

Тема

Основи модульного тестування програм з використанням мови C++

Мета

Навчитись створювати та виконувати модульні тести з використанням бібліотеки Google Test

Теоретичні відомості

Google Test – це бібліотека для модульного тестування коду мовою C++. Вона поширюється за ліцензією BSD та може компілюється під багато POSIX та Windows платформ та дозволяє тестувати код C та C++ з мінімальними змінами.

Розробка тестів полягає у створенні функцій-тестів, що містять виклик тестованого коду та спеціальних макросів стверджень (EXPECT_*, ASSERT_*, і тд.). Для створення функцій використовують спеціальні макроси (TEST, TEST_F, TEST_P).

Створення простого тесту

Щоб створити простий тест, потрібно виконати такі кроки:

1. Використайте макрос TEST() для визначення та іменування функції тесту. Функції тесту не повертають значення.
2. Всередині функції поряд із іншим C++ кодом використати різноманітні ствердження Google Test для перевірки значень.
3. Результат тесту визначається ствердженнями. Якщо хоча б одне із стверджень не успішне (фатально чи не фатально), чи якщо тест призводить до аварійного завершення, тоді увесь тест неуспішний. Інакше, він успішний.

```
TEST(TestSuiteName, TestName)
{
    ... test body ...
}
```

Рис. 1. Структура тесту у Google Test

Аргументи TEST позначають зліва направо від загального до конкретного. TestSuiteName – це назва набору тестів, а TestName – назва конкретного тесту. Обидва імені мають бути коректними C++ ідентифікаторами і не можуть містити підкреслень (_). Повне ім'я тесту складається з назви набору та власне тесту, така пара має бути унікальна в межах програми.

Розглянемо приклад нижче. Функція, що повертає факторіал числа:

```
unsigned int Factorial(unsigned int n); // Returns the factorial of n
```

Рис. 2. Оголошення функції Factorial.

Набір тестів для неї може виглядати наступним чином:

```
// Tests factorial of 0.
TEST(FactorialTest, HandlesZeroInput)
{
    EXPECT_EQ(Factorial(0), 1U);
}

// Tests factorial of positive numbers.
```

```
TEST(FactorialTest, HandlesPositiveInput)
{
    EXPECT_EQ(Factorial(1U), 1U);
    EXPECT_EQ(Factorial(2U), 2U);
    EXPECT_EQ(Factorial(3U), 6U);
    EXPECT_EQ(Factorial(8U), 40320U);
}
```

Рис. 3. Можливий набір тестів для функції Factorial

Google Test групує результати тестів по назві набору, тому логічно пов'язані тести мають бути в одному наборі – тобто перший аргумент макросу TEST() має бути однаковим. У прикладі вище в нас є два тести – HandlesZeroInput та HandlesPositiveInput, які належать одному набору – FactorialTest.

Ствердження

Google Test надає спеціальні макроси, за допомогою яких можна перевіряти поведінку коду. Щоб їх використовувати, необхідно підключити файл-заголовок gtest/gtest.h.

Більшість макросів є парними – з EXPECT_ варіантом, та з ASSERT_. В разі неуспіху EXPECT_ макрос генерує нефатальну помилку і дозволяє функції тесту продовжувати виконання, натомість ASSERT_ версії генерують фатальну помилку і переривають виконання функції тесту.

Всі макроси для стверджень підтримують задання власного повідомлення про помилку за допомогою оператора <<, наприклад:

```
EXPECT_TRUE(my_condition) << "My condition is not true";
```

Рис. 4. Використання оператора << для задання власного повідомлення про помилку

Нижче наводиться декілька прикладів макросів із поясненням їх роботи. Більше інформації про макроси можна знайти на сторінці офіційної документації за адресою

<http://google.github.io/googletest/reference/assertions.html>

1. Явний успіх чи неуспіх. До цієї групи належать такі макроси:
 - a. FAIL() – генерує неуспішне
 - b. SUCCEED() – генерує успіх
2. Перевірки булевої умови
 - a. EXPECT_TRUE(condition) та ASSERT_TRUE(condition). Перевіряють чи умова condition істина
 - b. EXPECT_FALSE(condition) та ASSERT_FALSE(condition). Перевіряють чи умова condition хибна
3. Бінарне порівняння
 - a. EXPECT_EQ(val1, val2) та ASSERT_EQ(val1, val2). Перевіряють чи val1==val2.
 - b. EXPECT_NE(val1, val2) та ASSERT_NE(val1, val2). Перевіряють чи val1!=val2.
 - c. EXPECT_LT(val1, val2) та ASSERT_LT(val1, val2). Перевіряють val1<val2.
 - d. EXPECT_LE(val1, val2) та ASSERT_LE(val1, val2). Перевіряють val1<=val2.
 - e. EXPECT_GT(val1, val2) та ASSERT_GT(val1, val2). Перевіряють val1>val2.
 - f. EXPECT_GE(val1, val2) та ASSERT_GE(val1, val2). Перевіряють val1>=val2.
4. Порівняння рядків
 - a. EXPECT_STREQ(str1, str2) та ASSERT_STREQ(str1, str2). Перевіряють, що два C рядки (тип char *) мають однаковий вміст.
 - b. EXPECT_STRNE(str1, str2) та ASSERT_STRNE(str1, str2). Перевіряють, що два C рядки (тип char *) мають різний вміст.
 - c. EXPECT_STRCASEEQ(str1, str2) та ASSERT_STRCASEEQ(str1, str2). Перевіряють, що два C рядки (тип char *) мають однаковий вміст ігноруючи регістр літер.

- d. EXPECT_STRCASENE(str1,str2) та ASSERT_STRCASENE(str1,str2). Перевіряють, що два C рядки (тип char *) мають різний вміст ігноруючи регістр літер.
- 5. Порівняння чисел із плаваючою крапкою. У зв'язку із помилками заокруглення, дуже рідко щоб два числа із плаваючою крапкою співпадали точно, тому EXPECT_EQ для таких випадків не підходить. Google Test надає макроси які порівнюють такі числа з певною точністю (використовується підхід UPL – Units in the Last Place).
 - a. EXPECT_FLOAT_EQ(val1,val2) та ASSERT_FLOAT_EQ(val1,val2). Перевіряють чи два float значення val1 та val2 приблизно однакові в межах 4 ULPs.
 - b. EXPECT_DOUBLE_EQ(val1,val2) та ASSERT_DOUBLE_EQ(val1,val2). Перевіряють чи два double значення val1 та val2 приблизно однакові в межах 4 ULPs.
 - c. EXPECT_NEAR(val1,val2,abs_error) та ASSERT_NEAR(val1,val2,abs_error). Перевіряють чи різниця між val1 та val2 не перевищує абсолютну межу похибки abs_error.
- 6. Ствердження для виключень. Дані макроси перевіряють чи частина коду генерує, або не генерує виключень.
 - a. EXPECT_THROW(statement, ex_type) та ASSERT_THROW(statement, ex_type). Перевіряють чи вираз statement генерує виключення типу ex_type
 - b. EXPECT_ANY_THROW(statement) та ASSERT_ANY_THROW(statement). Перевіряють чи вираз statement генерує виключення будь-якого типу.
 - c. EXPECT_NO_THROW(statement) та ASSERT_NO_THROW(statement). Перевіряють чи вираз statement не генерує жодних виключень.

Завдання

1. Встановити Visual Studio Community Edition (див. Додаток 1). Під час встановлення переконайтесь що обрано компонент 'Test Adapter for Google Test', як показано на рисунку нижче

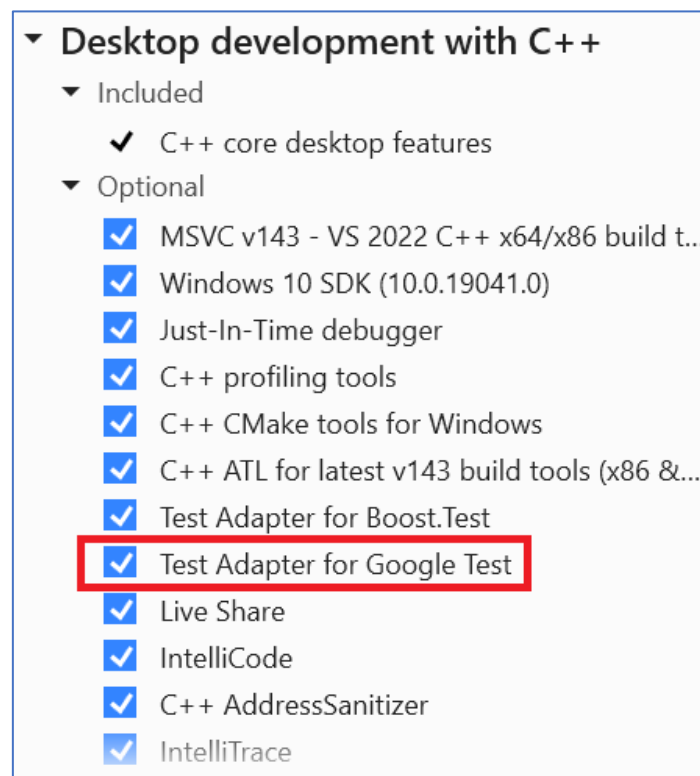


Рис. 5. Склад набору компонент для 'Desktop development with C++'

2. Створити порожній проєкт (Empty Project) у Visual Studio, обрати для нього назву - oop_lab2_prog. Обрати опцію 'Place solution and project in the same directory' щоб створити проєкт та рішення в тій же директорії.

