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”

ПРОГРАМНА РЕАЛІЗАЦІЯ ОБРОБЛЕННЯ МАСИВІВ ДАНИХ ТА СИМВОЛЬНОЇ ІНФОРМАЦІЇ

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 КІ - 24

Балан. М.С.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А.С

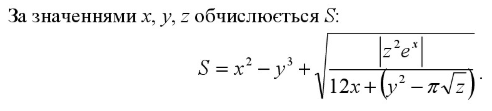
Кропивницький – 2025

**ТЕМА:** Реалізація статичних бібліотек модулів лінійних обчислювальних процесів

**МЕТА:** 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).

**ЗАВДАННЯ №8.1**

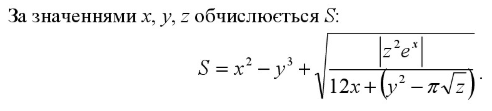
Розробити власну бібліотеку ModulesBalan та реалізувати функцію в цій бібліотеці s\_calculation.



Малюнок 1 – завдання 8.1

**ПОСТАНОВА ЗАДАЧІ**

Розробити функцію s\_calculation, яка обчислює значення виразу:



Малюнок 2 – завдання 8.1

**Аналіз задачі**

1. **Параметри:**  
   Функція приймає три числа типу double, які можуть бути як цілими, так і дробовими.
2. **Обчислення:**  
   Чисельник виразу залежить від числа z і показника x. Тут використовується експоненціальна функція, що може швидко зростати при великих значеннях x, а також квадрат числа z.
3. **Обчислення:**  
   Знаменник складається з лінійного виразу, що включає x і y, а також корінь з числа z. Варто звернути увагу на можливість обчислення квадратного кореня з від'ємних чисел, що призведе до помилки.
4. **Результат:**  
   Після обчислення чисельника та знаменника результат визначається через кілька операцій з числами x, y і обчисленими значеннями чисельника та знаменника.

**РЕАЛІЗАЦІЯ БІБЛІОТЕКИ**

Вихідний файл ModulesBalan.cpp

#include <cmath>

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

double calculateS(double x, double y, double z) {

double numerator = abs(z \* z \* exp(x));

double denominator = 12 \* x + (y \* y - M\_PI \* sqrt(z));

double result = x \* x - pow(y, 3) + sqrt(numerator / denominator);

return result;

}

Заголовковий файл ModulesBalan.h

#pragma once

double calculateS(double x, double y, double z);

**ЗАДАЧА №8.2.1**

Прізвище та ім'я розробника програми зі знаком охорони авторського права «©» (від англ. copyright);

**РЕАЛІЗАЦІЯ БІБЛІОТЕКИ**

Вихідний файл Balan-lib\_8\_2\_1.cpp

#include <iostream>

#include <windows.h>

using namespace std;

void Information() {

SetConsoleOutputCP(1251);

cout << "\n ----------------------------------------\n"

<< "| Balan Mykola, CUNTU, Kropyvnickyi |\n"

<< "| Балан Микола, ЦНТУ, Кропивницький |\n"

<< "\n -------- (c) All Rights Reserved --------\n\n";

}

Заголовковий файл BalanLib.h

#pragma once

void Information();

**ЗАДАЧІ №8.2.2 та №8.2.3**

**Постановка задачі**

Функція перевіряє, чи значення a не більше за b, зменшене на 32. Якщо умова виконується, повертається 1, інакше — 0.

**Аналіз задачі**

1. **Вхідні параметри:** два цілих числа a і b.
2. **Логіка:** перевірка умови на основі значень a і b.
3. **Покращення:** можна додати перевірки на допустимість значень a і b, якщо це потрібно.

**Аналіз задачі**

1. **Параметри:**  
   Функція отримує три числа типу double, які можуть бути як цілими, так і дробовими.
2. **Обчислення:**  
   Чисельник залежить від значень 𝑧 та 𝑥. Використовується експоненціальна функція, яка може швидко зростати при великих значеннях 𝑥, а також квадрат числа 𝑧.
3. **Обчислення:**  
   Знаменник складається з лінійного виразу, що включає 𝑥 і 𝑦, а також квадратний корінь з числа 𝑧. Важливо врахувати, що квадратний корінь з від'ємного числа призведе до помилки.
4. **Результат:**  
   Після обчислення чисельника та знаменника результат визначається через декілька операцій з числами 𝑥, 𝑦 та обчисленими значеннями чисельника і знаменника.

