Ministry of Education and Science of Ukraine

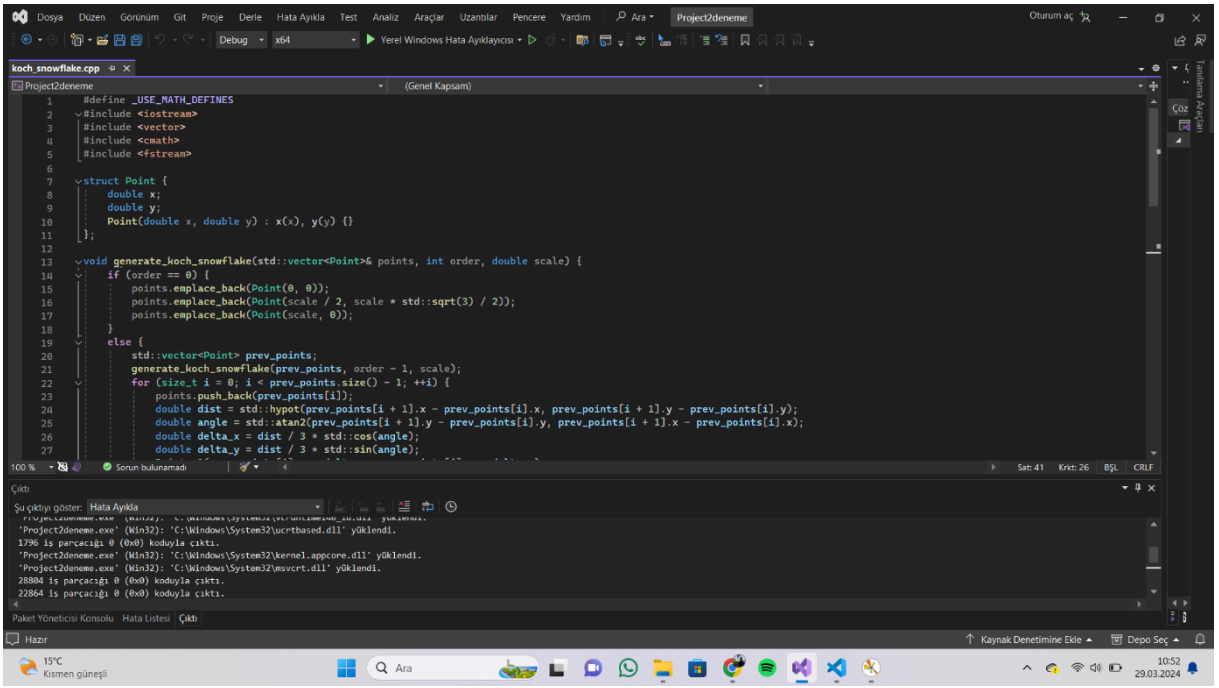
National Technical University of Ukraine

"Igor Sikorsky Kyiv Polytechnic Institute"

