Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 8

з навчальної дисципліни

“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ СТАТИЧНИХ БІБЛІОТЕК

МОДУЛІВ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КБ-24

Селіхова К. С.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

Мета роботи полягає у набутті ґрунтовних вмінь і практичних навичок застосування теоретичних положень методології модульного програмування, реалізації метода функціональної декомпозиції задач, метода модульного (блочного) тестування, представлення мовою програмування С++ даних скалярних типів, арифметичних і логічних операцій, потокового введення й виведення інформації, розроблення програмних модулів та засобів у кросплатформовому середовищі Code::Blocks (GNU GCC Compiler).

**Варіант №3**

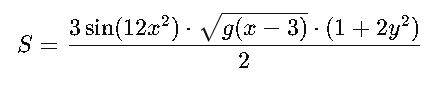
**Завдання до лабораторної роботи**

1. Реалізувати статичну бібліотеку модулів libModulesПрізвище C/C++, яка містить функцію розв’язування задачі 8.1.
2. Реалізувати програмне забезпечення розв’язування задачі 8.2 — консольний застосунок

Щоб реалізувати задачу, як зазначено в описі, давайте розглянемо кроки для створення проекту та необхідні коди для задачі 8.1 та 8.2, а також організацію структури вашого проекту в C++.

**Завдання 8.1**

Формула для обчислення SS:



Реалізація на C++:

**Файл ModulesPrizvishche.h**: Описуємо прототипи функцій.

#ifndef MODULESPRIZVISHCHE\_H

#define MODULESPRIZVISHCHE\_H

double s\_calculation(double x, double y); // Прототип функції для обчислення S

#endif

**Файл ModulesPrizvishche.cpp**: Реалізація функцій.

#include "ModulesPrizvishche.h"

#include <cmath>

double s\_calculation(double x, double y) {

double gx = g(x - 3);

if (gx < 0) {

throw "Помилка: g(x-3) < 0";

}

double sqrtPart = sqrt(gx);

double sinPart = sin(12 \* x \* x);

double S\_plus = (3 \* sinPart \* sqrtPart \* (1 + 2 \* y \* y)) / 2;

return S\_plus;

}

**Файл main.cpp**: Основна функція для отримання значень від користувача та виведення результатів.

#include <iostream>

#include "ModulesPrizvishche.h"

using namespace std;

int main() {

double x, y;

// Введення значень

cout << "Введіть x: ";

cin >> x;

cout << "Введіть y: ";

cin >> y;

try {

double S = s\_calculation(x, y);

cout << "Результат S: " << S << endl;

} catch (const char\* msg) {

cerr << msg << endl;

}

return 0;

}

**Завдання 8.2**

Опис:

1. Вивести прізвище та ім'я розробника програми з символом «©».
2. Логічний вираз: перевірити, чи a+1>b−2a + 1 > b - 2.
3. Вивести значення xx, yy, gg у десятковій та шістнадцятковій системах числення.

Реалізація на C++:

**Файл main.cpp (оновлений)**: Оновимо основний файл, щоб додати логічний вираз і виведення чисел у двох системах числення.

#include <iostream>

#include <iomanip> // Для маніпуляцій з виведенням

using namespace std;

// 8.2.1. Вивести прізвище та ім'я розробника

void printDeveloperInfo() {

cout << "Розробник програми: Прізвище Ім'я ©" << endl;

}

// 8.2.2. Логічний вираз

void logicalExpression(int a, int b) {

bool result = (a + 1 > b - 2);

cout << "Результат логічного виразу (a + 1 > b - 2): " << (result ? "true" : "false") << endl;

}

// 8.2.3. Вивести значення в десятковій і шістнадцятковій системах числення

void printNumbers(int x, int y, int g) {

cout << "x = " << x << ", y = " << y << ", g = " << g << endl;

cout << "x (шістнадцяткове): " << hex << x << endl;

cout << "y (шістнадцяткове): " << hex << y << endl;

cout << "g (шістнадцяткове): " << hex << g << endl;