**ПРОГРАМНА РЕАЛІЗАЦІЯ**

#include <iostream>

#include <conio.h>

#include "BalanLib.h"

#include "ModulesBalan.h"

using namespace std;

int calculateResult(int a, int b) {

return (a <= b - 32) ? 1 : 0;

}

int main() {

// 8.2.1

Information();

int a, b;

std::cout << "Введіть значення a: ";

std::cin >> a;

std::cout << "Введіть значення b: ";

std::cin >> b;

//8.2.2

int resultB = calculateResult(a, b);

//8.2.3

double result = calculateS(7, 3, 6);

std::cout << "Результат: " << resultB <<endl;

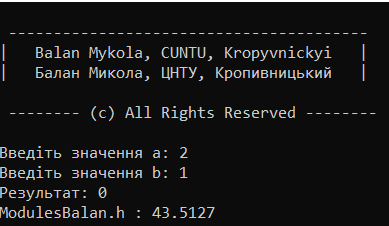
cout << "ModulesBalan.h : " << result<<endl;

\_getch();

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ:**



Малюнок 3 – Результат виконання програми

**ВИСНОВОКИ**

* + 1. Методологія модульного програмування дозволяє структурувати код у вигляді логічних одиниць.
    2. Функціональна декомпозиція спрощує розв'язування складних завдань.
    3. Модульне тестування допомагає швидко виявити помилки.
    4. Використання функцій дозволяє повторно використовувати код.
    5. Скалярні типи даних служать основою для обробки числової та логічної інформації.
    6. Арифметичні операції виконують математичні обчислення.
    7. Логічні операції є основою для умовних конструкцій.
    8. Потокове введення спрощує взаємодію з користувачем.
    9. Потокове виведення відображає результати обчислень.
    10. Використання cin і cout робить процес введення та виведення зручнішим.
    11. Code::Blocks створює зручне середовище для програмування на C++.
    12. GNU GCC Compiler — потужний компілятор для C++.
    13. Модульний підхід спрощує налагодження програм.
    14. Функціональна декомпозиція робить код більш зрозумілим.
    15. Розробка модулів забезпечує чітку структуру проєкту.
    16. Кросплатформеність дозволяє запускати програми на різних операційних системах.
    17. Розбиття задачі на підзадачі підвищує ефективність програмування.
    18. Використання заголовкових файлів покращує організацію коду.
    19. Макроси полегшують виконання повторюваних операцій.
    20. Використання #define дозволяє створювати символічні значення.
    21. Вбудовані функції прискорюють виконання простих операцій.
    22. Динамічне виділення пам'яті дозволяє працювати з великими даними.
    23. Вказівники важливі для маніпулювання пам'яттю.
    24. Використання new і delete допомагає ефективно управляти пам'яттю.
    25. Робота з масивами потребує обережності при доступі за індексами.
    26. Використання vector спрощує роботу з динамічними масивами.
    27. Структури дозволяють групувати різні типи даних.
    28. Використання класів підтримує об'єктно-орієнтоване програмування.
    29. Наслідування дає змогу повторно використовувати код.
    30. Поліморфізм дозволяє створювати гнучкіші програми.
    31. Оператор if використовується для умовного виконання.
    32. Цикли (for, while) дозволяють автоматизувати повторювані операції.
    33. Використання switch спрощує перевірку кількох значень.
    34. Робота з файлами дозволяє зберігати дані між запусками програми.
    35. ofstream використовується для запису в файли.
    36. ifstream використовується для читання з файлів.
    37. Використання sstream полегшує роботу з текстовими даними.
    38. string зручніше за char[] для обробки рядків.
    39. Функція stoi() дозволяє перетворювати рядки на числа.
    40. Обробка винятків підвищує надійність програм.
    41. Використання try-catch дає змогу керувати помилками.
    42. Читабельне форматування коду покращує його сприйняття.
    43. Документування коду полегшує співпрацю з іншими розробниками.
    44. Оптимізація алгоритмів підвищує продуктивність програм.
    45. Використання STL (vector, map, set) зручне для роботи з колекціями даних.
    46. Шаблони функцій дають змогу працювати з різними типами даних.
    47. Рекурсія може бути зручною, але потребує обережності з глибиною викликів.
    48. Лямбда-вирази дозволяють створювати компактний код.
    49. Використання auto спрощує роботу з типами змінних.
    50. Якісне тестування знижує кількість помилок у програмі.