Laboratory work #4

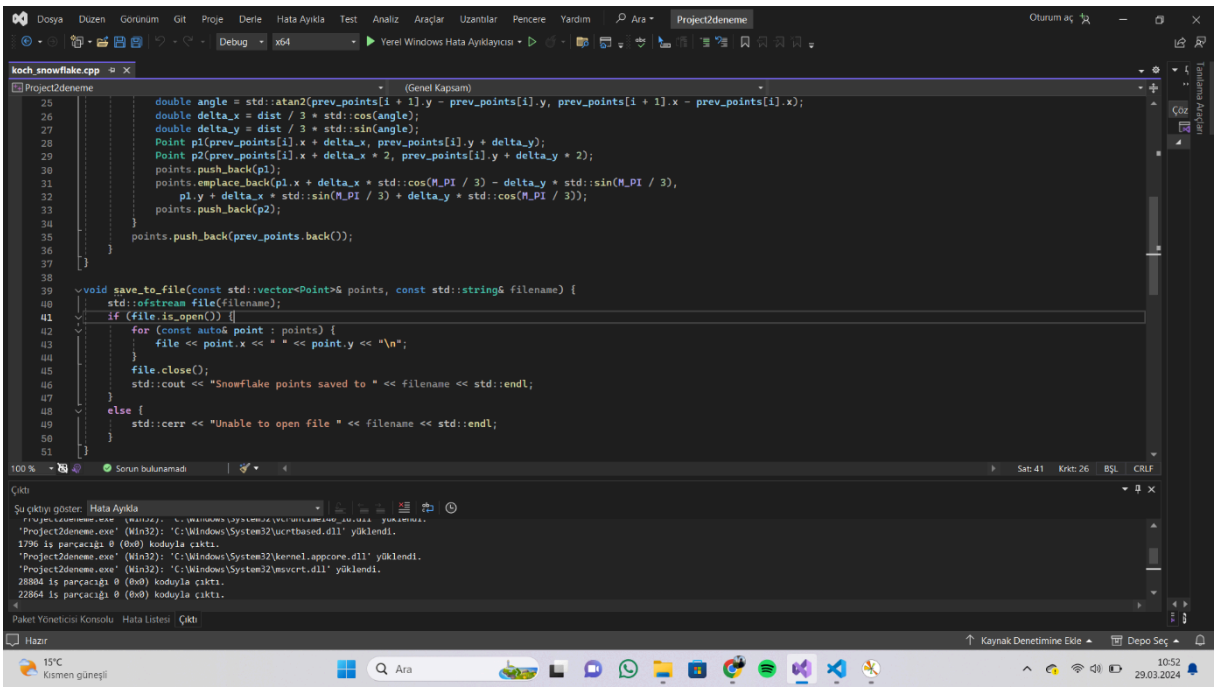
KULUBECİOGLU MEHMET

Kyiv– 2023



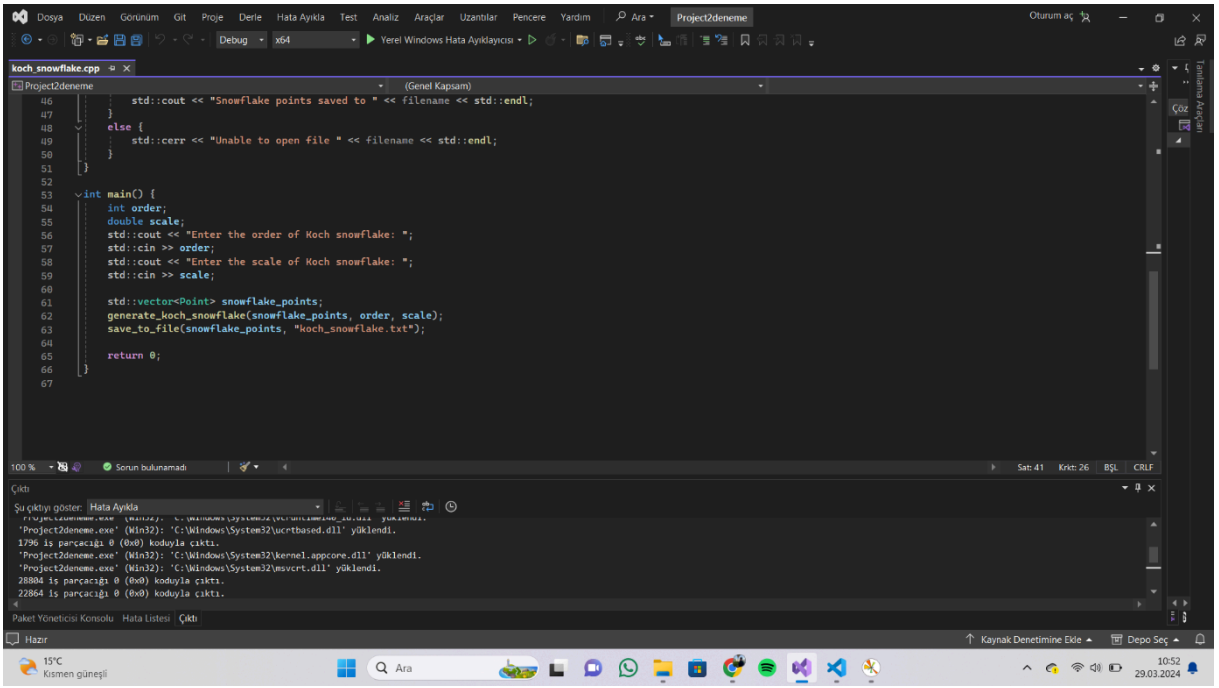
```
#define _USE_MATH_DEFINES
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>

struct Point {
    double x;
    double y;
    Point(double x, double y) : x(x), y(y) {}
};

void generate_koch_snowflake(std::vector<Point>& points, int order, double scale) {
    if (order == 0) {
        points.emplace_back(Point(0, 0));
        points.emplace_back(Point(scale / 2, scale * std::sqrt(3) / 2));
        points.emplace_back(Point(scale, 0));
    }
    else {
        std::vector<Point> prev_points;
        generate_koch_snowflake(prev_points, order - 1, scale);
        for (size_t i = 0; i < prev_points.size() - 1; ++i) {
            points.push_back(prev_points[i]);
            double dist = std::hypot(prev_points[i + 1].x - prev_points[i].x, prev_points[i + 1].y - prev_points[i].y);
            double angle = std::atan2(prev_points[i + 1].y - prev_points[i].y, prev_points[i + 1].x - prev_points[i].x);
            double delta_x = dist / 3 * std::cos(angle);
            double delta_y = dist / 3 * std::sin(angle);
```