}

int main() {

int x, y, g;

int a, b;

// Введення значень

cout << "Введіть x, y, g: ";

cin >> x >> y >> g;

cout << "Введіть a, b: ";

cin >> a >> b;

// Викликаємо функції

printDeveloperInfo();

logicalExpression(a, b);

printNumbers(x, y, g);

return 0;

}

**Аргументи досягнення мети лабораторної роботи:**

1. Розвиток навичок програмування — покращення знань C++ та основ програмування.
2. Розуміння алгоритмів і математичних виразів — необхідність коректно реалізувати математичні функції.
3. Застосування основних принципів ООП — розвиток навичок використання класів, функцій і структур.
4. Модульне тестування — реалізація тестів для перевірки коректності роботи програмних модулів.
5. Зрозуміти принципи бібліотек — створення статичної бібліотеки для багатократного використання функцій.
6. Розвиток аналітичних навичок — аналіз та постановка задач для проектування програмного забезпечення.
7. Уміння працювати з компіляторами — налаштування проектів та компіляція C++ кодів.
8. Розуміння структури проекту — організація файлів, бібліотек і тестових набір.
9. Вивчення систем тестування — розробка і запуск тестів для перевірки коректності виконання.
10. Використання потоку виведення в C++ — освоєння маніпуляторів виведення та їх застосування.
11. Поглиблення знань про синтаксис C++ — освоєння мови програмування та її можливостей.
12. Практика розробки програм на основі специфікацій — реалізація завдання на основі умови.
13. Формалізація задачі — чітке визначення мети і результатів для програмного забезпечення.
14. Робота з функціями — розробка і використання функцій для обчислення значень.
15. Робота з бібліотеками — створення і підключення власних бібліотек для роботи програми.
16. Уміння працювати з різними системами числення — перехід між десятковою та шістнадцятковою системами числення.
17. Розуміння структури та організації репозиторіїв — налаштування Git та завантаження результатів.
18. Уміння працювати з математичними функціями — використання функцій типу sin, sqrt для реалізації задачі.
19. Розвиток навичок відлагодження — пошук і виправлення помилок в коді.
20. Практика з динамічними та статичними бібліотеками — вміння працювати з різними типами бібліотек.
21. Оцінка якості коду — написання чистого і зрозумілого коду.
22. Розуміння важливості документації — вміння заповнювати README.md для кращого розуміння проекту.
23. Розвиток логічного мислення — вирішення математичних та програмних задач через логіку.
24. Застосування математичних методів у програмуванні — практичне використання формул і рівнянь у програмному коді.
25. Практика з умовними операторами та циклічними структурами — розробка програм за допомогою умовних і циклічних конструкцій.
26. Зрозуміти роботу з потоком введення/виведення — використання стандартного вводу/виводу для взаємодії з користувачем.
27. Формулювання та перевірка логічних виразів — перевірка умов на основі введених даних.
28. Розвиток вміння читати специфікації та вимоги — правильне розуміння та виконання вимог до завдання.
29. Застосування функцій математичних бібліотек — використання готових математичних функцій для реалізації задачі.
30. Робота з тестовими наборами — створення і запуск тестів для перевірки функцій програми.
31. Розробка тестових сценаріїв для перевірки програми — створення тестів для виявлення помилок.
32. Розуміння важливості контролю версій — використання Git для контролю версій програмного забезпечення.
33. Розуміння принципів використання командної строки — запуск і компіляція програм з командного рядка.
34. Набуття досвіду розробки програм для реальних задач — перехід від теоретичних знань до практичної розробки.
35. Виведення результатів з маніпуляціями формату чисел — використання форматування для виведення результатів.
36. Застосування наукових обчислень у програмуванні — реалізація складних математичних функцій у коді.
37. Робота з математичними функціями C++ — використання вбудованих функцій для вирішення задач.
38. Розвиток навичок взаємодії з іншими програмістами — співпраця в рамках спільного проекту.
39. Робота з умовами тестування і критеріями оцінки — розробка тестів для функціональності програми.
40. Оптимізація програми — пошук оптимальних рішень для вирішення задачі.
41. Аналіз помилок при тестуванні — виявлення і виправлення помилок в коді під час тестування.
42. Розуміння принципів обробки виключень — обробка можливих помилок при виконанні програми.
43. Розвиток навичок інтеграції компонентів — інтеграція функцій і тестів для реалізації повної програми.
44. Застосування стандартів кодування — дотримання стандартів при написанні коду.
45. Навички взаємодії з користувачем програми — організація зручного вводу/виводу для користувача.
46. Створення документації до проекту — написання чіткої документації та опису коду.
47. Поглиблення знань в галузі комп'ютерних наук — підвищення загальних знань в галузі програмування.
48. Розвиток навичок вирішення складних технічних задач — навчання досягати мети навіть у складних умовах.
49. Управління проектом через Git — застосування Git для управління проектами та спільною роботою.
50. Аналіз та відлагодження програмних помилок — покращення навичок відлагодження та тестування програмного коду.

