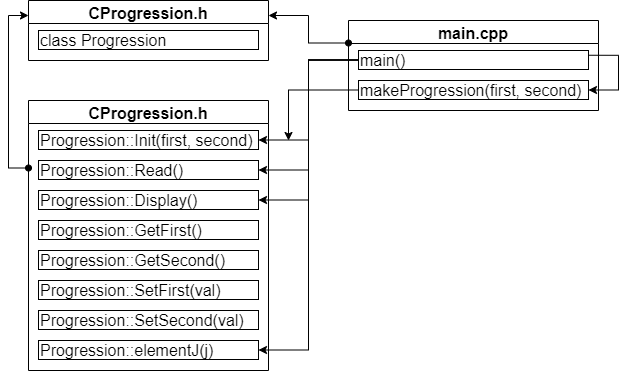
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_1.1>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // CProgression.h

#pragma once

#include <iostream>

#include <cmath>

class Progression

{

float \_first = 0;

float \_second = 0;

public:

bool Init(const float &first, const float &second);

void Read();

void Display() const;

float GetFirst() const;

float GetSecond() const;

void SetFirst(const float& val);

void SetSecond(const float& val);

float elementJ(const unsigned int& j) const;

};

### // CProgression.cpp

#include "CProgression.h"

using namespace std;

bool Progression::Init(const float &first, const float &second)

{

\_first = first;

if (second <= 0)

return false;

return \_second = second;

}

void Progression::Read()

{

float tfirst, tsecond;

do

{

cout << "Input progression values:" << endl;

cout << " b(0) = "; cin >> tfirst;

cout << " q = "; cin >> tsecond;

} while (!Init(tfirst, tsecond));

}

void Progression::Display() const

{

cout << " b(0) = " << \_first << endl

<< " q = "<< \_second << endl;

}

float Progression::GetFirst() const

{

return \_first;

}

float Progression::GetSecond() const

{

return \_second;

}

void Progression::SetFirst(const float& val)

{

\_first = val;

}

void Progression::SetSecond(const float& val)

{

\_second = val;

}

float Progression::elementJ(const unsigned int& j) const

{

return \_first \* pow(\_second, j);

}

### // main.cpp

#include <iostream>

#include "CProgression.h"

using namespace std;

Progression makeProgression(const float &first, const double &second)

{

Progression obj;

if (!obj.Init(first, second))

cout << "Wrong arguments to Init(second)!" << endl;

return obj;

}

void main()

{

unsigned int j;

Progression pgr;

pgr.Read();

pgr.Display();

cout << "Input j = "; cin >> j;

cout << "elementJ(j) = " << pgr.elementJ(j) << endl << endl;

double tfirst, tsecond;

cout << "Input progression values:" << endl;

cout << " b(0) = "; cin >> tfirst;

cout << " q = "; cin >> tsecond;

pgr = makeProgression(tfirst, tsecond);

pgr.Display();

cout << "Input j = "; cin >> j;

cout << "elementJ(j) = " << pgr.elementJ(j) << endl << endl;

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Program/CProgression.h"

#include "../Program/CProgression.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(TestProgression)

{

public:

TEST\_METHOD(Init)

{

Progression pgr;

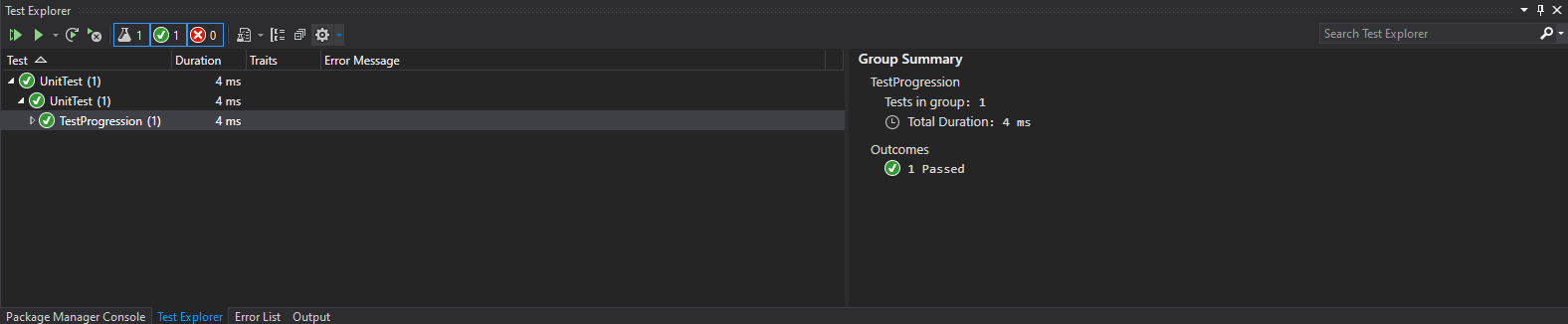
Assert::AreEqual(pgr.Init(1, 0), false);

}

};

}

### Результат



## Висновок

Навчився використовувати класи.