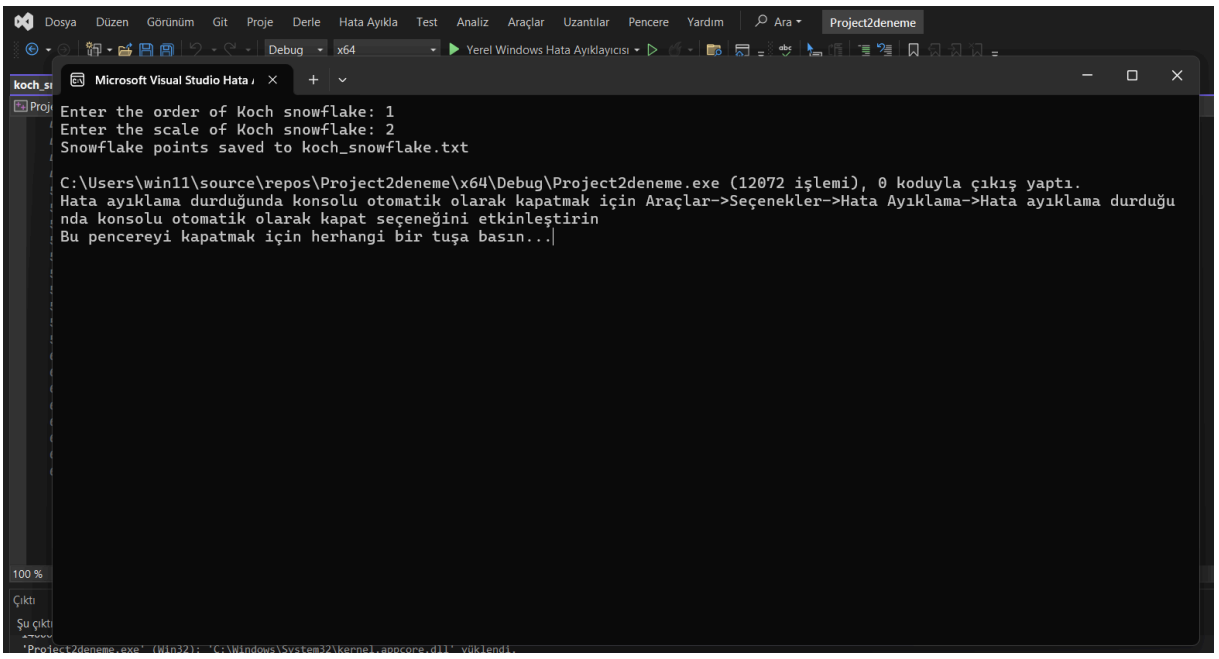
```
'Project2deneme.exe' (Win32): 'C:\Windows\System32\ucrtbased.dll' yüklendi.
1796 iş parçacığı 0 (0x0) koduyla çıktı.
'Project2deneme.exe' (Win32): 'C:\Windows\System32\kernel.appcore.dll' yüklendi.
'Project2deneme.exe' (Win32): 'C:\Windows\System32\msvcrt.dll' yüklendi.
28804 iş parçacığı 0 (0x0) koduyla çıktı.
22864 iş parçacığı 0 (0x0) koduyla çıktı.
```