- У вікні 'Solution Explorer' натиснути правою клавішею миші на назві проекту ('oop_lab2_prog') та обрати пункт 'Properties', щоб відкрити властивості проекту.

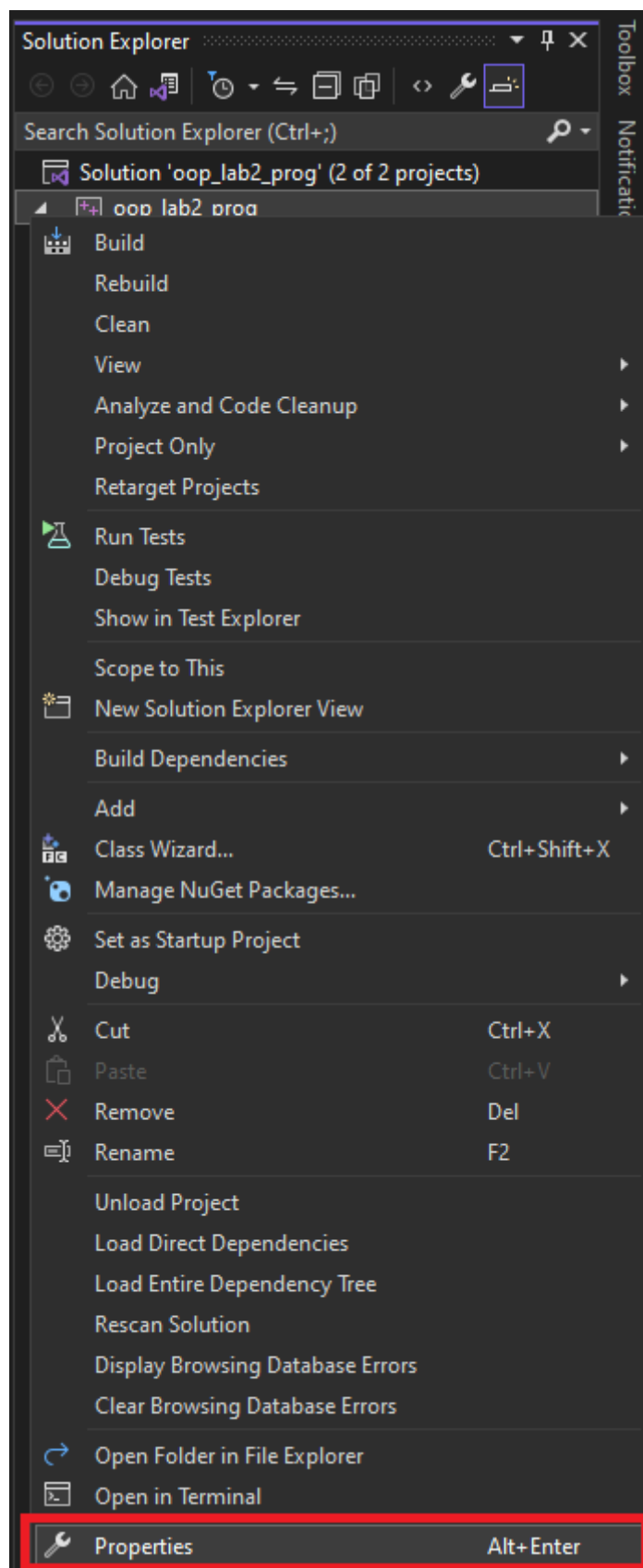


Рис. 6. Контекстне меню проекту у Solution Explorer

- Задати значення для Configuration Type – Static library.
- Задати значення для C++ Language Standard – ISO C++ 17 Standard
- Та натиснути кнопку 'OK', щоб зберегти зміни.

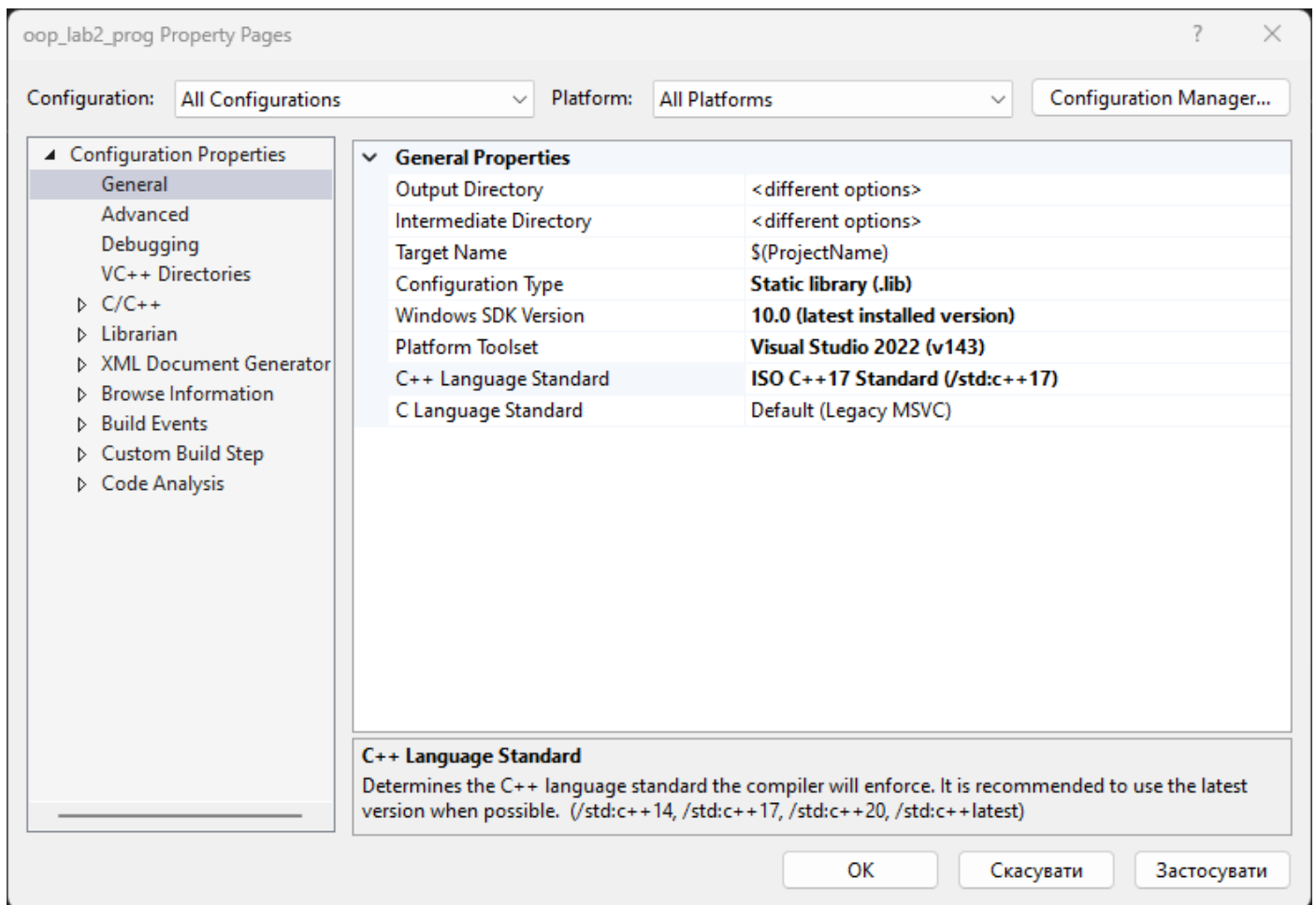


Рис. 7. Налаштування проекту 'oop_lab2_prog'

- Додати 'lab2.h'. Для цього натиснути правою кнопкою миші на фільтр 'Header Files', обрати 'Add -> New Item...'.

