TECHNICAL UNIVERSITY OF TALLINN

Faculty of Engineering

Department of Computer Systems

IAX0584 Programming II

C++

Homework 3

Author: Marietta Madisson

Group: MVEB21

Supervisor: Vladimir Viies

2025 Tallinn

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. All works and major viewpoints of the other authors, data from other sources of literature and elsewhere used for writing this paper have been referenced.

Name: Marietta Madisson

Date: 23.04.2025

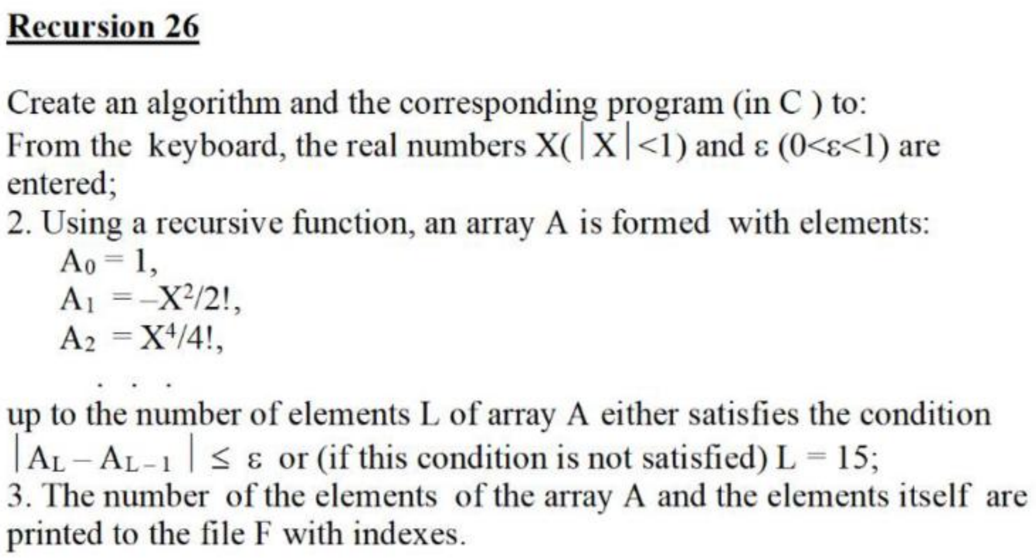
Contents

Task

In Homework 3, the re-implementation of either Homework 1 or Homework 2 is required, with a specific focus on utilizing C++ and object-oriented programming (OOP) principles. Instead of a procedural approach, the solution should be structured around the design and implementation of C++ classes. The key objective is to demonstrate the ability to effectively apply C++ class concepts (including encapsulation, inheritance, and polymorphism) and OOP design techniques to structure the solution.

I've chosen to do Homework 2 for this task.

Task description:



Recursion 26

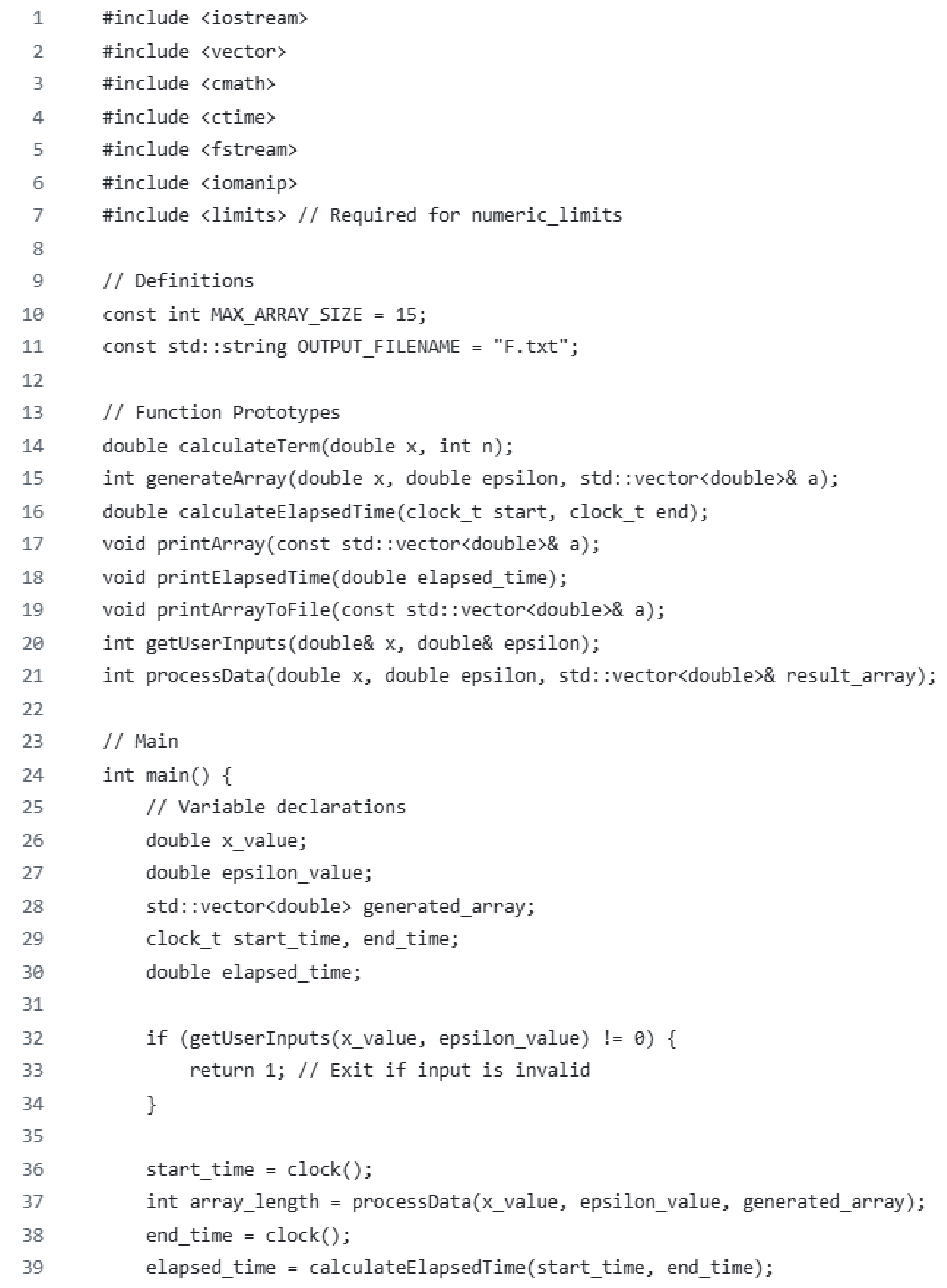
Create an algorithm and the corresponding program (in C) to:
From the keyboard, the real numbers X($|X|<1$) and ε ($0<\varepsilon<1$) are entered;
2. Using a recursive function, an array A is formed with elements:

$A_0 = 1$,
$A_1 = -X^2/2!$,
$A_2 = X^4/4!$,
. . .

up to the number of elements L of array A either satisfies the condition $|A_L - A_{L-1}| \leq \varepsilon$ or (if this condition is not satisfied) L = 15;
3. The number of the elements of the array A and the elements itself are printed to the file F with indexes.

