Прізвище: Хоменко
Ім'я: Віталій
Група: КН-108
Варіант: 11

Кафедра: САПР
Дисципліна: Об`єктно-орієнтоване програмування
Перевірив: Головатий А.І.

ЗВІТ
до лабораторної роботи №8
на тему "Динамічне під'єднання бібліотеки (DLL)"

Мета роботи: Ознайомитись з методикою створення та використання dll-бібліотеки.

Індивідуальне завдання:
Завдання 1:
Написати програму для виконання геометричних обчислень відповідно до власного варіанту. Створені функції для реалізації геометричних формул виділити в dll-бібліотеку.
11. Обчислити площу повної поверхні прямого кругового конуса, якщо відомі радіус кола основи та довжина твірної бічної поверхні.

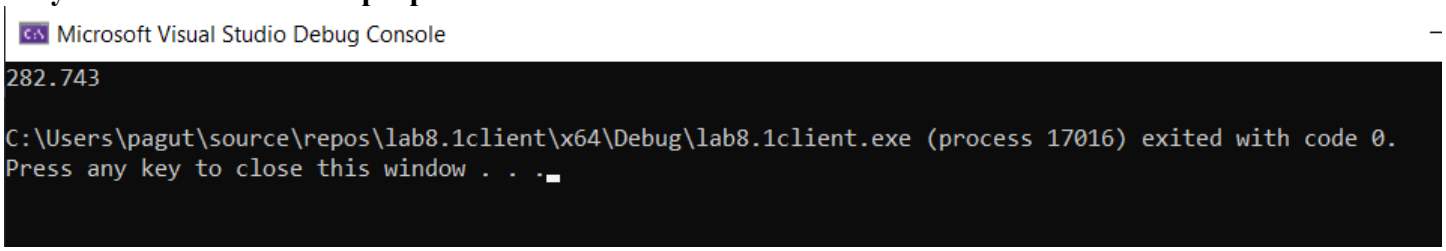
Код програми:

MathLibrary.cpp
```
#include "pch.h"
#include<cmath>
#include "Source.h"
# define PI 3.14159265358979323846264338327950288 4L
const long double coneArea(const long double radius, const long double slantHeight)
{
	return PI * pow(radius, 2) + PI * radius * slantHeight;
```

MathLibrary.h
```
#pragma once
#ifdef MATHLIBRARY_EXPORTS
#define MATHLIBRARY_API __declspec(dllexport)
#else
#define MATHLIBRARY_API __declspec(dllimport)
#endif
extern "C" MATHLIBRARY_API const long double coneArea(const long double radius, const long
double slantHeight);
```

MathClient.cpp
```
#include <iostream>
#include "source.h"

int main()
{
    std::cout << coneArea(6, 9) << std::endl;
    return 0;
}
```

Результати виконання програми:



Завдання 2:
Створимо DLL-бібліотеку на мові C++ та виклик її функції з Windows-проекту.
1. Створити рішення
2. Створити DLL-бібліотеку, яка містить функції, що необхідні для реалізації індивідуального завдання.
3. Створити в тому ж рішенні Windows-проект, який викликає ці бібліотечні методи для виконання необхідних обчислень. Форма для введення вхідних даних та виведення результатів повинна бути достатньо інформативною, щоб користувачу було зрозуміло, які саме обчислення і за якими формулами виконуються у програмі.
4. При введенні вхідних даних перевіряти їх на допустимість значень. Наприклад, якщо використовується операція ділення, перевіряти, щоб знаменник не дорівнював нулю. Діапазони допустимих значень описати у звіті. Проілюструвати скріншотами роботу програми на допустимих та недопустимих значеннях вхідних даних (повідомлення про помилки, або невірно введені дані).

11.
1. Обчислити середнє арифметичне та середнє геометричне трьох дійсних чисел.
2. Обчислити функцію arccos(x) через ряд Тейлора:

$$arccos\, x = \frac{\pi}{2} - \arcsin x = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)!} x^{2n+1}$$ для |x|<1.

Значення x – дійсне число, n – ціле (кількість членів ряду). Значення x та n вводяться у формі. Результат вивести на форму разом з результатом, отриманим з застосуванням вбудованих функцій для тригонометричних обчислень та порівняти їх для різних значень n.
3. Обчислити площу та радіус вписаного у рівнобедрену трапецію кола за сторонами цієї трапеції.

Код програми:

MathFunction.h

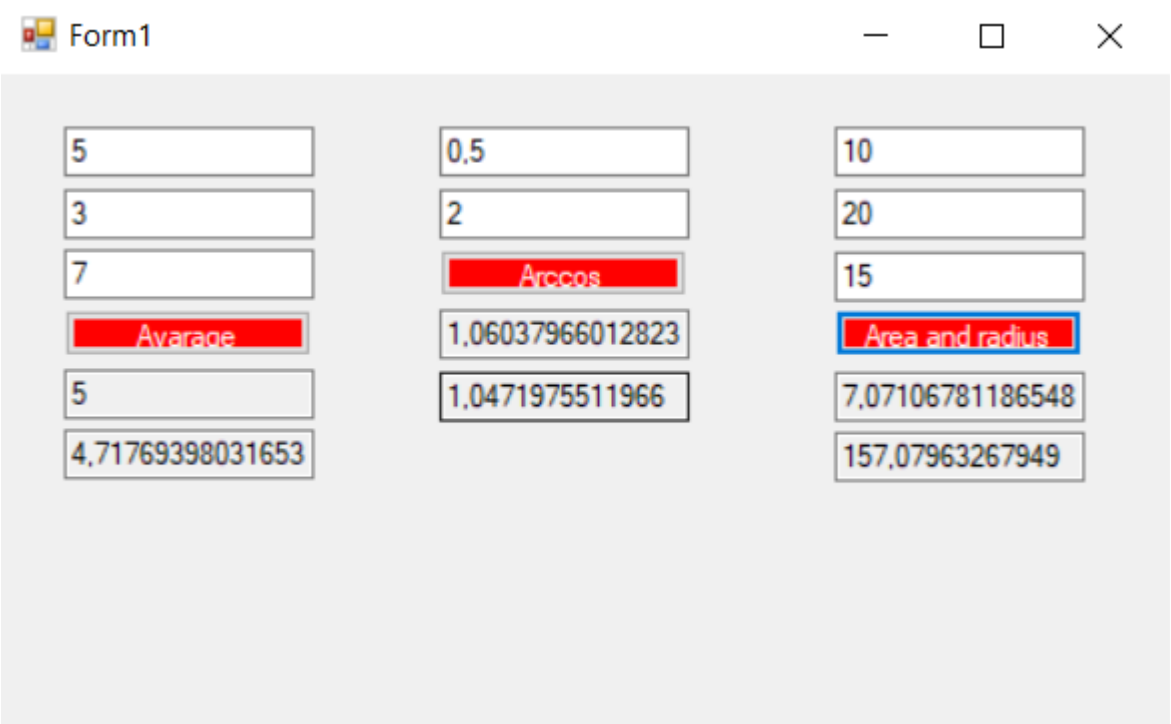
```
#pragma once
#ifdef MATHLIBRARY_EXPORTS
#define MATHLIBRARY_API __declspec(dllexport)
#else
#define MATHLIBRARY_API __declspec(dllimport)
#endif

using namespace std;

extern "C" MATHLIBRARY_API const long double* average(const long double a, const long double b,
const long double c);
extern "C" MATHLIBRARY_API double arcCos(const double x, int n);
```

```
extern "C" MATHLIBRARY_API double* radius(double x, double y, double z);
```

MathFunction.cpp:

```
#include "pch.h" // use stdafx.h in Visual Studio 2017 and earlier
#include "MathFunctions.h"
#include <stdexcept>
#define PI 3.14159265358979323846

using namespace std;

const long double* average(const long double a, const long double b, const long double c) {
    long double* result = new long double[2];
    result[0] = (a + b + c) / 3;
    result[1] = pow(a * b * c, 1.0 / 3);
    return result;

}
int factorial(int n)
{
    if (n > 1)
        return n * factorial(n - 1);
    else
        return 1;
}


double arcSin(double x, int n) {
    long double y = 0;
    for (int i = 0; i < n; i++)
    {
        y += (factorial(2 * i) / (pow(4.0, i) * pow(factorial(i), 2.0) * factorial(2 * i + 1)))
* pow(x, 2.0 * i + 1.0);
    }
    return y;
}
double arcCos(const double x, int n)
{
    if (x > 1 || x < -1)
    {
        throw std::invalid_argument("Невірно введені дані. Косинус знаходиться в межах від -1 до
1");
    }
    if (n < 0) {
        throw std::invalid_argument("Кількість елементів ряду повинна бути додатньою");
    }

    return PI / 2.0 - arcSin(x, n);

}
double radius(double x, double y) {
    return pow(x * y, 0.5) / 2.0;
}
double area(double x, double y) {
    return PI * pow(radius(x, y), 2);
}
double* radius(double x, double y, double z) {
    double* result = new double[2];
    if (x < 0 || y < 0 || z < 0) {
        throw std::invalid_argument("Довжина сторін трапеції повинна бути додатньою");
    }

    if ((x + y) != (z * 2))
    {
        throw std::invalid_argument("Неможливо вписати коло в трапецію");
    }
    result[0] = radius(x, y);
    result[1] = area(x, y);
    return result;
}
```

Результат виконання роботи:



Висновок: ознайомився з методикою створення та використання dll-бібліотеки.

Відповіді на контрольні запитання:

1. Що таке dll-бібліотека?

Бібліотека DLL – це бібліотека, що містить код та дані, які можуть використовуватися кількома програмами одночасно.

2. Для чого використовуються dll-бібліотеки?

Вони викристовуються для того, щоб замінити велику кількість коду, викликом однієї функції замість цього

3. Яким чином експортується функція з dll-бібліотеки?

Динамічне завантаження застосовна і до ресурсів DLL, використовуваним MFC для завантаження стандартних ресурсів програми. Для цього спочатку необхідно ви-
кликати функцію LoadLibrary і розмістити DLL в пам'яті. Потім за допомогою функції AfxSetResourceHandle потрібно підготувати вікно програми до прийому ресурсів з знову завантаженої бібліотеки. В іншому випадку ресурси будуть заванта-
жуватися з файлів, підключених до виконуваного файлу процесу. Такий підхід зру-
чний, якщо потрібно використовувати різні набори ресурсів, наприклад для різних мов.

Зауваження. За допомогою функції LoadLibrary можна також завантажувати в пам'ять виконувані файли (не запускати їх на виконання!). Дескриптор виконуваного модуля може потім використовуватися при зверненні до функцій FindResource
і LoadResource для пошуку і завантаження ресурсів програми. Вивантажують модулі з пам'яті також за допомогою функції FreeLibrary.

4. Яким чином імпортується функція з dll-бібліотеки?

Переважна більшість DLL імпортує функції із системних DLL (kernel32.dll, user32.dll, gdi32.dll і ін.). У більшості випадків при створенні бібліотеки до неї авто-

матично підключається стандартний набір таких бібліотек. Іноді в цей список необ-
хідно додати необхідні DLL (наприклад, у випадку використання бібліотеки сокетів потрібно додатково підключити бібліотеку ws2_32.dll).