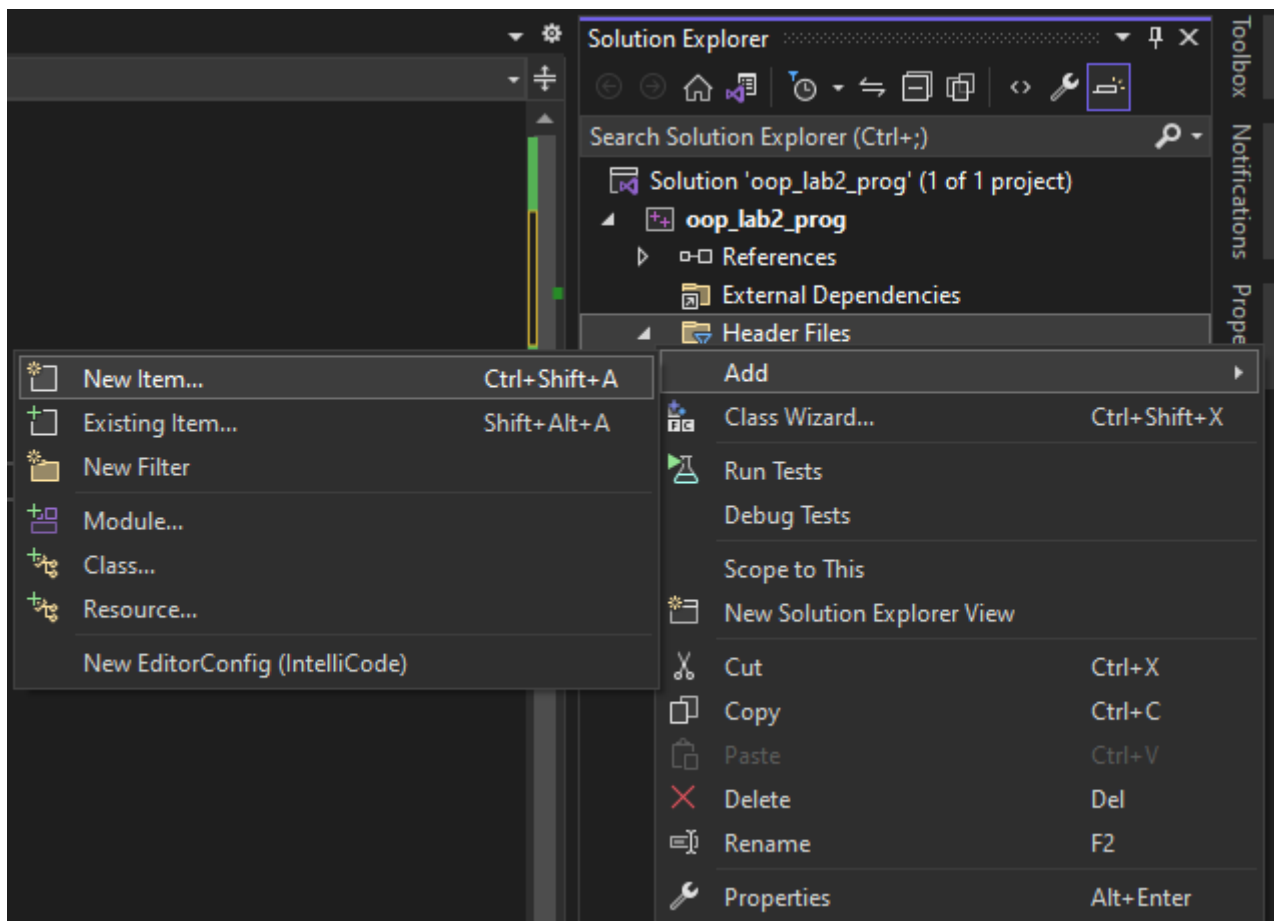


Рис. 8. Контекстне меню для додавання .h файлу

8. У діалоговому вікні обрати тип 'Header File' та ввести назву 'lab2.h'

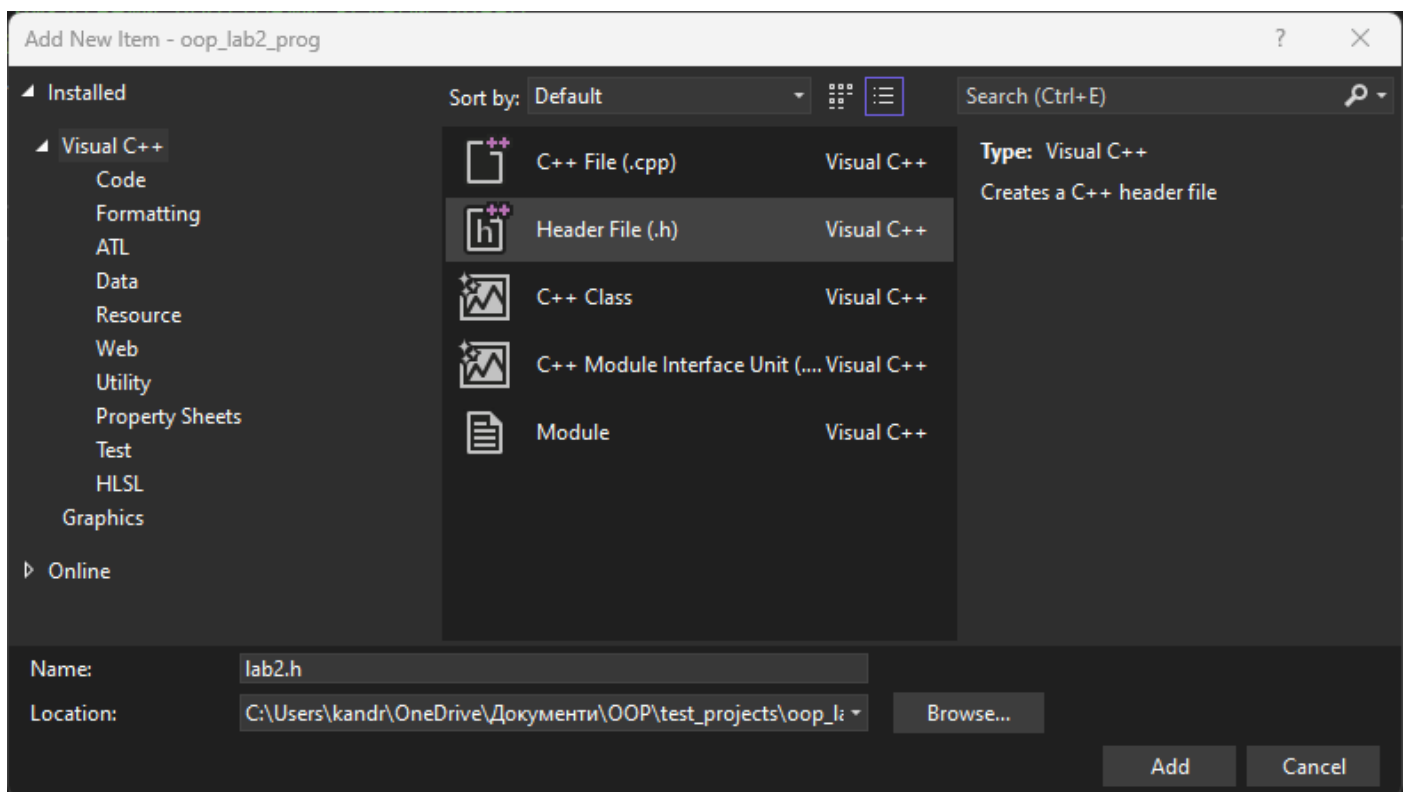


Рис. 9. Діалогове вікно додавання .h файлу

9. У файлі 'lab2.h' додати оголошення функції згідно з варіантом.

10. Додати файл 'lab2.cpp'. Для цього натиснути правою кнопкою миші на фільтр 'Source Files', обрати 'Add -> New Item...'

