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/

ВИКОНАВ
студент академічної групи КН-24
Ковальова Єва

ПЕРЕВІРИВ
ст. викладач кафедри кібербезпеки
та програмного забезпечення
Коваленко А.С

Кропивницький – 2024

ЛАБОРАТОРНА РОБОТА № 8

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 C++ 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 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.

Варіант 11

— *Задача 8.1* —

За значеннями x, y, z обчислюється S:

$$S = z \sin\left(x^2 \cdot y\right) + \frac{\sqrt{|z - 12\,x|}}{y^3}.$$

— *Задача 8.2* —

За послідовними запитами вводяться числа x, y, z та символи a і b.

Вивести (включити у потік STL — cout)*:

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

8.2.2. Результат логічного виразу в текстовому вигляді (`false`/`true`):

$$a + 1 = b + 2\ ?$$

8.2.3. Значення x, y, z в десятковій і шістнадцятковій системах числення; S, що обчислюється функцією `s_calculation()` заголовкового файлу Modules*Прізвище*.**h**.

* Підзадачі 8.2.1–8.2.3 варто реалізувати у вигляді функцій, результат виконання яких включається у вихідний потік `cout` за допомогою оператора вставки `<<` (наприклад, "`cout << YourFunc(a,b);`").

Задача 8.1

Аналіз вимог:

Вхідні дані: x, y, z.

Обчислення:

$$S = z \sin\left(x^2 \cdot y\right) + \frac{\sqrt{|z - 12\,x|}}{y^3}.$$

Вихідні дані: Значення S

Вихідний код ModulesKovalova.cpp :

```
#include <cmath>
#include <cstdlib>
#include <string>
#include "ModulesKovalova.h"

double s_calculation(double x, double y, double z) {
    // Перевірка ділення на нуль
    if (y == 0) {
        return NAN;
    }

    double denominator = y * y * y;
    double sinPart = z * sin(x * x * y);
    double sqrtArg = fabs(z - 12 * x);
    double sqrtPart = sqrt(sqrtArg);

    return sinPart + sqrtPart / denominator;
}
```

ModulesKovalova.h :

```
#ifndef MODULESKOVALOVA_H_INCLUDED
#define MODULESKOVALOVA_H_INCLUDED


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

#endif
```

TestDriver :

```
#include <iostream>
#include <windows.h>
#include "ModulesKovalova.h"

using namespace std;

int main()
```

```
{
    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);

    double x, y, z;

    cout << "Введіть x: " << endl;
    cin >> x;
    cout << "Введіть y: " << endl;
    cin >> y;
    cout << "Введіть z: " << endl;
    cin >> z;

    double result = s_calculation(x, y, z);

    cout << "Відповідь: " << result << endl;

    return 0;
}
```

Задача 8.2
Аналіз вимог:

Вхідні дані: x, y, z, a, b.

Вихідні дані:
Ковальова Єва ©, Усі права захищіні.
Результат логічного виразу (a + 1 = b + 2):
false
Значення в десятковій та шістнадцятковій
системах:
Значення S, що обчислюється функцією

Лістининг:

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <windows.h>
#include "ModulesKovalova.h"

using namespace std;


void copyright_info() {
    cout << "Ковальова Єва ©, Усі права захищіні." << endl;
}

void logical_expression(int a, int b) {
    cout << "Результат логічного виразу (a + 1 = b + 2): "
         << ((a + 1 == b + 2) ? "true" : "false") << endl;
}

void display_values(double x, double y, double z) {
    cout << endl << "Значення в десятковій та шістнадцятковій системах:" <<
endl;
```

```
    cout << "x = " << x << " (десяткова), "
         << "0x" << hex << uppercase << (int)x << nouppercase << dec << "
(шістнадцяткова)" << endl;

    cout << "y = " << y << " (десяткова), "
         << "0x" << hex << uppercase << (int)y << nouppercase << dec << "
(шістнадцяткова)" << endl;

    cout << "z = " << z << " (десяткова), "
         << "0x" << hex << uppercase << (int)z << nouppercase << dec << "
(шістнадцяткова)" << endl;

    double S = s_calculation(x, y, z);
    cout << "Значення S, що обчислюється функцією s_calculation(): " << S <<
dec << endl;
}

int main()
{
    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);

    int a, b;
    double x, y, z;
    cout << "Введіть значення a: " << endl;
    cin >> a;
    cout << "Введіть значення b: " << endl;
    cin >> b;

