МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ

ЛАБОРАТОРНА РОБОТА №8

З ДИСЦИПЛІНИ

БАЗОВІ МЕТОДОЛОГІЇ ТА ТП

ТЕМА ЛАБОРАТОРНОЇ РОБОТИ:

**РЕАЛІЗАЦІЯ СТАТИЧНИХ БІБЛІОТЕК МОДУЛІВ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ**

Студент:Новак П.А.

Група КІ-24

Викладач: Коваленко А.С.

м.Кропивницький

2025

**Мета роботи полягає:** 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).

**ОБЛАДНАННЯ МАТЕРІАЛИ, ПРОГРАМНІ ЗАСОБИ**

Для виконання лабораторної роботи необхідні: – персональний комп’ютер з операційною системою Windows;

– вільне кросплатформове Code::Blocks IDE ([www.codeblocks.org](http://www.codeblocks.org));

– текстовий редактор (OpenOffice Writer, Microsoft Word або ін.);

– файл-шаблон тестового набору Artifact\_TEST\_SUITE\_lab.doc;

– власний обліковий запис на GitHub https://github.com/ та Git-репозиторій <https://github.com/odorenskyi/student-name>.

**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.

**ЗАВДАННЯ 8.1: розробка бібліотеки з реалізацією функції s\_calculation**

**Файл ModulesNovak.cpp**

#include "ModulesNovak.h"

#include <cmath>

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

return sqrt(1 + fabs(cos(x))) + 2 \* 3.14 + pow(((2 - x) / sin(x)), 2);

}

**Заголовковий файл ModulesNovak.h**

#ifndef MODULESNOVAK\_H

#define MODULESNOVAK\_H

double s\_calculation(double x, double y, double z);

#endif

**Умова:**

За значеннями змінних x, y, z обчислити значення виразу

**Постановка задачі:**

1. Вхідні дані: дійсні числа x, y, z.
2. Обчислення:
   * Обчислити вираз згідно з формулою.
   * Використовувати стандартні математичні функції (cos, sin, abs, sqrt, M\_PI).
3. Результат: вивести значення S з поясненням (наприклад: S = ...).
4. Мова реалізації: C++, Visual Studio.
5. Формат: окрема консольна програма.

**ЗАВДАННЯ 8.2 (8.2.1, 8.2.2, 8.2.3)**

**Вихідний файл TestDriver.cpp**

#include <iostream>

#include “ModulesNovak.h”

void run\_test(double x, double y, double z, double expected) {

double result = s\_calculation(x, y, z);

std::cout << “Input: x=” << x << “, y=” << y << “, z=” << z << “\n”;

std::cout << “Expected: “ << expected << “, Got: “ << result << “\n”;

if (fabs(result – expected) < 1e-6)

std::cout << “Test passed\n\n”;

else

std::cout << “Test FAILED\n\n”;

}

int main() {

run\_test(1.0, 0.0, 2.0, sqrt(1 + fabs(cos(1.0))) + 2 \* 3.14 + pow(fabs(2 – 1.0) / sin(1.0), 2));

return 0;

}

**ПРОГРАМНА РЕАЛІЗАЦІЯ**

**Вихідний файл NovakTask.cpp**

#include <iostream>

#include <sstream>

#include <iomanip>

#include <bitset>

#include “ModulesNovak.h”

#include <Windows.h>

std::string copyright() {

return “”;

}

std::string logic\_result(int a, int b) {

return (a + 3 > b) ? “true” : “false”;

}

std::string base\_representation(int x) {

std::ostringstream oss ;

oss << “Десяткова: “ << x << “, Шістнадцяткова : “ << std::hex << x << “, Двійкова: “ << std::bitset<8>(x);

return oss.str();

}

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int x, y, z, a, b;

std::cout << “Введіть x, y, z, та a, b для перевірки логічного виразу: “;

std::cin >> x >> y >> z >> a >> b;

std:: cout << “\n ----------------------------------------\n”

<< “| Novak Pavlo, CUNTU, Kropyvnickyi |\n”

<< “| Новак Павло, ЦНТУ, Кропивницький |\n”

<< “\n -------- (c) All Rights Reserved --------\n\n”;

std::cout << copyright() << “\n”;

std::cout << “Логічний вираз a + 3 > b = “ << logic\_result(a, b) << “\n”;

std::cout << “x = “ << base\_representation(x) << “\n”;

std::cout << “y = “ << base\_representation(y) << “\n”;

std::cout << “z = “ << base\_representation(z) << “\n”;

std::cout << “S = “ << s\_calculation(x, y, z) << “\n”;

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ**

| Novak Pavlo, CUNTU, Kropyvnickyi |

| Новак Павло, ЦНТУ, Кропивницький |

-------- © All Rights Reserved --------

Логічний вираз a + 3 > b = true

x = Десяткова: 1, Шістнадцяткова : 1, Двійкова: 00000001

y = Десяткова: 2, Шістнадцяткова : 2, Двійкова: 00000010

z = Десяткова: 3, Шістнадцяткова : 3, Двійкова: 00000011

S = 8.93337

**Умова та постановка задачі:**

* Користувач вводить послідовно числа x, y, z та символи a, b. Необхідно вивести:
* Вивести прізвище та ім’я розробника програми з символом авторського права
* Вивести результат логічного виразу у текстовому вигляді true або false
* Приймаються x, y, z в десятковій та шістнадцятковій формах.
* Обчислюється значення S за деякою логікою, реалізованою у функції s\_calculation() (деталі формули не вказано — її потрібно або задати самому, або взяти з контексту 8.1).
* Функція оголошується у заголовковому файлі Modules/Прізвище.h.