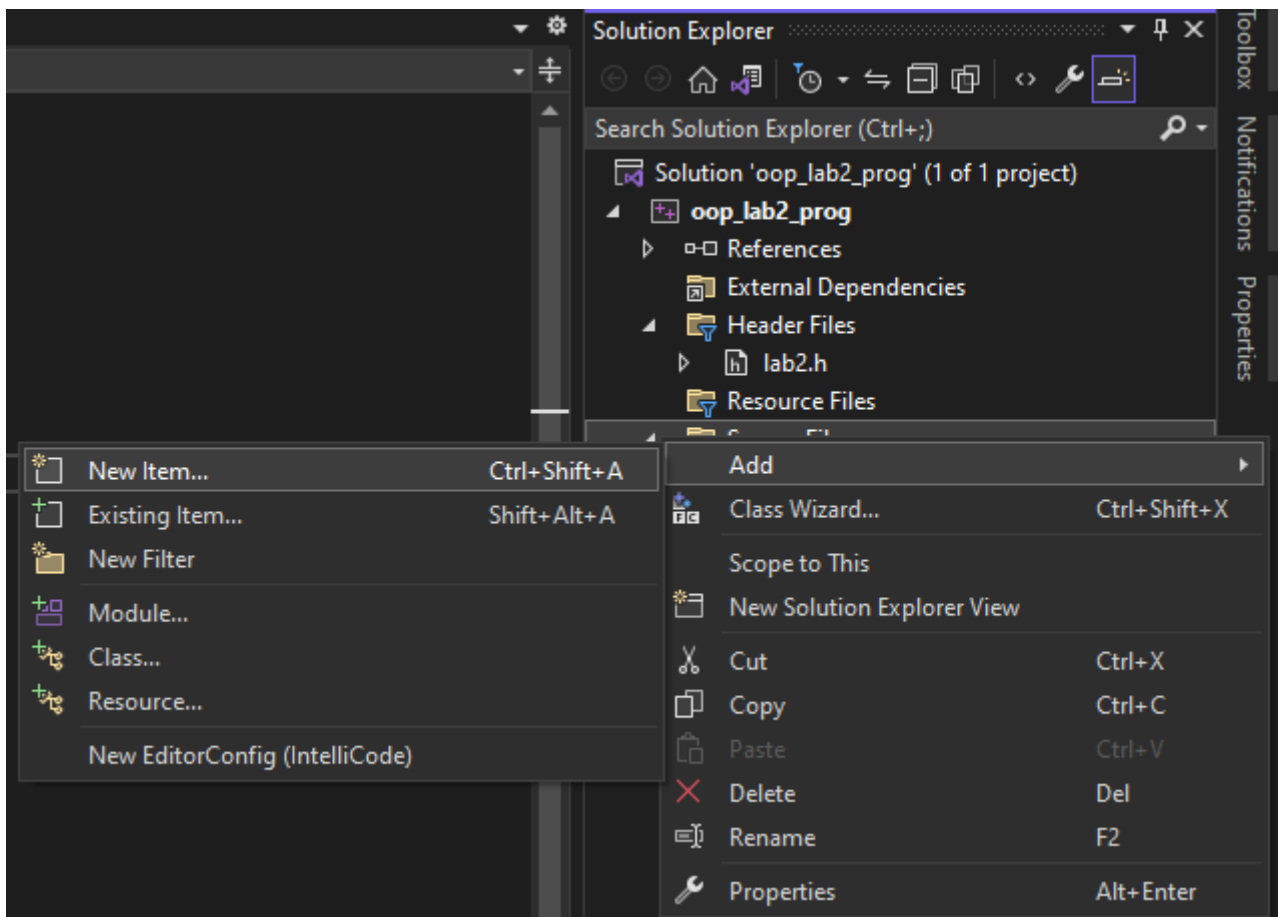


Рис. 10. Контекстне меню для додавання .cpp файлу

11. У діалоговому вікні обрати тип 'C++ File' та ввести назву 'lab2.cpp'

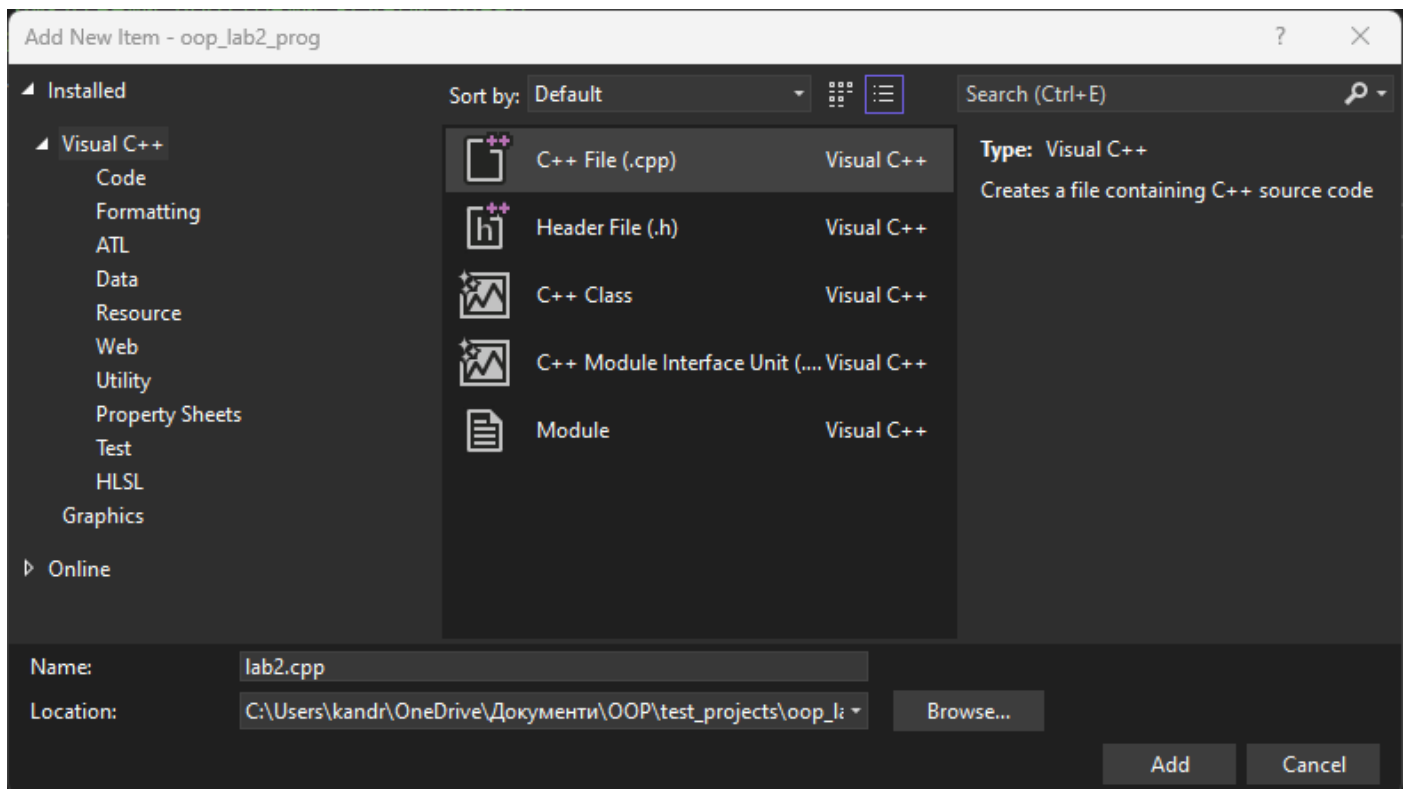


Рис. 11. Діалогове вікно додавання .cpp файлу

12. У файлі 'lab2.cpp' додати визначення функції згідно з варіантом. Визначення функції необхідно реалізувати самостійно.

13. Після виконання кроків 7-12 проект має мати такий вигляд

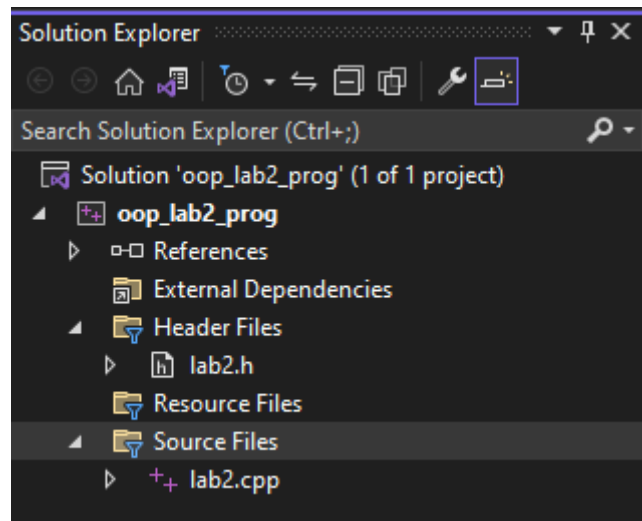


Рис. 12. Вигляд проекту після додавання lab2.h та lab2.cpp файлів

14. Додати новий проект до рішення (Solution). Для цього натиснути правою клавiшею миші на його назві та обрати пункт 'Add -> New project', щоб додати новий проект.