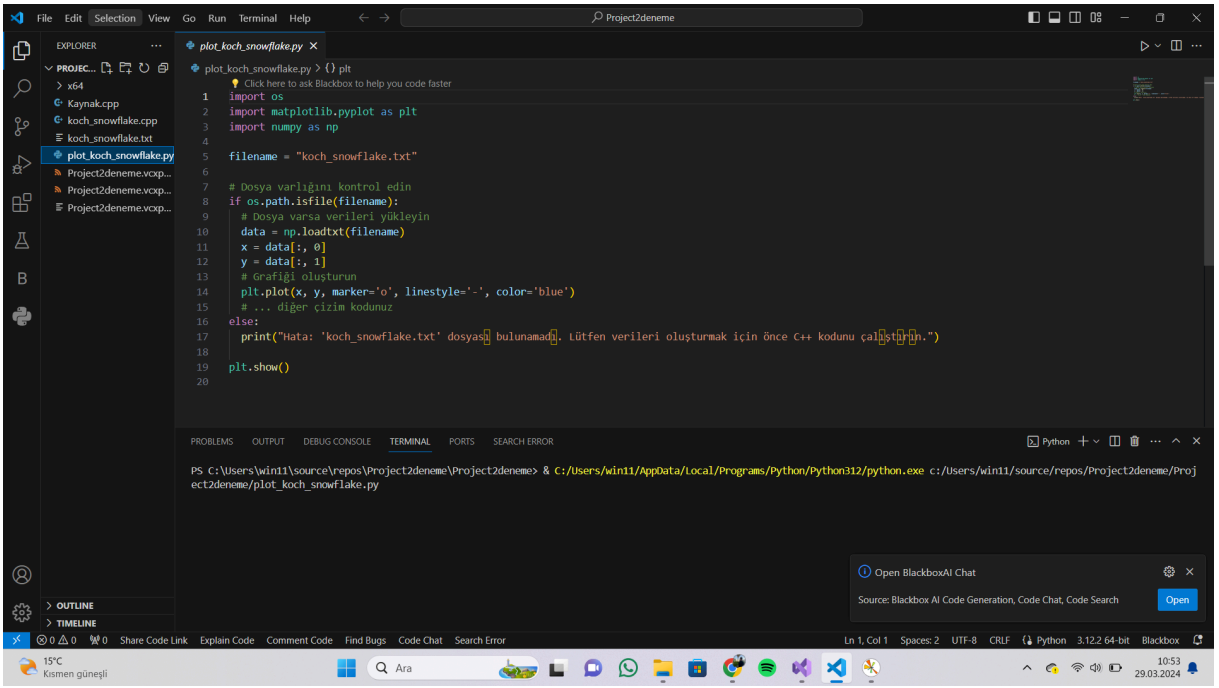


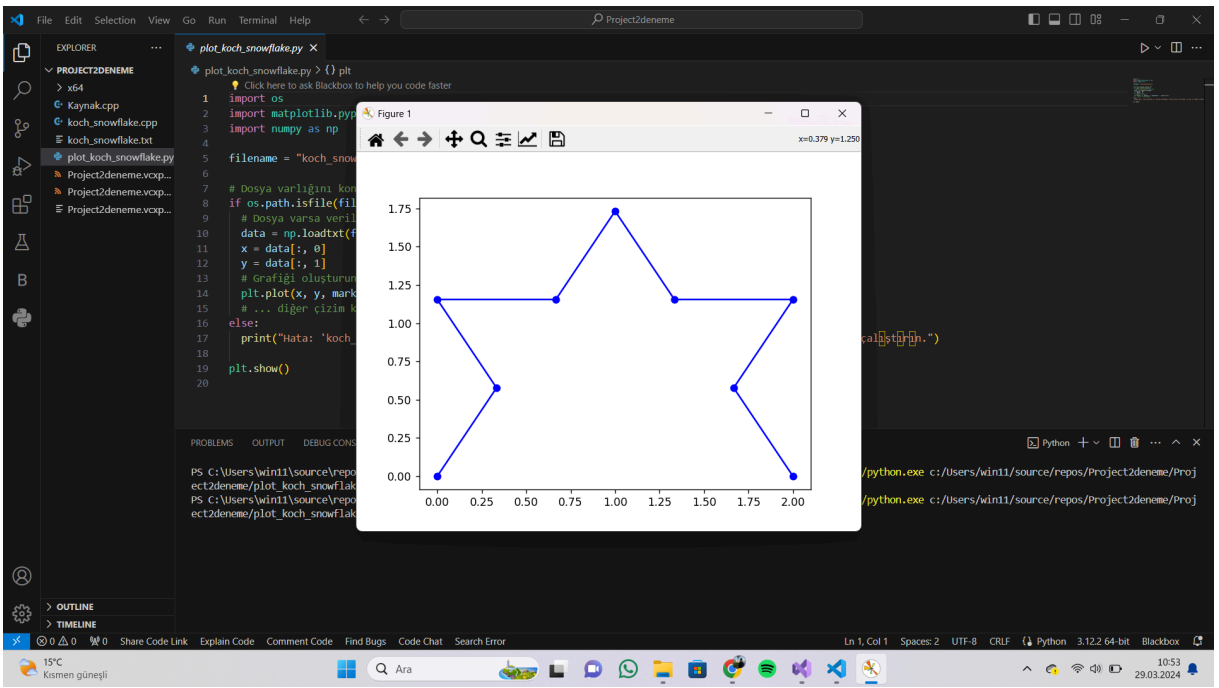
```
            double angle = std::atan2(prev_points[i + 1].y - prev_points[i].y, prev_points[i + 1].x - prev_points[i].x);
            double delta_x = dist / 3 * std::cos(angle);
            double delta_y = dist / 3 * std::sin(angle);
            Point p1(prev_points[i].x + delta_x, prev_points[i].y + delta_y);
            Point p2(prev_points[i].x + delta_x * 2, prev_points[i].y + delta_y * 2);
            points.push_back(p1);
            points.emplace_back(p1.x + delta_x * std::cos(M_PI / 3) - delta_y * std::sin(M_PI / 3),
                p1.y + delta_x * std::sin(M_PI / 3) + delta_y * std::cos(M_PI / 3));
            points.push_back(p2);
        }
        points.push_back(prev_points.back());
    }
}

void save_to_file(const std::vector<Point>& points, const std::string& filename) {
    std::ofstream file(filename);
    if (file.is_open()) {
        for (const auto& point : points) {
            file << point.x << " " << point.y << "\n";
        }
        file.close();
        std::cout << "Snowflake points saved to " << filename << std::endl;
    }
    else {
        std::cerr << "Unable to open file " << filename << std::endl;
    }
}
```

```
'Project2deneme.exe' (Win32): 'C:\Windows\System32\ucrtbased.dll' yüklendi.
1796 iş parçacığı 0 (0x0) koduyla çıktı.
'Project2deneme.exe' (Win32): 'C:\Windows\System32\kernel.appcore.dll' yüklendi.
'Project2deneme.exe' (Win32): 'C:\Windows\System32\msvcrt.dll' yüklendi.
28804 iş parçacığı 0 (0x0) koduyla çıktı.
22864 iş parçacığı 0 (0x0) koduyla çıktı.
```