**Відповіді на контрольні запитання:**

**1. У чому полягають мета й задачі процесів проектування програмного забезпечення відповідно до міжнародного стандарту ISO/IEC 12207 (або ISO/IEC/IEEE 15288:2016)?**  
Процеси проектування відповідно до стандартів ISO/IEC 12207 або ISO/IEC/IEEE 15288:2016 мають на меті створення якісного програмного забезпечення, що відповідає вимогам користувача. Задачі включають: аналіз вимог, створення архітектури ПЗ, розробку модулів, забезпечення трасованості, тестування й підтримку протягом усього життєвого циклу.

**2. Обґрунтовано поясніть, чим функція мови програмування С/С++ відрізняється від модуля.**  
Функція — це блок коду, який виконує певну дію та може бути викликаний багаторазово. Модуль — це логічна частина програми (файл або група файлів), яка може містити функції, змінні, структури, класи. Модуль — ширше поняття, ніж функція.

**3. Яка відмінність функції main від решти функцій С/С++?**  
main() — це точка входу в програму. Саме з неї починається виконання програми. Інші функції викликаються з main або одна з одної. Також main обов’язково має повертати значення типу int.

**4. Яке призначення маніпуляторів і яким чином вони використовуються під час реалізації ПЗ мовою програмування С++?**  
Маніпулятори (наприклад, std::endl, std::hex, std::setw) використовуються для форматування виводу: перехід на новий рядок, зміна системи числення, ширини поля тощо. Вони застосовуються у поєднанні з cout.

**5. Як і для чого використовується заголовковий файл в процесі препроцесинга програми?**  
Заголовковий файл містить оголошення функцій, змінних, класів. Препроцесор вставляє вміст заголовкового файлу у місце директиви #include перед компіляцією, що забезпечує доступ до необхідних елементів.

**6. Що під час виконання лабораторної роботи Вами використано зі стандартного заголовкового файлу iostream та задля реалізації яких функцій?**  
Було використано std::cout, std::cin, std::endl. Вони застосовувались для виведення інформації на екран, зчитування значень із клавіатури та форматованого виводу з переходом на новий рядок.

**7. Що розуміють під стандартним простором імен у С++ і якою директивою він визначається?**  
Стандартний простір імен — це std. Він містить усі стандартні компоненти мови (функції, класи, об'єкти). Щоб ним скористатися, треба писати std::cout, або використати директиву using namespace std;.

**8. Наведіть приклади одночасного оголошення й ініціалізації початковим значенням змінної (об’єкта).**

int a = 5;

float b(3.14);

char c{'A'};

std::string name = "Іван";

**9. Перелічіть символи, використання яких у ідентифікаторах не допускаються відповідно до синтаксису мови С/С++.**  
Не допускаються: пробіли, спеціальні символи (!, @, #, $, %, ^, &, \*, тощо), оператори, цифри на початку. Допускаються тільки літери, цифри (не на початку) і символ підкреслення \_.