    cout << "Введіть x: " << endl;
    cin >> x;
    cout << "Введіть y: " << endl;
    cin >> y;
    cout << "Введіть z: " << endl;
    cin >> z;

    cout << endl;

    copyright_info();
    logical_expression(a, b);
    display_values(x, y, z);
    return 0;
}
```

Аналіз реалізації функції s_calculation

Функція s_calculation обчислює:

$$S = z \sin\left(x^2 \cdot y\right) + \frac{\sqrt{|z - 12x|}}{y^3}.$$

- Перевіряє, чи y == 0, щоб уникнути ділення на нуль (повертає NAN).
- Використовує sin(), fabs(), sqrt().
- Корінь не дасть помилку, бо fabs() забезпечує невід’ємний аргумент.

Аналіз вимог та висновки для Задачі 8.2:

Вимоги:

Вхідні дані: Числа x, y, z. Символи a і b.
Вихідні дані: Результат логічного виразу (true/false), значення x, y, z в 10-вій та 16-вій системах числення, значення S, обчислене функцією s_calculation.

Висновки:

1. **Вхідні дані:**
   Вхідні дані є зрозумілими і не потребують складних перевірок, окрім можливих обмежень на типи значень для x, y, z та символів a і b. Для логічного виразу важливо чітко визначити умови, за яких потрібно вивести true чи false.
2. **Вихідні дані:**
   - Виведення чисел у двох системах числення є стандартною задачею для обчислення та форматування.
   - Використання функції s_calculation дозволить отримати результат за заданою формулою, при цьому треба буде врахувати можливу помилку через ділення на нуль.
3. **Рекомендації для реалізації:**
   - Перевірити коректність значень x, y, z (особливо на нуль для y, щоб уникнути ділення на нуль).
   - Переконатися, що всі виведені дані правильно форматовані для 10 та 16 систем числення.
   - Логічний вираз треба чітко сформулювати, щоб точно визначити умови для true/false.

50 Аргументів

1. Статичні бібліотеки дозволяють ефективно повторно використовувати код.
2. Модульне програмування покращує організацію складних проектів.
3. Функціональна декомпозиція сприяє чіткому структуруванню коду.
4. Створення власних бібліотек покращує розуміння принципів роботи стандартних бібліотек.
5. Робота з Git розвиває навички командної розробки програмного забезпечення.
6. Модульне тестування підвищує надійність коду.
7. Статичні бібліотеки пришвидшують час компіляції при повторному використанні.
8. Вивчення Code::Blocks розширює досвід роботи з IDE.
9. Розробка модульних програм спрощує підтримку та оновлення коду.
10. Статичне зв'язування забезпечує стабільність програмного забезпечення.
11. Практична робота з GitHub покращує навички версійного контролю.
12. Тестування з використанням Artifact_TEST_SUITE розвиває навички автоматизованого тестування.
13. Розуміння лінійних обчислювальних процесів є основою для складніших алгоритмів.
14. Статичні бібліотеки забезпечують швидшу роботу програми порівняно з динамічними.
15. Модульне програмування дозволяє розподілити роботу в команді.
16. Абстракція даних через модулі покращує структуру програми.
17. Застосування теоретичних положень на практиці закріплює знання.
18. Робота з компілятором GNU GCC розширює розуміння процесу компіляції.
19. Структурування програми на модулі покращує її масштабованість.

20. Інкапсуляція функціональності в бібліотеці спрощує інтерфейс програми.
21. Крос-платформна розробка розширює спектр застосування створеного програмного забезпечення.
22. Створення бібліотеки навчає стандартизації інтерфейсів.
23. Функціональний підхід до декомпозиції задач розвиває аналітичне мислення.
24. Тестування підвищує якість програмного забезпечення перед випуском.
25. Оптимізація лінійних обчислювальних процесів покращує продуктивність програм.
26. Документування бібліотек є важливою навичкою професійного розробника.
27. Створення консольних застосунків навчає розробляти програми без залежності від GUI.
28. Практика з арифметичними та логічними операціями поглиблює розуміння низькорівневих аспектів програмування.
29. Статичне зв'язування зменшує залежності при розгортанні програми.
30. Розробка власних модулів розвиває навички проектування програмних систем.
31. Практична робота з текстовими редакторами для розробки коду покращує навички використання цих інструментів.
32. Уміння розв'язувати задачі через створення власних бібліотек підвищує ефективність розробки.
