Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

# ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 8

з навчальної дисципліни “Базові методології та технології програмування”

РЕАЛІЗАЦІЯ СТАТИЧНИХ БІБЛІОТЕК МОДУЛІВ ЛІНІЙНИХ

ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

ВИКОНАВ

студент академічної групи

КІ-21-1 Стрюк В. Є.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

\_\_\_\_\_\_\_\_\_  П.С. Усік

Кропивницький – 2022

# ЛАБОРАТОРНА РОБОТА №8

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

**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 10

**Завдання:**

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.

# ХІД РОБОТИ

## 

Рисунок 1.1 - Умова задачі 8.1

### **Строга постановка задачі:**

Вхідні дані: x, y, z - дійсні знакові числа;

Вихідні дані: S - дійсне число (результат розрахунків);

### **Проектування програмного модуля:**

Модуль ModulesStriuk, що складається з заголовкового файлу (містить оголошення прототипу функції) та файлу вихідного коду (в которому пристуня реалізація функції s\_calculate).

Функція s\_calculate приймає три аргументи в якості вхідних даних, та повертає значення, розраховане за виразом, зазначеним в умові.

Тестовий драйвер TestDriver - функція, що містить в собі:

* масиви для вхідних значень x, y та z відповідно;
* масив значень очікуваних результатів;
* цикл, для виведення еталонних вхідних значень, очікуваного результату та результату розрахунку модуля ModuleStriuk.

Для вихідного коду, див. ДОДАТОК В.

## 

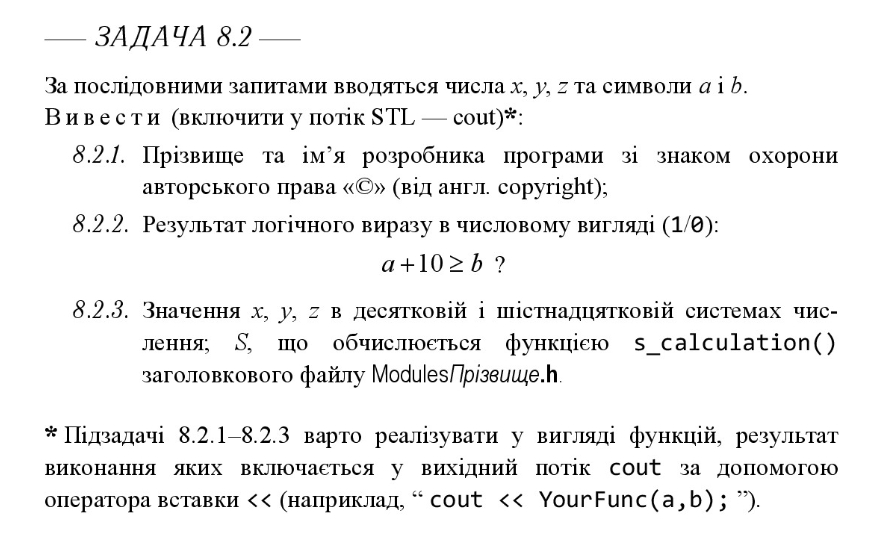


Рисунок 1.2 - Умова задачі 8.2

### **Строга постановка задачі:**

Вхідні дані:

* x, y, z - цілочисельні знакові числа;
* a, b - символьні літерали;

Вихідні дані:

* Ім’я розробника з символом copyright;
* Результат логічного виразу: a + 10 >= b;
* x, y, z в десятковій та шістнадцятковій системі числення;
* Результат функції s\_calculate з модуля ModulesStriuk;

### **Процедурна алгоритмізація:**

Модуль ModulesStriuk використовуємо з попереднього проекту.

inputVariables - функція, що приймає 5 аргументів (x, y, z, a, b) та використовує їх для запиту на введення користувачем.

authorCopyright - виводить прізвище ім’я по-батькові розробника, з використанням символу copyright.

logicalExpression - виводить у потік числове логічне значення (1/0) виразу a + 10 >= b;

decimalHexNumbersOutput - слугує для виведення значень x, y, z у десятковій та шістнадцятковій системі числення.