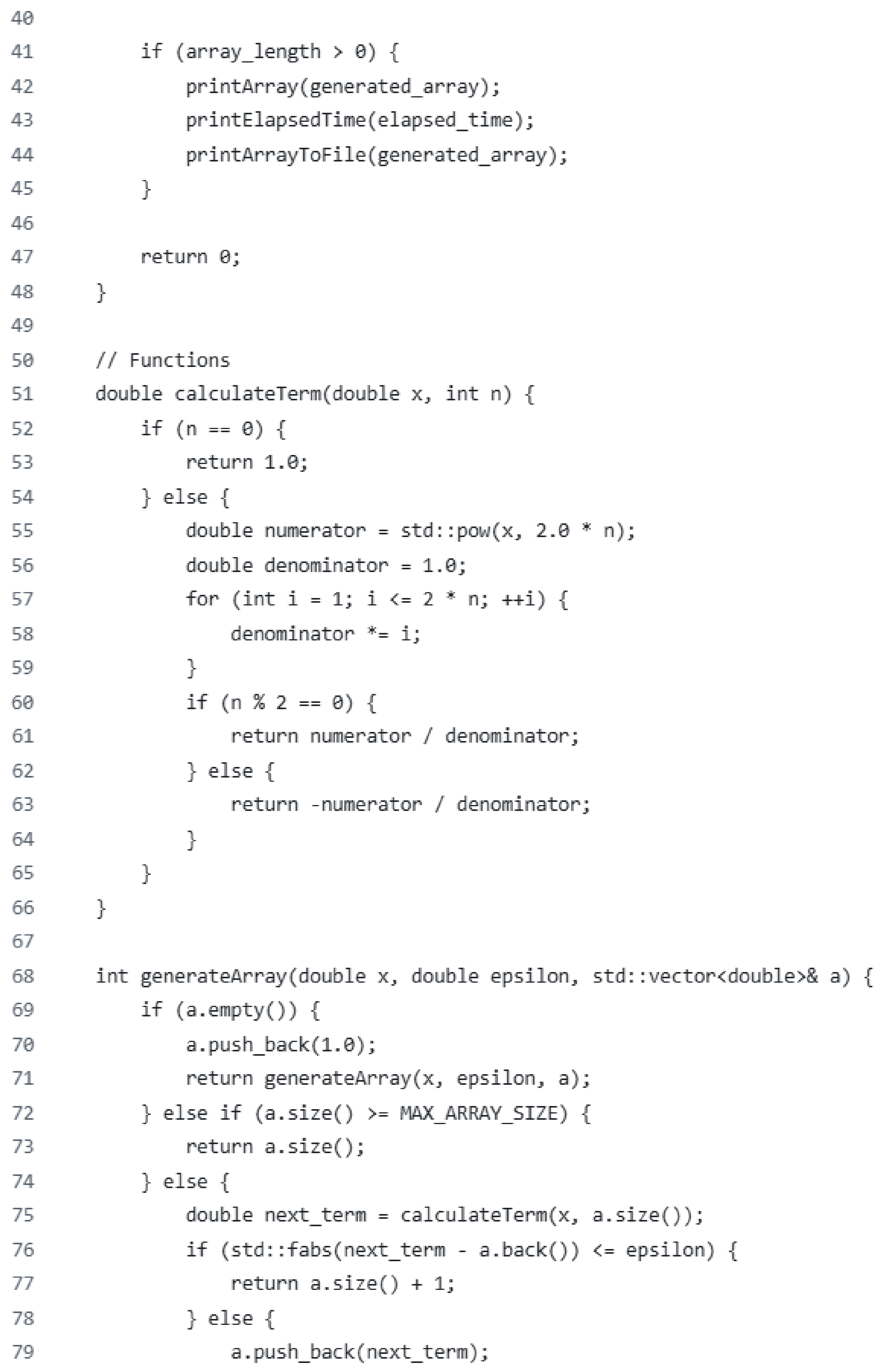
Picture 1. Task

Code in C++



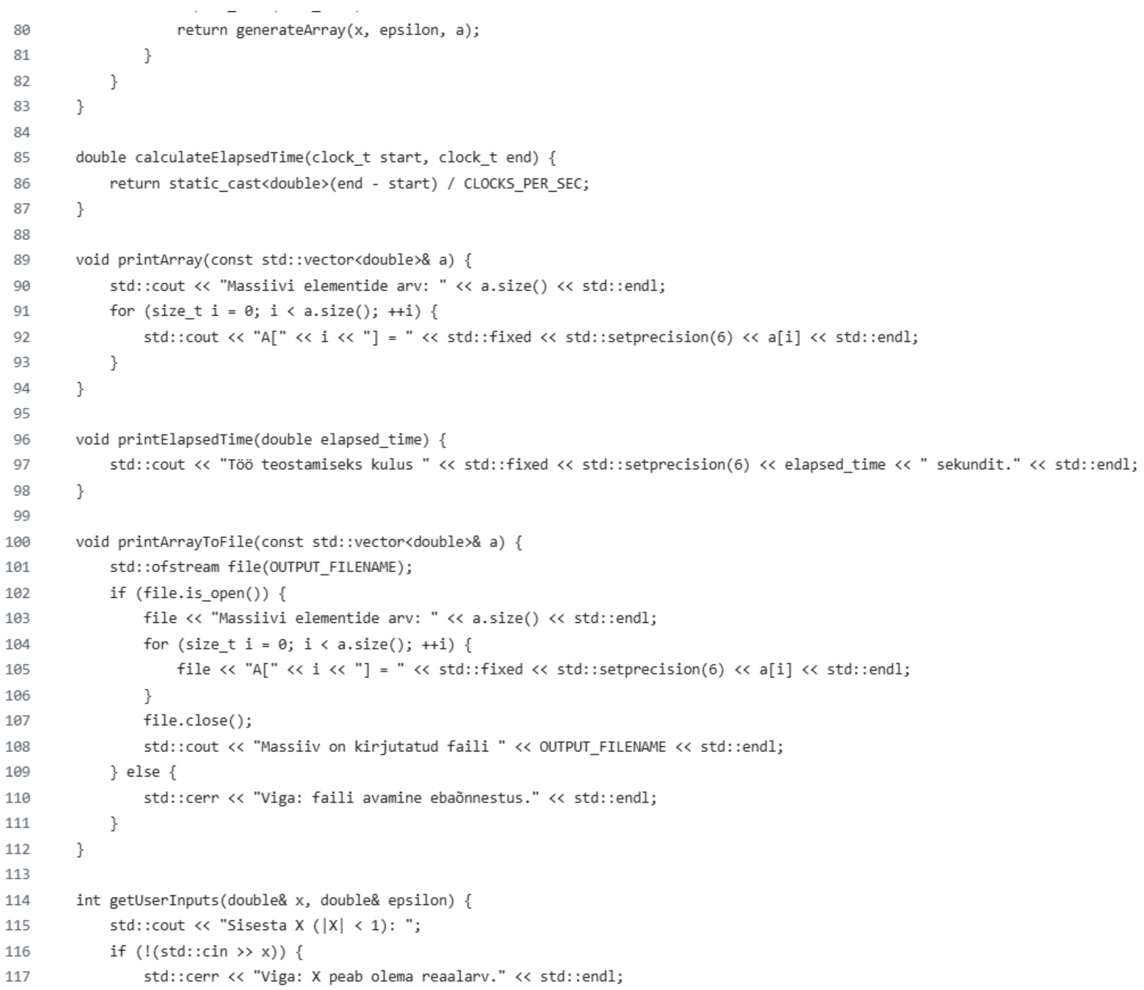
```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <ctime>
#include <fstream>
#include <iomanip>
#include <limits> // Required for numeric_limits

// Definitions
const int MAX_ARRAY_SIZE = 15;
const std::string OUTPUT_FILENAME = "F.txt";

// Function Prototypes
double calculateTerm(double x, int n);
int generateArray(double x, double epsilon, std::vector<double>& a);
double calculateElapsedTime(clock_t start, clock_t end);
void printArray(const std::vector<double>& a);
void printElapsedTime(double elapsed_time);
void printArrayToFile(const std::vector<double>& a);
int getUserInputs(double& x, double& epsilon);
int processData(double x, double epsilon, std::vector<double>& result_array);

// Main
int main() {
    // Variable declarations
    double x_value;
    double epsilon_value;
    std::vector<double> generated_array;
    clock_t start_time, end_time;
    double elapsed_time;

    if (getUserInputs(x_value, epsilon_value) != 0) {
        return 1; // Exit if input is invalid
    }

    start_time = clock();
    int array_length = processData(x_value, epsilon_value, generated_array);
    end_time = clock();
    elapsed_time = calculateElapsedTime(start_time, end_time);
```