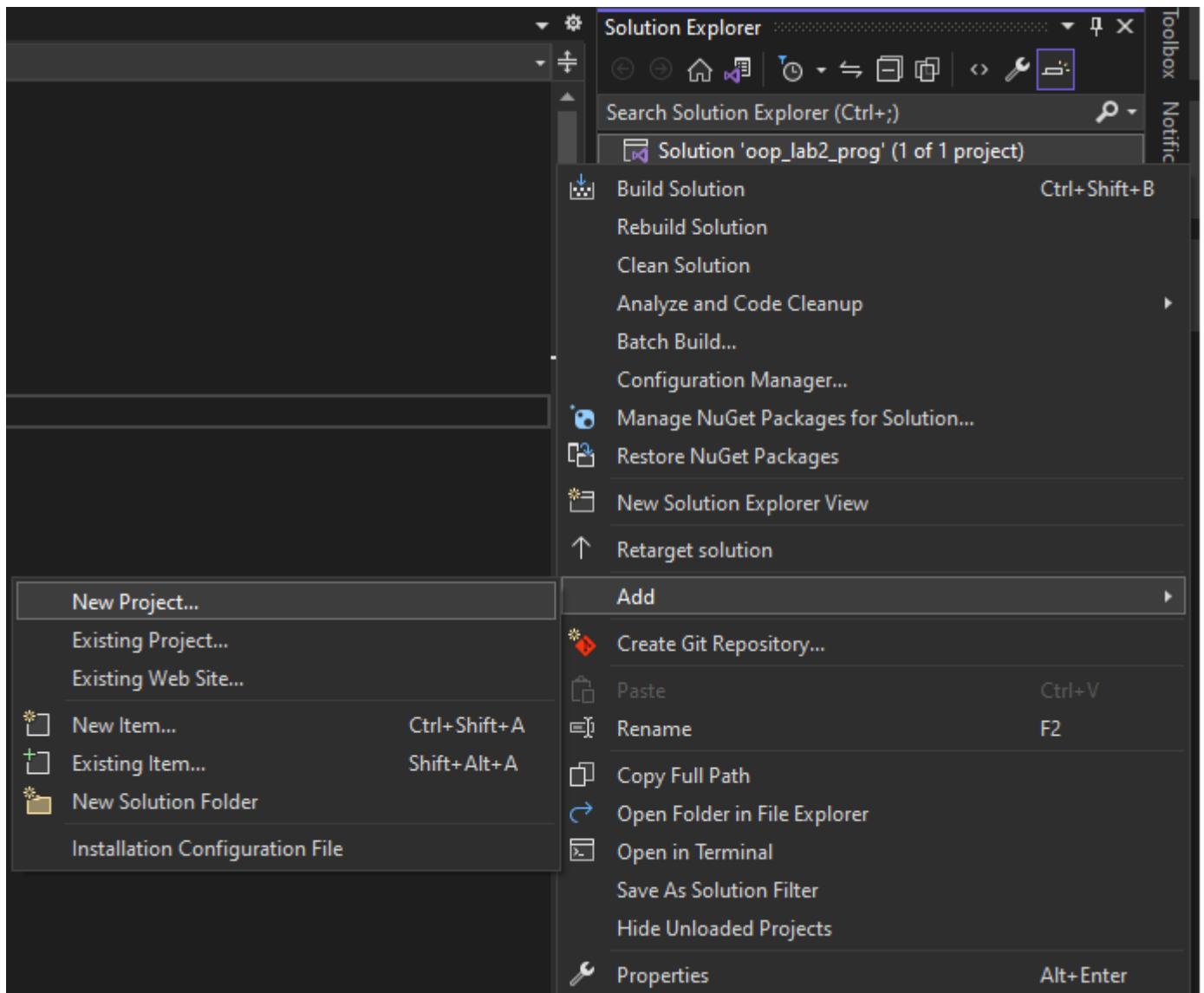


Рис. 13. Контекстне меню додавання нового проекту до рішення

15. Обрати проект 'Google Test' та натисніть 'Next'. Введіть назву 'oop_lab2_test' та натисніть 'Create'

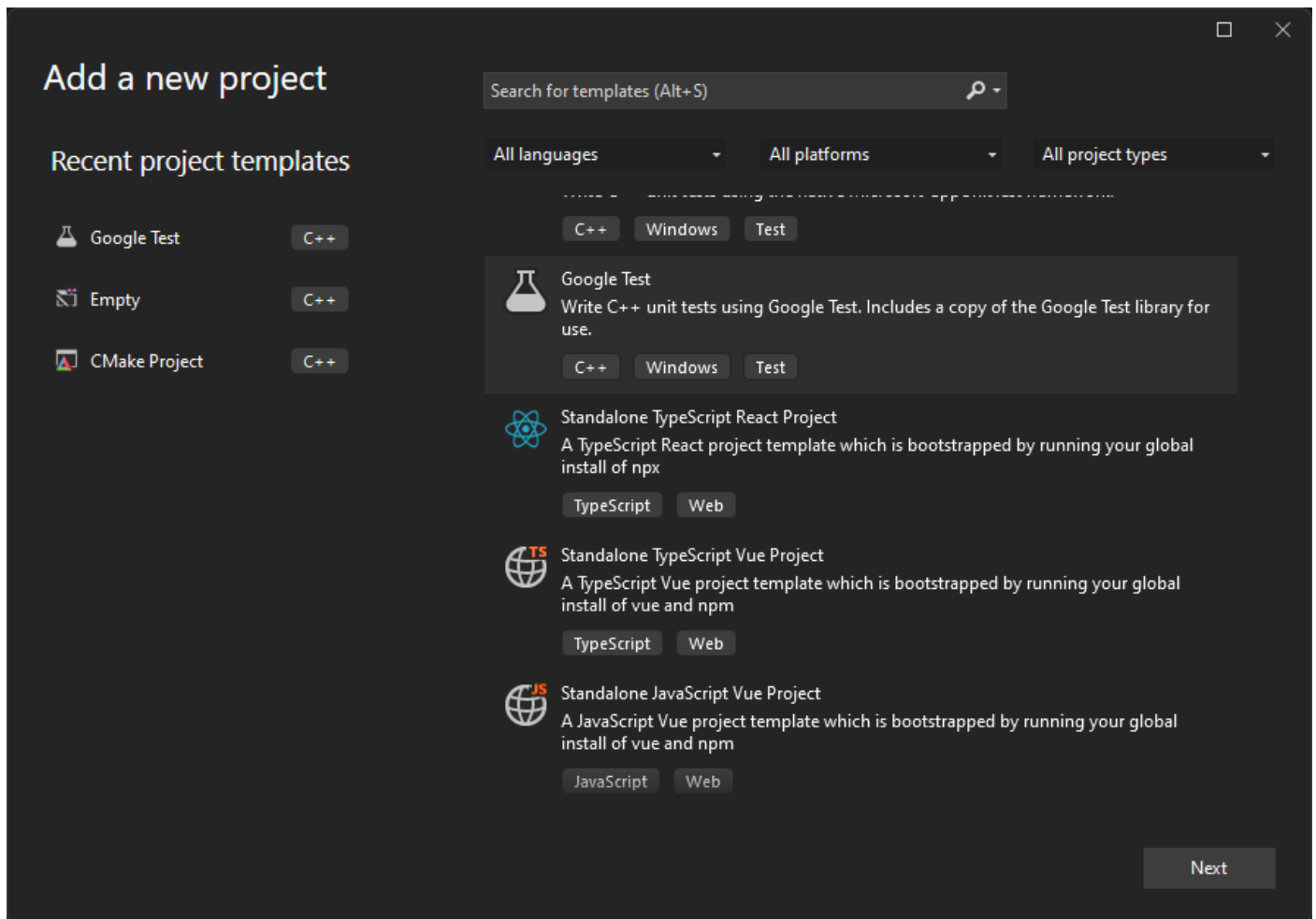


Рис. 14. Діалогове вікно вибору типу нового проекту

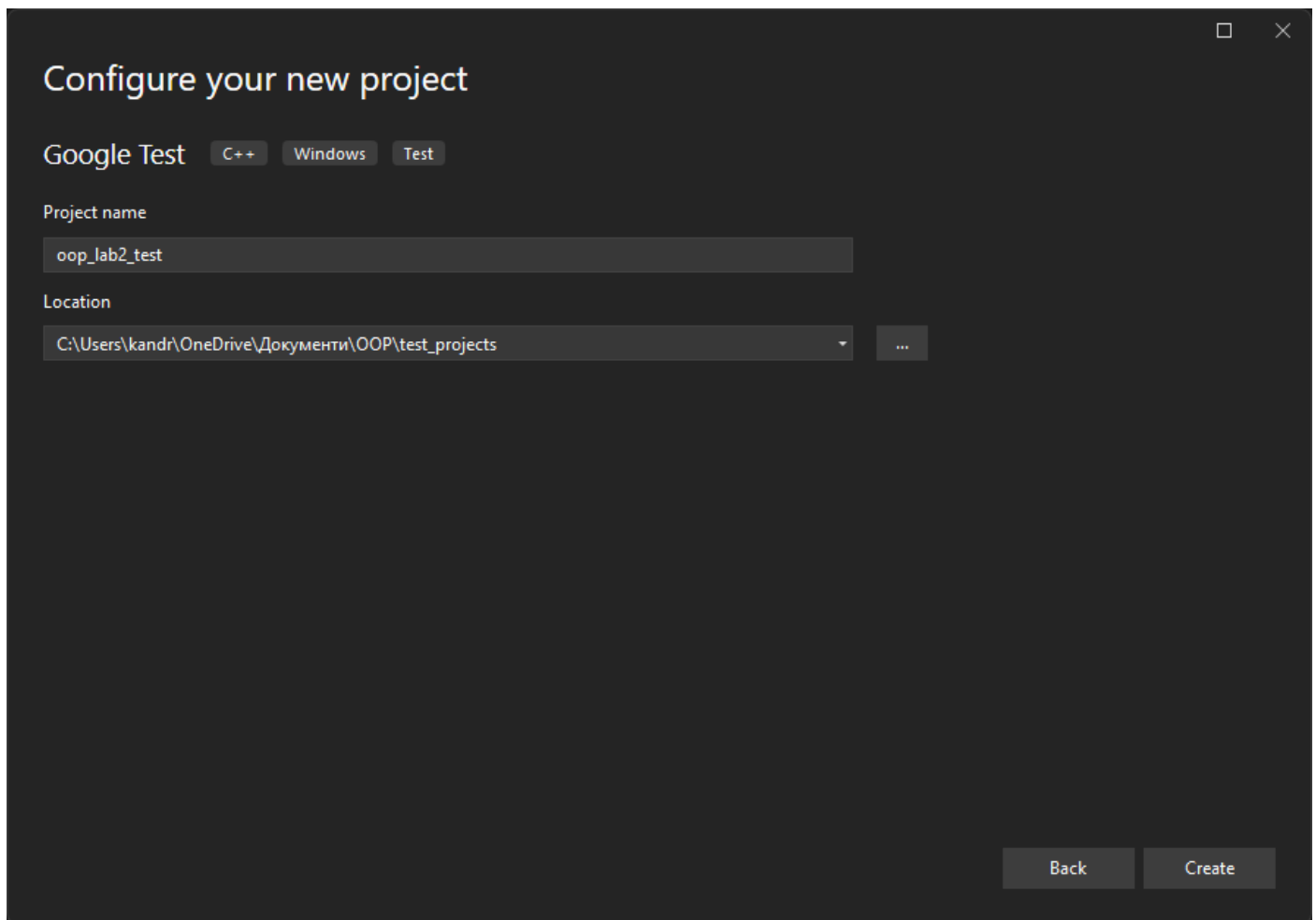


Рис. 15. Вікно налаштування імені та шляху до проекту

16. У полі 'Select project to test (Optional)' оберіть 'oop_lab2_prog'. Решта параметрів залиште зі значенням за замовчуванням, як показано нижче. Натисніть 'ОК'