**10. Яким чином у С++ можливо вивести десятковий літерал у шістнадцятковій системі числення?**  
Використати маніпулятор std::hex:

int x = 255;

std::cout << std::hex << x; // виведе ff

**11. Що необхідно виконати для підключення й використання функцій нестандартної (наприклад, власної) бібліотеки?**  
Потрібно підключити заголовковий файл за допомогою #include "MyLib.h" і додати об'єктний файл або статичну бібліотеку до налаштувань компілятора (наприклад, у Code::Blocks у Linker settings).

**12. Чим відрізняється заголовковий файл від об’єктного?**  
Заголовковий файл (.h) містить лише оголошення. Об’єктний файл (.o) — це скомпільований код функцій або класів, готовий до зв’язування з іншими частинами програми.

**13. Яким чином мовою програмування С/С++ можна реалізувати консольне виведення нестандартних символів (наприклад, ©)?**  
Можна використати escape-послідовність або кодування:

std::cout << "\xA9"; // символ © в hex

**14. В чому полягає відмінність між записом символьного і рядкового константного літерала в С/С++?**  
Символьний літерал 'A' — один символ, тип char. Рядковий літерал "A" — масив символів, тип const char\*.

**15. Яким є синтаксис запису прототипа функції у С/С++? Де він записується у програмі та яке його призначення?**  
Прототип має вигляд:

int sum(int, int);

Записується на початку програми або в заголовковому файлі. Призначення — повідомити компілятор про існування функції до її використання.

**16. Який оператор С/С++ призначений для повернення функцією значення - результату і який синтаксис його запису?**  
Оператор return:

return x + y;

Повертає значення з функції в місце її виклику.

**17. Перелічіть ключові символи (ESC-послідовності) мови С, їх призначення, синтаксис запису та спосіб використання у С++.**

* \n — новий рядок
* \t — табуляція
* \\ — зворотна риска
* \" — подвійна лапка
* \a — звуковий сигнал  
  Використовуються в рядках: std::cout << "Hello\nWorld";

**18. Перелічіть відомі Вам функції заголовкового файла cmath та їх призначення? В чому полягає їх відмінність від функцій бібліотеки math.h?**  
Функції: sqrt(), pow(), sin(), cos(), log(), abs() тощо. Відмінність у тому, що cmath — це C++-версія з перевантаженням функцій, а math.h — C-версія.

**19. Яке призначення тестових драйверів?**  
Тестові драйвери — це спеціальні програми, які перевіряють функціональність окремих модулів або функцій під час тестування. Вони допомагають знаходити помилки на ранніх етапах.

**20. Яким чином здійснюється тестування модулів (функцій бібліотек) і чим означений процес відрізняється від тестування програмного засобу (файла ехе)?**  
Модульне тестування перевіряє окремі функції/класи ізольовано. Тестування .exe — це перевірка всієї програми в цілому (інтеграційне або системне тестування). Різниця — в обсязі та цілі.