Picture 2. Code part 1



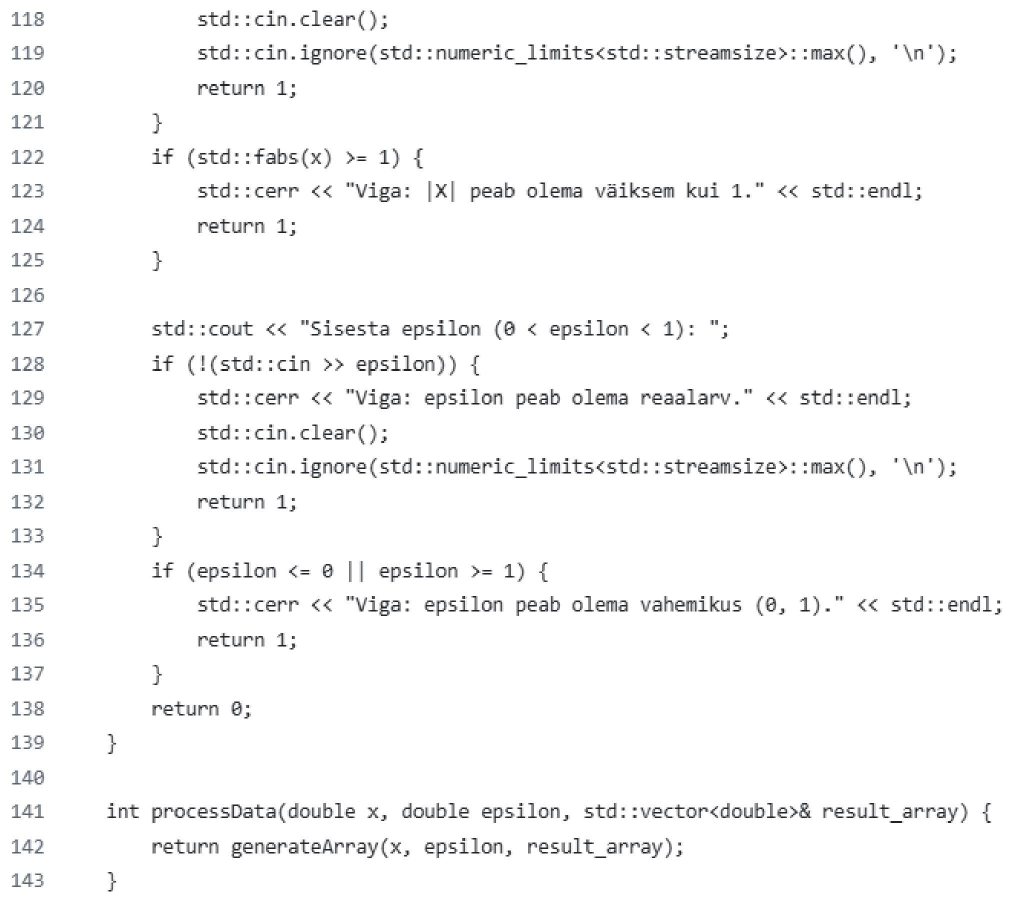
```cpp

    if (array_length > 0) {
        printArray(generated_array);
        printElapsedTime(elapsed_time);
        printArrayToFile(generated_array);
    }

    return 0;
}

// Functions
double calculateTerm(double x, int n) {
    if (n == 0) {
        return 1.0;
    } else {
        double numerator = std::pow(x, 2.0 * n);
        double denominator = 1.0;
        for (int i = 1; i <= 2 * n; ++i) {
            denominator *= i;
        }
        if (n % 2 == 0) {
            return numerator / denominator;
        } else {
            return -numerator / denominator;
        }
    }
}

int generateArray(double x, double epsilon, std::vector<double>& a) {
    if (a.empty()) {
        a.push_back(1.0);
        return generateArray(x, epsilon, a);
    } else if (a.size() >= MAX_ARRAY_SIZE) {
        return a.size();
    } else {
        double next_term = calculateTerm(x, a.size());
        if (std::fabs(next_term - a.back()) <= epsilon) {
            return a.size() + 1;
        } else {
            a.push_back(next_term);
```