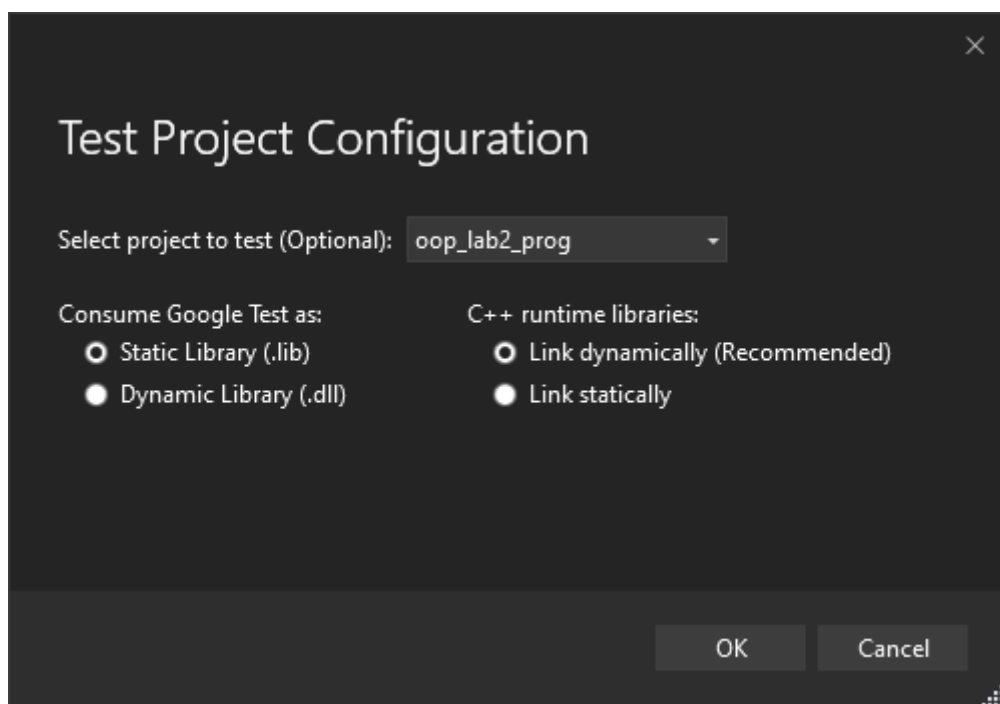


Рис. 16. Вікно налаштування проекту Google Test

17. Після цього вікно Solution Explorer має мати наступний вигляд

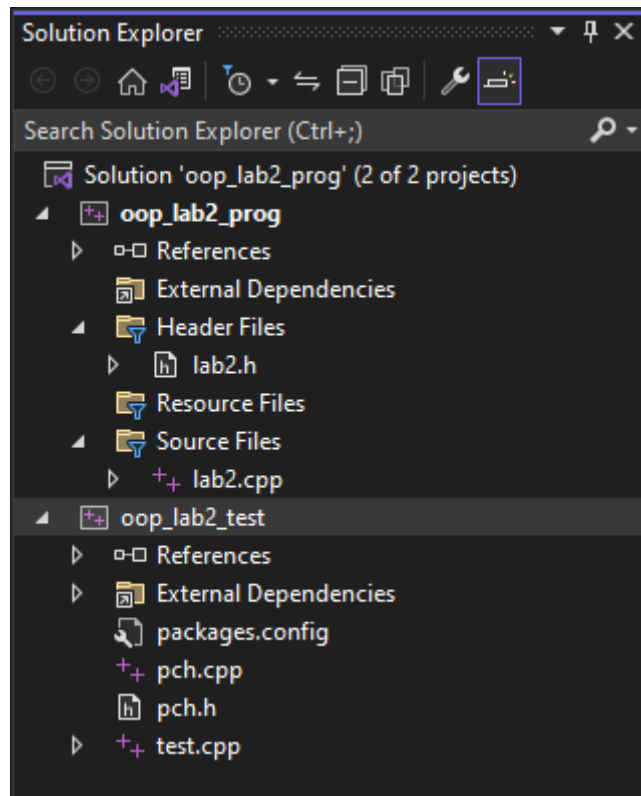


Рис. 17. Вигляд вікна Solution Explorer після додавання проекту 'oop_lab2_test'

18. Змініть стандарт мови C++ на C++ 17 для проекту oop_lab2_test виконуючи такі ж дії, як для проекту 'oop_lab2_prog' у кроках 4, 6, 7.
19. Зробіть проект 'oop_lab2_test' активним для виконання. Для цього в його контекстному меню оберіть пункт 'Set as Startup Project'

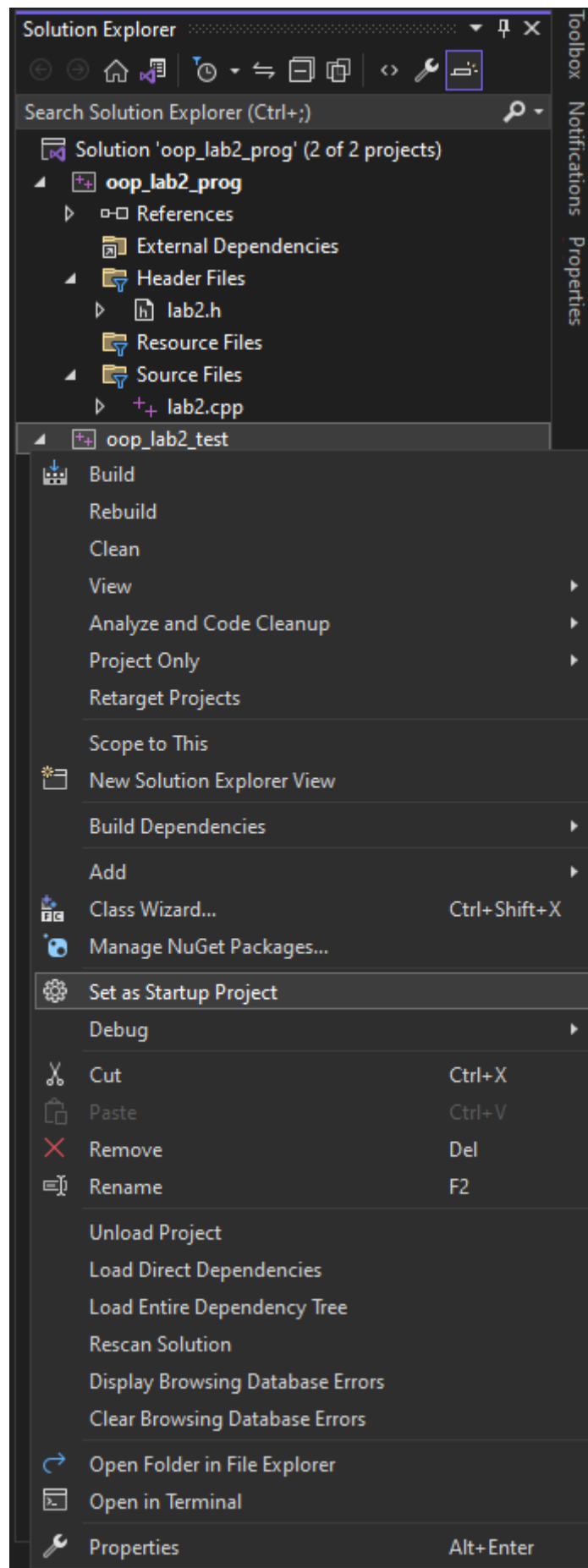
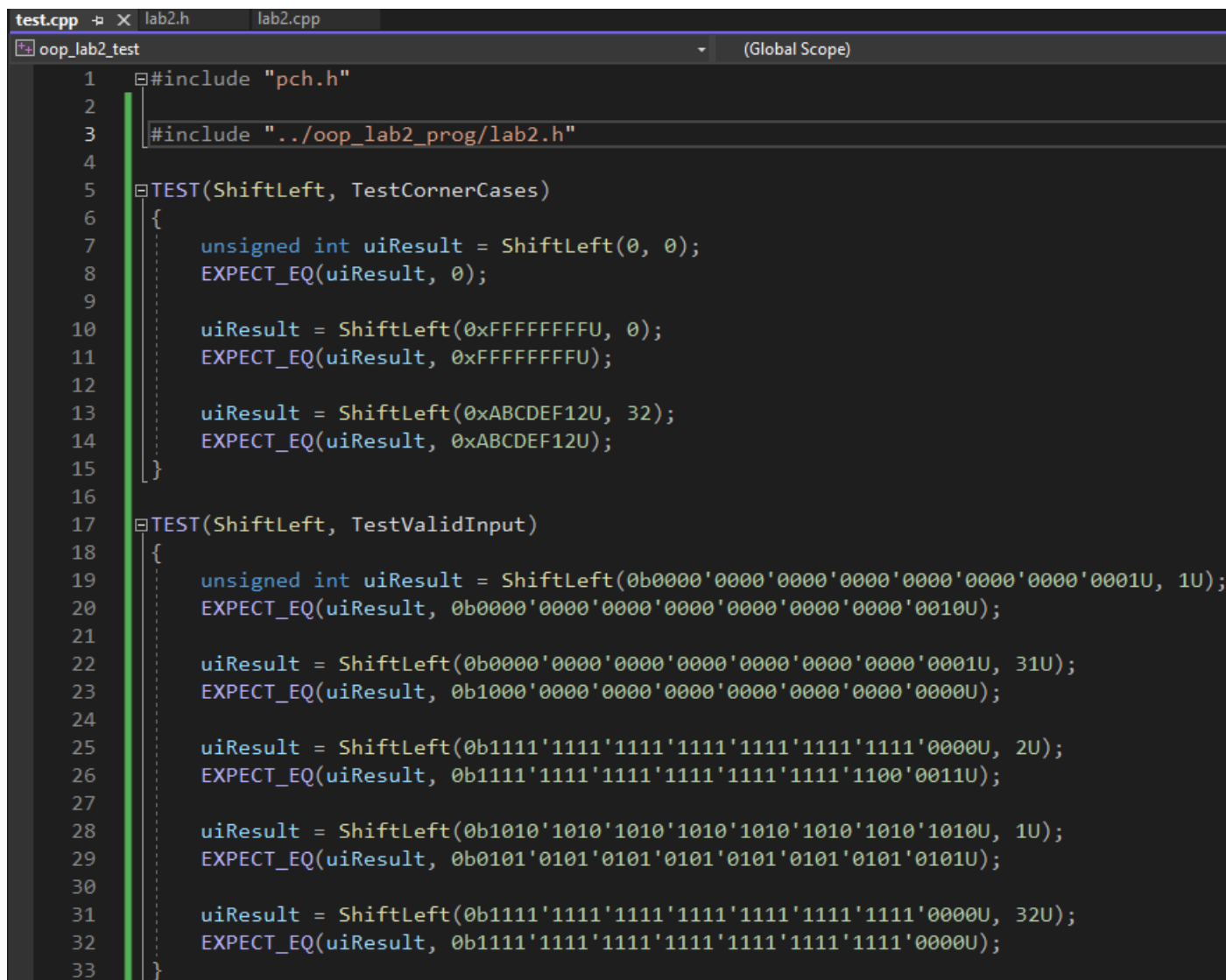