Вищеперераховані функції викликаються в main, а також - до потоку включається функція s\_calculate що приймає аргументи x, y, z та повертає результат обчислень.

# ВИСНОВКИ

Дана лабораторна робота націлена на набуття навичок у використанні більш упорядкованої парадигми програмування. На відміну від процедурної, де всі функції реалізовані в одному кодовому файлі (що на достатньо масивних проектах викликає ієрархію) - модульна фігурує поняттям “статичної бібліотеки”. Статичні бібліотеки містять в собі інформацію о модулі, що реалізований в якості .h та .cpp файлів.

Заголовковий файл .h формату складається з препроцессорних макропідстановок та оголошених прототипів функцій. #ifndef - макрос, що перевіряє визначеність файлу. Якщо файл не визначено - виконуються #define макропідстановка, що задає файлу визначеності. Внаслідок цього, виконується наступний набір операторів, а в кінці вихідного файлу, перевірка на визначеність файлу припиняється макросом #endif. Інакше кажучи - дані макроси дозволяють, або забороняють компіляцію виділеної ними частини коду.

Файл вихідних кодів формату .cpp містить ініціалізацію функцій, раніше оголошених в .h файлі. Потребує підключення директиви .h файлу через макропідстановку #include для належної реалізації функцій.

Робота була виконана за допомогою дистрибутиву C++ та кроссплатформенного IDE - QtCreator 5.12.5. Дане програмне забезпечення дозволяє виконати лінкування файлів за допомогою відповідних команд системи збору QML. В корінному каталозі проекту міститься файл формату .pro, в якому були вказані всі заголовкові файли та файли вихідного коду (зокрема і main.cpp). Рішення про зміну середовища розробки аргументується тим, що лінкування QtCreator на відміну від лінкування, реалізованому в Code::Blocks, є більш наочним та визначеним, що дозволяє безпосередньо в редакторі коду налаштовувати базові поля збірки проекту системою QML.

При виконанні завдань 8.1 та 8.2 був задіяний метод висхідного проектування програмного забезпечення, що полягав у початковій реалізації менш значимих (або допоміжних) функцій нижнього рівня. Такою функцією виявилась s\_calculate, з модуля ModulesStriuk, розробленим на етапі задачі 8.1. До даного застосовувались всі правила побудови файлів вихідного коду та заголовкового. Надалі, модуль ModulesStriuk використовувався в main функціях обох задач.

Також, необхідно виділити увагу бібліотеці <cmath>, що використовувалась для операцій всередині s\_calcuate. Базова математична функція log(int) використовувалась замість ln (натуральный логарифм), адже вона відсутня у стандартній бібліотеці cmath (або math.h).

Реалізація тестового модуля визначена в main.cpp проекту TestDriver. Основною метою створення даного модуля була автоматизація процесу Unit Testing, що зобов'язує ізоляцію певної процедури всередині цього ж модуля. Це гарантує більш якісне проведення процесу тестування, адже вірогідність виявити залежності та позбутися від них - збільшується.

Для використання та подальшого порівняння з очікуваним результатом, розробляємо множину контрольних прикладів (вхідних даних) в якості одновимірного масиву для кожної вхідної змінної. Окремо від функціонуючої програми вирішуємо приклад з задачі 8.1 та створюємо масив очікуваних даних. У вигляді циклу, та виведення повідомлень створюємо модуль системного тестування. Результатом його виконання буде наочне представлення еталонних значень, очікуваних та отриманих результатів у консольному вікні. На основі отриманих та завчасно створених даних формуємо TestSuite.

Завдання 8.2 потребує використання відлагодженого модуля ModulesStriuk. Проект 8.2 користується як процедурною, так і модульною парадигмою програмування (дана потреба викликана умовою задачі). Завдання націлене на детальний огляд роботи потокового виведення за допомогою cout (з простору імен std), функціонування маніпуляторів форматованого виведення та запису виклику функції безпосередньо в потік виведення.