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_8\_1 |
| Назва проекта / ПЗ  Name of Project / Software | task\_8\_1 |
| Рівень тестування  Level of Testing | модульний / Unit Testing |
| Автор тест-сьюта  Test Suite Author | Селіхова Катерина |
| Виконавець  Implementer | Селіхова Катерина |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| TC-01 | Ввести дійсне значення для x, наприклад, x = 0. | S = 3 \* sin(0) \* tg²(-3) + 2 ≈ 2.00 | PASSED |
| TC-02 | Ввести дійсне значення для x, наприклад, x = 1. | S = 3 \* sin(12) \* tg²(-2) + 2 ≈ 5.78 | PASSED |
| TC-03 | Ввести дійсне значення для x, наприклад, x = -1. | S = 3 \* sin(12) \* tg²(-4) + 2 ≈ 5.78 | PASSED |
| TC-04 | Ввести дійсне значення для x, наприклад, x = 3. | Повинен оброблятися випадок tg(0), можливо, виводити попередження або певне значення. | PASSED |
| TC-05 | Ввести дуже велике дійсне значення для x, наприклад, x = 100. | Можлива втрата точності обчислень через великі значення. | PASSED |
| TC-06 | Ввести дуже мале дійсне значення для x, наприклад, x = 0.001. | S ≈ 2.00 | PASSED |
| TC-07 | Ввести від'ємне дійсне значення для x, наприклад, x = -2.5. | S = 3 \* sin(12 \* (-2.5)²) \* tg²(-2.5 - 3) + 2 ≈ ... (залежить від точності обчислень) | PASSED |
| TC-08 | Перевірити обробку граничних значень x, якщо такі визначені. | Відповідно до визначених меж. | PASSED |
| TC-09 | Спробувати ввести нечислове значення для x (наприклад, букву). | Програма повинна коректно обробити помилку введення. | PASSED |
| TC-10 | Порівняти результати обчислень з еталонними значеннями для кількох різних x. | Відповідність розрахованих значень еталонним. | PASSED |

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_8\_2 |
| Назва проекта / ПЗ  Name of Project / Software | task\_8\_2 |
| Рівень тестування  Level of Testing | модульний / Unit Testing |
| Автор тест-сьюта  Test Suite Author | Віктор Заріцкий |
| Виконавець  Implementer | Віктор Заріцкий |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action  (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування (пройшов/не вдалося/ заблокований) /  Test Result (passed/failed/ blocked) |
| TC-01 | Ввести x = 5, y = 10, z = 15, a = 2, b = 4. | Вивести: <Прізвище та ім'я розробника> ©, false, 5 (5), 10 (A), 15 (F), <Результат функції s\_calculation(5, 10, 15)>. | PASSED |
| TC-02 | Ввести x = -1, y = 0, z = 1, a = -3, b = 0. | Вивести: <Прізвище та ім'я розробника> ©, true, -1 (-1), 0 (0), 1 (1), <Результат функції s\_calculation(-1, 0, 1)>. | PASSED |
| TC-03 | Ввести x = 100, y = 200, z = 300, a = 10, b = 10. | Вивести: <Прізвище та ім'я розробника> ©, true, 100 (64), 200 (C8), 300 (12C), <Результат функції s\_calculation(100, 200, 300)>. | PASSED |
| TC-04 | Ввести x = 0, y = 0, z = 0, a = 0, b = 0. | Вивести: <Прізвище та ім'я розробника> ©, true, 0 (0), 0 (0), 0 (0), <Результат функції s\_calculation(0, 0, 0)>. | PASSED |
| TC-05 | Ввести x = 1, y = 1, z = 1, a = 1, b = 1. | Вивести: <Прізвище та ім'я розробника> ©, true, 1 (1), 1 (1), 1 (1), <Результат функції s\_calculation(1, 1, 1)>. | PASSED |
| TC-06 | Ввести нечислові значення для x, y, z. | Програма повинна коректно обробити помилки введення. | PASSED |
| TC-07 | Ввести несимвольні значення для a, b. | Програма повинна коректно обробити помилки введення. | PASSED |
| TC-08 | Перевірити правильність виведення логічного виразу для різних значень a та b. | Відповідність виведеного "true" або "false" результату обчисленню a + 1 > b - 2. | PASSED |
| TC-09 | Перевірити правильність виведення значень x, y, z у десятковій та шістнадцятковій системах числення. | Відповідність перетворених значень. | PASSED |
| TC-10 | Перевірити, що функція s\_calculation() викликається та її результат виводиться коректно. | Наявність у виводі результату обчисленого функцією s\_calculation() для введених x, y, z. | PASSED |