Рис. 18. Контекстне меню для налаштування проекту активним до виконання

20. У файлі 'test.cpp' додайте включення 'lab2.h' файлу та відповідну кількість тестів для функції, яка там оголошена. Тести мають перевіряти всі можливі гілки виконання та граничні значення вхідних параметрів. Приклад, як виглядає файл 'test.cpp' подано нижче.



```
test.cpp lab2.h lab2.cpp
+ oop_lab2_test (Global Scope)
1  #include "pch.h"
2
3  #include "../oop_lab2_prog/lab2.h"
4
5  TEST(ShiftLeft, TestCornerCases)
6  {
7      unsigned int uiResult = ShiftLeft(0, 0);
8      EXPECT_EQ(uiResult, 0);
9
10     uiResult = ShiftLeft(0xFFFFFFFFU, 0);
11     EXPECT_EQ(uiResult, 0xFFFFFFFFU);
12
13     uiResult = ShiftLeft(0xABCDEF12U, 32);
14     EXPECT_EQ(uiResult, 0xABCDEF12U);
15 }
16
17 TEST(ShiftLeft, TestValidInput)
18 {
19     unsigned int uiResult = ShiftLeft(0b0000'0000'0000'0000'0000'0000'0001U, 1U);
20     EXPECT_EQ(uiResult, 0b0000'0000'0000'0000'0000'0000'0010U);
21
22     uiResult = ShiftLeft(0b0000'0000'0000'0000'0000'0000'0001U, 31U);
23     EXPECT_EQ(uiResult, 0b1000'0000'0000'0000'0000'0000'0000U);
24
25     uiResult = ShiftLeft(0b1111'1111'1111'1111'1111'1111'0000U, 2U);
26     EXPECT_EQ(uiResult, 0b1111'1111'1111'1111'1111'1111'1100'0011U);
27
28     uiResult = ShiftLeft(0b1010'1010'1010'1010'1010'1010'1010U, 1U);
29     EXPECT_EQ(uiResult, 0b0101'0101'0101'0101'0101'0101'0101U);
30
31     uiResult = ShiftLeft(0b1111'1111'1111'1111'1111'1111'0000U, 32U);
32     EXPECT_EQ(uiResult, 0b1111'1111'1111'1111'1111'1111'1111'0000U);
33 }
```