33. Використання стандартних типів даних з C++ покращує портативність коду.
34. Модульне програмування сприяє ізоляції помилок і полегшує їх виявлення.
35. Реалізація власної бібліотеки допомагає розуміти принципи організації коду в великих проектах.
36. Навички роботи з потоковим введенням та виведенням є основними для більшості програм.
37. Статичні бібліотеки спрощують розгортання програми на клієнтських машинах.
38. Розуміння принципів компіляції та лінкування кодової бази покращує здатність до усунення помилок.
39. Створення бібліотек відповідно до конкретних задач оптимізує роботу програми.
40. Робота з репозиторіями Git розвиває навички колаборативної розробки.
41. Блочне тестування допомагає виявити помилки на ранніх етапах розробки.
42. Дотримання стандартів програмування під час створення бібліотек підвищує читабельність коду.
43. Навички написання функцій для вирішення конкретних задач підвищують кваліфікацію програміста.
44. Документування методів бібліотеки полегшує її подальше використання.
45. Практика з модульним програмуванням закладає основи об'єктно-орієнтованого підходу.
46. Розробка бібліотек вимагає абстрактного мислення і планування.
47. Правильна структура проекту з модулями полегшує його підтримку та розширення.
48. Робота з компілятором GNU GCC розширює знання з різних стандартів C++.
49. Навички розробки консольних застосунків важливі для системного програмування.
50. Використання власного облікового запису на GitHub допомагає формувати професійне портфоліо розробника.

Контрольні запитання

Запитання 1:
У чому полягають мета й задачі процесів проєктування програмного забезпечення відповідно до міжнародного стандарту ISO/IEC 12207 (або ISO/IEC/IEEE 15288:2016)?
Мета процесів проєктування програмного забезпечення згідно з міжнародним стандартом ISO/IEC 12207 (або ISO/IEC/IEEE 15288:2016) полягає у створенні якісного, надійного та ефективного програмного забезпечення, що задовольняє вимоги користувачів та забезпечує досягнення бізнес-цілей.
Основні задачі цих процесів:

1. Визначення та аналіз вимог до програмного забезпечення
2. Проєктування архітектури програмного забезпечення
3. Детальне проєктування компонентів та інтерфейсів
4. Реалізація (кодування) програмного забезпечення
5. Тестування та валідація розробленого ПЗ
6. Впровадження та обслуговування програмного продукту
7. Належна документація на всіх етапах розробки
8. Управління конфігурацією та версіями
9. Забезпечення якості програмного забезпечення
10. Управління змінами та ризиками протягом життєвого циклу ПЗ
11. Системна інтеграція різних компонентів
12. Верифікація відповідності програмного продукту вимогам

Стандарт встановлює спільну структуру процесів, термінологію та практики, що забезпечують систематичний підхід до розробки програмного забезпечення та його подальшого супроводу.

Запитання 2:
Обґрунтовано пояснити, чим функція мови програмування C/C++ відрізняється від модуля.
Функція і модуль у C/C++ відрізняються за масштабом, призначенням та організацією:
Функція в C/C++:

- Це окрема одиниця коду, що виконує визначену логічну операцію
- Складається з заголовка (прототипу) та тіла функції
- Може приймати параметри та повертати значення
- Має локальну область видимості для змінних
- Є елементарною одиницею повторного використання коду
- Може бути викликана багаторазово з різних частин програми

Модуль в C/C++:

- Логічна одиниця організації коду, що об'єднує групу взаємопов'язаних функцій, структур даних та інших елементів
- Зазвичай реалізується через файли заголовків (.h) та файли реалізації (.c/.cpp)
- Забезпечує інкапсуляцію даних та функціональності
- Може містити багато функцій, класів, змінних та констант
- Визначає чіткі інтерфейси взаємодії з іншими частинами програми

- Дозволяє контролювати видимість елементів через ключові слова static, extern, namespace тощо
- Сприяє організації великих проєктів через логічне розділення за функціональністю

Основні відмінності: модуль є більш високорівневою абстракцією, що може містити багато функцій та інших елемен’тів, тоді як функція є одиничним елементом коду з конкретним призначенням. Модуль забезпечує організацію та структурування програми, сприяє повторному використанню коду та дотриманню принципів інкапсуляції.