Picture 3. Code part 2



```
            return generateArray(x, epsilon, a);
        }
    }
}

double calculateElapsedTime(clock_t start, clock_t end) {
    return static_cast<double>(end - start) / CLOCKS_PER_SEC;
}

void printArray(const std::vector<double>& a) {
    std::cout << "Massiivi elementide arv: " << a.size() << std::endl;
    for (size_t i = 0; i < a.size(); ++i) {
        std::cout << "A[" << i << "] = " << std::fixed << std::setprecision(6) << a[i] << std::endl;
    }
}

void printElapsedTime(double elapsed_time) {
    std::cout << "Töö teostamiseks kulus " << std::fixed << std::setprecision(6) << elapsed_time << " sekundit." << std::endl;
}

void printArrayToFile(const std::vector<double>& a) {
    std::ofstream file(OUTPUT_FILENAME);
    if (file.is_open()) {
        file << "Massiivi elementide arv: " << a.size() << std::endl;
        for (size_t i = 0; i < a.size(); ++i) {
            file << "A[" << i << "] = " << std::fixed << std::setprecision(6) << a[i] << std::endl;
        }
        file.close();
        std::cout << "Massiiv on kirjutatud faili " << OUTPUT_FILENAME << std::endl;
    } else {
        std::cerr << "Viga: faili avamine ebaõnnestus." << std::endl;
    }
}

int getUserInputs(double& x, double& epsilon) {
    std::cout << "Sisesta X (|X| < 1): ";
    if (!(std::cin >> x)) {
        std::cerr << "Viga: X peab olema reaalarv." << std::endl;
```

Picture 4. Code part 3



```cpp
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        return 1;
    }
    if (std::fabs(x) >= 1) {
        std::cerr << "Viga: |X| peab olema väiksem kui 1." << std::endl;
        return 1;
    }

    std::cout << "Sisesta epsilon (0 < epsilon < 1): ";
    if (!(std::cin >> epsilon)) {
        std::cerr << "Viga: epsilon peab olema reaalarv." << std::endl;
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        return 1;
    }
    if (epsilon <= 0 || epsilon >= 1) {
        std::cerr << "Viga: epsilon peab olema vahemikus (0, 1)." << std::endl;
        return 1;
    }
    return 0;
}

int processData(double x, double epsilon, std::vector<double>& result_array) {
    return generateArray(x, epsilon, result_array);
}
```

Picture 5. Code part 4

AI contribution

The C++ source code was produced through the application of an AI-powered code generation tool. Input to the AI consisted of a comprehensive task description and a preliminary version of the code written in the C programming language. I wrote as much as I knew the code into C++ and then gave all the information I had to AI. The AI subsequently processed this information to create the final C++ implementation, adhering to the defined objectives.