Рис. 19. Файл 'test.cpp' із набором тестів

21. Виконати програму та переконатись, що всі тести успішні.

```
Microsoft Visual Studio Debug Console
Running main() from D:\a_work\1\s\ThirdParty\googletest\googletest\src\gtest_main.cc
[=====] Running 2 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 2 tests from ShiftLeft
[ RUN      ] ShiftLeft.TestCornerCases
[ OK       ] ShiftLeft.TestCornerCases (0 ms)
[ RUN      ] ShiftLeft.TestValidInput
[ OK       ] ShiftLeft.TestValidInput (0 ms)
[-----] 2 tests from ShiftLeft (0 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test case ran. (1 ms total)
[ PASSED  ] 2 tests.

C:\Users\kandr\OneDrive\Документи\OOP\test_projects\oop_lab2_prog\x64\Debug\oop_lab2_test.exe (process 20384) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Рис. 20. Результат виконання проекту 'oop_lab2_test'

22. Виконати всі тести користуючись меню 'Test -> Run all tests'

Test Explorer

Test run finished: 2 Tests (2 Passed, 0 Failed, 0 Skipped) run in 1,1 sec

Test	Duration	Traits	Error Message
oop_lab2_test (2)	< 1 ms		
<Empty Namespace> (2)	< 1 ms		
ShiftLeft (2)	< 1 ms		
TestCornerCases	< 1 ms		
TestValidInput	< 1 ms		

Test Detail Summary

- ShiftLeft.TestValidInput
 - Source: [test.cpp](#) line 18
 - Duration: < 1 ms

Рис. 21. Результат виконання тестів через меню 'Test -> Run all tests'

23. Оформити звіт про виконання лабораторної роботи. Звіт має містити:

- Тему, мету та теоретичні відомості
- Індивідуальний варіант завдання
- Реалізацію функції відповідно до варіанту завдань (текст коду lab2.cpp)
- Реалізацію тестів для функції (текст коду test.cpp)
- Результат виконання тестів (скріншот вікна Text Explorer та консольного виконання проекту oop_lab2_test)

Варіанти завдань

Варіант	Прототип функції
1	// функція повертає периметр трикутника або NAN якщо вхідні дані неправильні <code>double GetTrianglePerimeter(double sideA, double sideB, double sideC);</code>
2	// функція повертає площу трикутника або NAN якщо вхідні дані неправильні <code>double GetTriangleArea(double sideA, double sideB, double sideC);</code>
3	// функція повертає розв'язок лінійного рівняння $ax + b = 0$ або NAN якщо вхідні дані неправильні <code>double SolveLinearEquation(double coefA, double coefB);</code>
4	// функція повертає кількість розв'язків квадратного рівняння $ax^2 + bx + c = 0$ та дійсні розв'язки через параметри pX1, pX2 <code>unsigned int SolveQuadraticEquation(double coefA, double coefB, double coefC, double * pX1, double * pX2);</code>
5	// функція вертає true, якщо число є простим та false якщо ні <code>bool IsPrime(unsigned int uiN);</code>
6	// функція повертає найбільший спільний дільник для заданих двох чисел <code>unsigned int GetGCD(unsigned int uiFirst, unsigned int uiSecond);</code>
7	// функція повертає найменше спільне кратне для заданих двох чисел <code>unsigned int GetLCM(unsigned int uiFirst, unsigned int uiSecond);</code>
8	// функція повертає число Фібоначі за номером <code>unsigned int GetFibonacciNumber(unsigned int N);</code>
9	// функція повертає число, яке утворене циклічним побітовим зсувом право заданого числа на задану кількість бітів <code>unsigned int ShiftRight(unsigned int uiNumber, unsigned int uiBits);</code>
10	// функція повертає число, яке утворене циклічним побітовим зсувом вліво заданого числа на задану кількість бітів <code>unsigned int ShiftLeft(unsigned int uiNumber, unsigned int uiBits);</code>
11	// функція повертає об'єм циліндра з радіусом dRadius та висотою dHeight або NAN якщо вхідні дані неправильні <code>double GetCylinderVolume(double dRadius, double dHeight);</code>
12	// функція повертає об'єм прямокутного паралелепіпеда з шириною dSideA, довжиною dSideB та висотою dHeight або NAN якщо вхідні дані неправильні <code>double GetBoxVolume(double dSideA, double dSideB, double dHeight);</code>
13	// Функція вертає тривалість безперебійної роботи споживачів (год) з потужністю dW (Вт) від джерела аварійного живлення ємністю dC (Вт*год) з коефіцієнтом перетворення dRatio ($0 < \text{dRatio} < 1$). <code>double GetTimeOfOperationInHours(double dW, double dC, double dRatio);</code>
14	// Функція вертає необхідну ємність джерела аварійного живлення з коефіцієнтом перетворення dRatio ($0 < \text{dRatio} < 1$), яке забезпечить споживачів потужністю dW (Вт) протягом часу T (год). <code>double GetNeededCapacity(double dW, double dTimeInHours, double dRatio);</code>
15	// Функція вертає вартість електроенергії, спожитої за dDays днів, споживачем потужністю dW (Вт) // вартість електроенергії dKW (грн за кВт*год) <code>double GetElectricityBill(double dW, double dTimeInHours, double dKW);</code>

Контрольні питання

1. Які кроки потрібно виконати щоб створити простий тест використовуючи Google Test?
2. Як визначається результат тесту?
3. Що таке макроси-ствердження?
4. Яка різниця між макросами групи EXPECT_ та макросами групи ASSERT_?
5. Яка є особливість порівняння чисел із плаваючою крапкою? Чому для їх порівняння не підходить макроси EXPECT_EQ() та ASSERT_EQ()?

Додаток 1

1. Перейдіть за посиланням <https://visualstudio.microsoft.com/downloads> та завантажте встановлювач із лінійки Community.

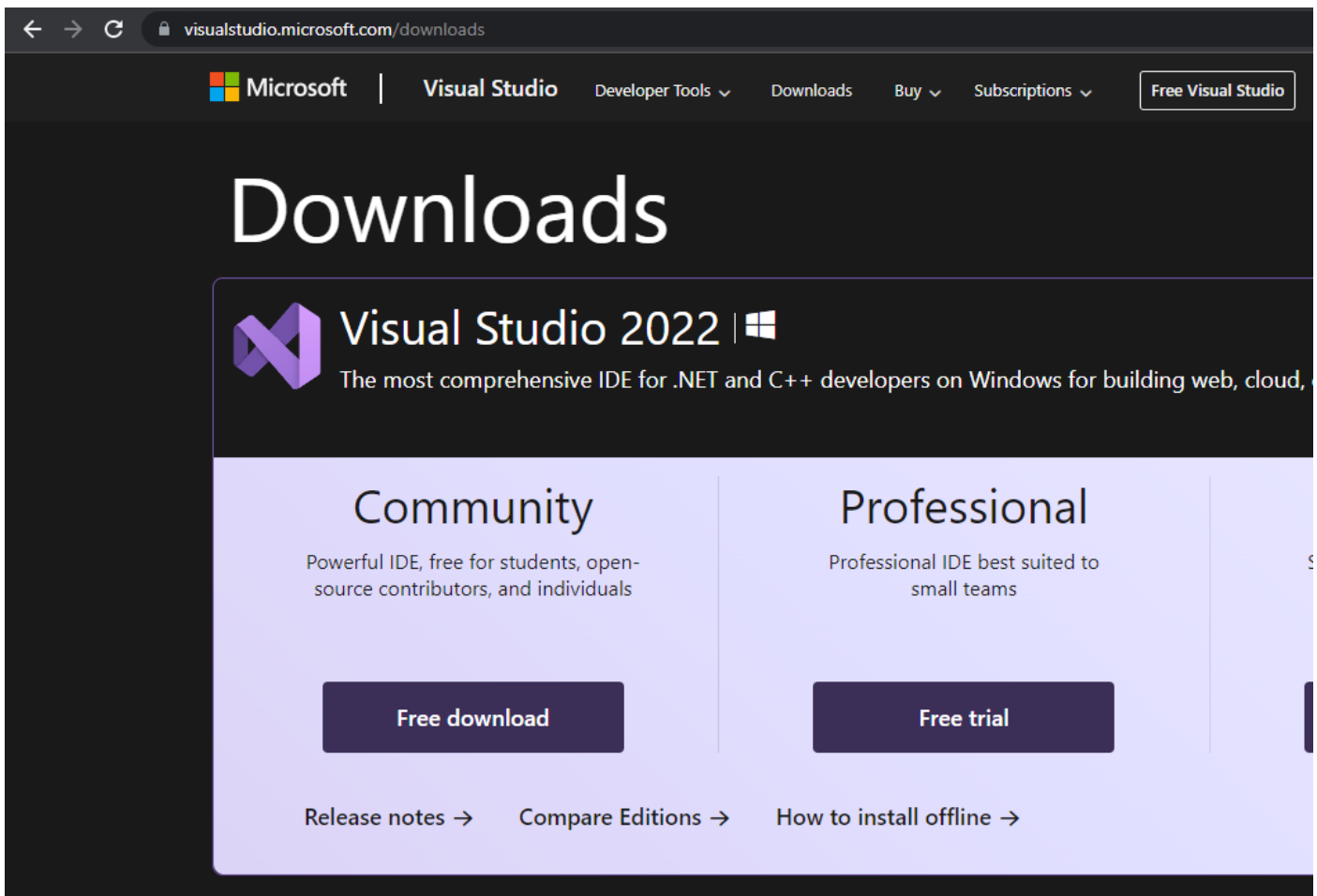


Рис. 21. Сторінка завантаження Microsoft Visual Studio

2. Із директорії Завантаження (Downloads), виконайте VisualStudioSetup.exe щоб почати інсталяцію
3. Під час інсталяції переконайтесь, що обрано набір компонентів 'Desktop development with C++'.

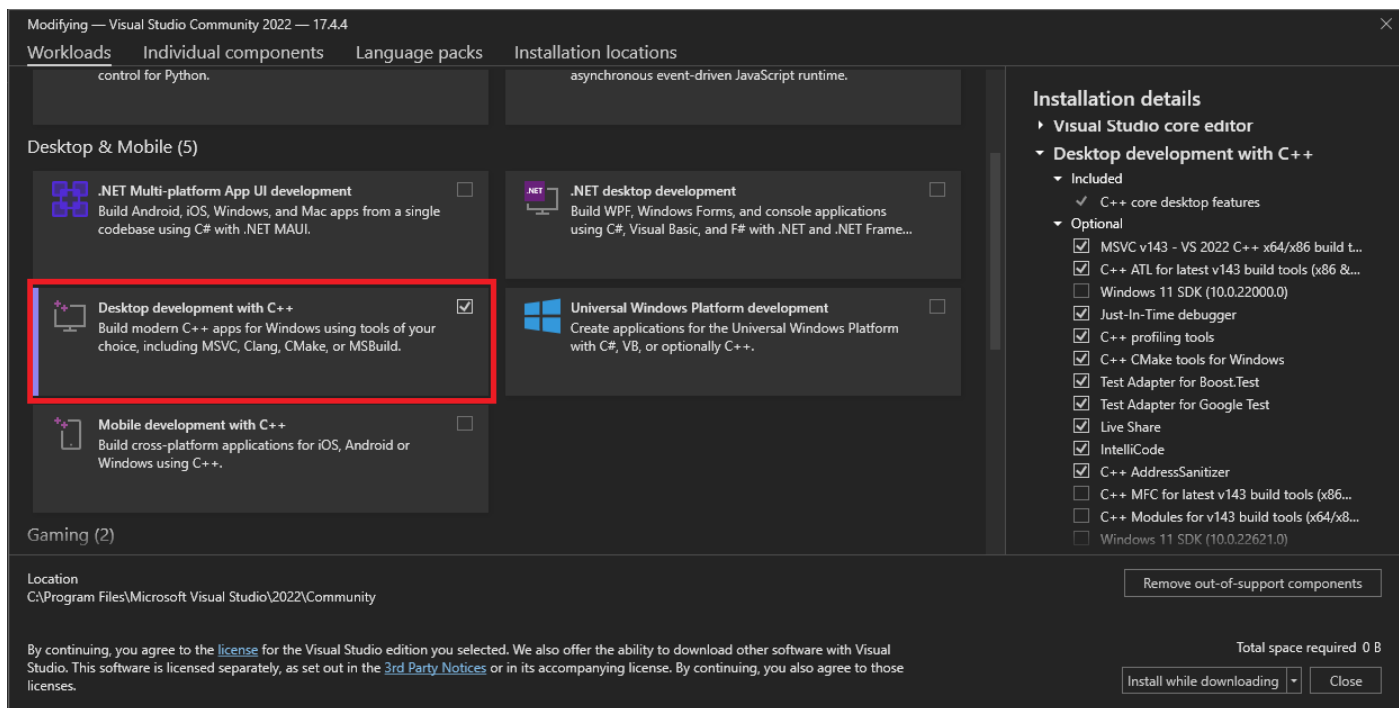


Рис. 22. Вікно вибору набору компонент для Visual Studio під час встановлення