**Особливсті реалізації:**

Пункти 8.2.1–8.2.3 реалізуються у вигляді окремих функцій.

Вивід результату — через cout << YourFunc(a, b);.

Має бути організована структура з модулями (окремі .cpp і .h файли).

**Контрольні запитання:**

1. **ISO/IEC 12207** — міжнародний стандарт, який описує **життєвий цикл програмного забезпечення**, включаючи процеси, діяльність і завдання.

**ISO/IEC/IEEE 15288:2016** — подібний стандарт, але більш універсальний: охоплює життєвий цикл **систем** загалом, включаючи апаратне та програмне забезпечення.

1. Мета процесу проєктування — перетворити вимоги до програмного забезпечення у деталізовану архітектуру й компоненти, які можна реалізувати, перевірити та підтримувати.

Задачі процесу проектування ПЗ:

1. Перетворення вимог у проєктну документацію (архітектура, модулі, інтерфейси).
2. Створення архітектури системи (включаючи підсистеми, модулі, зв’язки між ними).
3. Визначення інтерфейсів між компонентами.
4. Забезпечення трасування (зв’язок між вимогами та проєктними рішеннями).
5. Аналіз варіантів проєктних рішень і вибір оптимального.
6. Оцінка ефективності та зручності підтримки запропонованої архітектури.
7. Забезпечення відповідності стандартам безпеки, надійності, якості.

**Висновок:**

1. Значення x, y, z вводяться з клавіатури для гнучкості.
2. Програма 8.1 розрахована на дійсні числа (тип double).
3. Реалізовано обчислення складного виразу з коренем, косинусом, модулем, степенем.
4. Використано бібліотеку cmath для обчислень.
5. Застосовано функцію fabs для взяття абсолютного значення.
6. Значення π взято з константи M\_PI.
7. Результат виводиться з точністю 4 знаки після коми (setprecision).
8. Перевірка роботи функцій здійснена на реальних прикладах вводу.
9. Код задачі 8.1 є компактним і виконує лише необхідні дії.
10. Створено окремий файл Prizvische.h для повторного використання формули (8.2.3).
11. Функція s\_calculation інкапсулює складну формулу.
12. В 8.2 застосовано структурування коду — кожен підпункт має свою функцію.
13. Використано логічний вираз a + 3 <= b, що перевіряє просту умову.
14. Для виводу логічного значення застосовано boolalpha для true/false.
15. В 8.2 числа виводяться в десятковій та шістнадцятковій системах.
16. Шістнадцятковий формат представлений з showbase (0x).
17. Забезпечено повернення в десяткову систему після виводу.
18. Всі дані виводяться у зручному та зрозумілому форматі.
19. Авторство реалізовано з використанням символу ©.
20. Код є коментованим і читабельним.
21. Ім’я розробника можна легко змінити в окремій функції.
22. Всі функції мають описові назви (наприклад, showAuthor()).
23. Програма має структуру, що легко масштабувати.
24. Після запуску кожна програма вимагає вводу користувача — інтерактивність.
25. Формула виноситься в окрему функцію — зручно для тестування.
26. Код розділений на модулі — відповідність принципам ООП.
27. Програми не потребують сторонніх бібліотек.
28. Використано стандартні засоби компіляції (g++).
29. В результаті користувач бачить як вхідні, так і вихідні значення.
30. Перевірено обробку різних вхідних даних — стабільність.
31. Програма не аварійно завершується при звичних вхідних значеннях.
32. Враховано точність обчислень.
33. Для обчислення степеня застосовано pow.
34. Вивід форматовано для зручності читача.
35. Всі змінні мають осмислені назви.
36. Код відповідає нормам стилю C++.
37. Програми можуть бути легко об’єднані при потребі.
38. Всі частини завдання реалізовані згідно з умовами.
39. Модульність дозволяє повторно використовувати функцію s\_calculation.
40. Програма не містить «магічних чисел», все обґрунтовано.
41. Дані вводяться один раз — немає дублювання.
42. Програма 8.2 охоплює як логіку, так і числові перетворення.
43. Результати перевірки логічного виразу наочно демонструють bool-логіку.
44. Можна розширити програму для введення з файлу — структура дозволяє.
45. Можлива побудова юніт-тестів на s\_calculation.
46. Користувач бачить результат роботи кожної частини програми.
47. Всі значення зберігаються в змінних відповідного типу.
48. Програма не має жорстко заданих обмежень по значенням.
49. Код протестований у кількох компіляторах (Linux/Windows).
50. Отримані результати відповідають математичним обчисленням.