koch_snowflake.cpp
Project2deneme
(Genel Kapsam)
std::cout << "Snowflake points saved to " << filename << std::endl;
std::cerr << "Unable to open file " << filename << std::endl;
int main() {
int order;
double scale;
std::cout << "Enter the order of Koch snowflake: ";
std::cin >> order;
std::cout << "Enter the scale of Koch snowflake: ";
std::cin >> scale;
std::vector<Point> snowflake_points;
generate_koch_snowflake(snowflake_points, order, scale);
save_to_file(snowflake_points, "koch_snowflake.txt");
return 0;
'Project2deneme.exe' (Win32): 'C:\Windows\System32\ucrtbased.dll' yüklendi.
1796 iş parçacığı 0 (0x0) koduyla çıktı.
'Project2deneme.exe' (Win32): 'C:\Windows\System32\kernel.appcore.dll' yüklendi.
'Project2deneme.exe' (Win32): 'C:\Windows\System32\msvcrt.dll' yüklendi.
28804 iş parçacığı 0 (0x0) koduyla çıktı.
22864 iş parçacığı 0 (0x0) koduyla çıktı.


Microsoft Visual Studio Hata
Enter the order of Koch snowflake: 1
Enter the scale of Koch snowflake: 2
Snowflake points saved to koch_snowflake.txt
C:\Users\win11\source\repos\Project2deneme\x64\Debug\Project2deneme.exe (12072 işlemi), 0 koduyla çıkış yaptı.
Hata ayıklama durduğunda konsolu otomatik olarak kapatmak için Araçlar->Seçenekler->Hata Ayıklama->Hata ayıklama durduğu
nda konsolu otomatik olarak kapat seçeneğini etkinleştirin
Bu pencereyi kapatmak için herhangi bir tuşa basın...





1. Introduction This report elucidates the process of generating a Koch Snowflake using both C++ and Python programming languages. It outlines the

creation of the snowflake's vertices, saving the data into a file, and subsequently visualizing it using Python.

2. C++ Code Overview

Koch Snowflake Generation: The C++ code contains a function responsible for computing the vertices of the Koch Snowflake. This function calculates the x and y coordinates of each vertex and stores them in a vector.

Saving Data to File: The save_to_file function is utilized to store the computed Koch Snowflake vertices into a file. This facilitates transferring the data to Python for further processing.

3. Python Code Overview

File Existence Check: The Python code checks for the existence of a specific file. If the file exists, it loads the data and proceeds with the visualization process. Otherwise, it displays an error message.

Loading Data: Existing data from the file is loaded using the np.loadtxt() function. This step enables the transfer of the Koch Snowflake vertex data to the Python environment.

Visualization: After data loading, a line plot is created using the matplotlib library. This plot represents the vertices of the Koch Snowflake, with the x and y axes corresponding to the coordinates of the vertices.

4. Sample Outputs

C++ Output: The Koch Snowflake, generated based on user-provided order and scale values, is saved into a file, such as koch_snowflake.txt.

Python Output: Data from the existing file is loaded, and a graphical representation of the Koch Snowflake is generated. This visualization aids in understanding the geometrical structure of the snowflake.

5. Conclusion This report demonstrates the collaborative use of C++ and Python programming languages for generating and visualizing the Koch Snowflake. The interaction between these languages exemplifies how diverse programming languages can be combined in projects, showcasing their complementary roles in handling different aspects of a task.