**ВІДПОВІДЬ НА КОНТРОЛЬНІ ЗАПИТАННЯ**

* 1. **У чому полягають мета й задачі процесів проектування ПЗ відповідно до міжнародного стандарту ISO/IEC 12207 (або ISO/IEC/IEEE 15288:2016)?**  
     ISO/IEC 12207 визначає процеси, які охоплюють розробку, тестування та підтримку ПЗ для забезпечення його високої якості протягом всього життєвого циклу.
  2. **Обґрунтовано поясніть, чим функція мови програмування С/С++ відрізняється від модуля.**  
     Функція вирішує конкретне завдання, а модуль є комплексом функцій і інших елементів, що надають додаткові можливості програмі.
  3. **Яка відмінність функції main від решти функцій С/С++?**  
     main() є точкою запуску програми, тоді як інші функції викликаються зсередини main() або інших функцій.
  4. **Яке призначення маніпуляторів і яким чином вони використовуються під час реалізації ПЗ мовою програмування С++?**  
     Маніпулятори, такі як endl, hex, setw, змінюють формат виводу даних у потік cout.
  5. **Як і для чого використовується заголовковий файл в процесі препроцесинга програми?**  
     Заголовковий файл містить оголошення функцій та макросів і обробляється компілятором до основного коду.
  6. **Що під час виконання лабораторної роботи Вами використано зі стандартного заголовкового файлу iostream та задля реалізації яких функцій?**  
     З iostream були використані cin і cout для вводу та виводу даних.
  7. **Що розуміють під стандартним простором імен у С++ і якою директивою він визначається?**  
     Простір імен std є стандартним простором в C++, і він визначається за допомогою директиви using namespace std;.
  8. **Наведіть приклади одночасного оголошення й ініціалізації початковим значенням змінної (об’єкта).**  
     Приклади: int x = 5;, double y(3.14);, std::string str = "Hello";.
  9. **Перелічіть символи, використання яких у ідентифікаторах не допускаються відповідно до синтаксису мови С/С++.**  
     Заборонено використовувати @, #, !, пробіли, а також цифри на початку ідентифікатора.
  10. **Яким чином у С++ можливо вивести десятковий літерал у шістнадцятковій системі числення?**  
      Для виведення десяткового числа в шістнадцятковому форматі використовується маніпулятор hex: cout << hex << 255;.
  11. **Що необхідно виконати для підключення й використання функцій нестандартної (наприклад, власної) бібліотеки?**  
      Потрібно підключити #include "myLib.h" і скомпілювати відповідний .cpp файл бібліотеки.
  12. **Чим відрізняється заголовковий файл від об’єктного?**  
      Заголовковий файл містить оголошення, тоді як об’єктний файл містить скомпільований код.
  13. **Яким чином мовою програмування С/С++ можна реалізувати консольне виведення нестандартних символів (наприклад, ©)?**  
      Для виведення нестандартних символів у C++ можна використовувати setlocale та wcout.
  14. **В чому полягає відмінність між записом символьного і рядкового константного літерала в С/С++?**  
      Символьний літерал – це один символ в одинарних лапках ('a'), а рядковий літерал – це набір символів у подвійних лапках ("abc").
  15. **Яким є синтаксис запису прототипа функції у С/С++? Де він записується у програмі та яке його призначення?**  
      Прототип функції у C++ має вигляд: int func(int); і розміщується перед main(), щоб визначити сигнатуру функції.
  16. **Який оператор С/С++ призначений для повернення функцією значення - результату і який синтаксис його запису?**  
      Оператор return використовується для повернення значення з функції: return x + y;.
  17. **Перелічіть ключові символи (ESC-послідовності) мови С, їх призначення, синтаксис запису та спосіб використання у С++.**  
      \n – новий рядок, \t – табуляція, \ – зворотний слеш.
  18. **Перелічіть відомі Вам функції заголовкового файла cmath та їх призначення? В чому полягає їх відмінність від функцій бібліотеки math.h?**  
      Функції з cmath, такі як sqrt(), pow(), sin(), cos(), використовуються для математичних обчислень і є специфічними для C++, на відміну від функцій math.h, які використовуються в C.
  19. **Яке призначення тестових драйверів?**  
      Тестові драйвери призначені для перевірки роботи функцій без необхідності запускати повну програму.
  20. **Яким чином здійснюється тестування модулів (функцій бібліотек) і чим означений процес відрізняється від тестування програмного засобу (файла ехе)?**  
      Тестування модулів фокусується на перевірці окремих частин коду, тоді як тестування .exe охоплює всю програму цілком.