TECHNICAL UNIVERSITY OF TALLINN

Faculty of Engineering

Department of Computer Systems

IAX0584 Programming II

Recursion

Homework II

Author: Marietta Madisson

Student code: 243484MVEB

Group: MVEB21

Supervisor: Vladimir Viies

2025 Tallinn

Table of Contents

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: 16.04.2025

1. Assignment

1.1 Generating task

The task was assigned based on the last digit of the student's date of birth. Since mine is 6, I was given the following task:

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 $|A_L - A_{L-1}| \le \varepsilon$ 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. Screenshot of task

1.2 Understanding task

The given task requires developing a C program that generates a mathematical sequence using recursion and writes the results to a file. The key functional requirements of the program include:

Reading User Input

- The program prompts the user to input a real number X, ensuring that |X|<1|.
- The user must also provide a small positive value ϵ\epsilonϵ, ensuring that 0<ϵ<1.
- If the input values do not meet these conditions, the program displays an error message and exits.

Generating the Sequence Recursively

- The sequence follows the pattern:
  $A_0 = 1,$
  $A_1 = -X^2/2!$
  $A_2 = X^4/4!,$
  . . .
- A recursive approach is used to construct the sequence until one of the following conditions is met:
  - The absolute difference between two consecutive terms satisfies $|A_L - A_{L-1}| \leq \varepsilon$
  - The maximum number of terms (15) is reached if the first condition is not met.

Writing and Displaying the Results

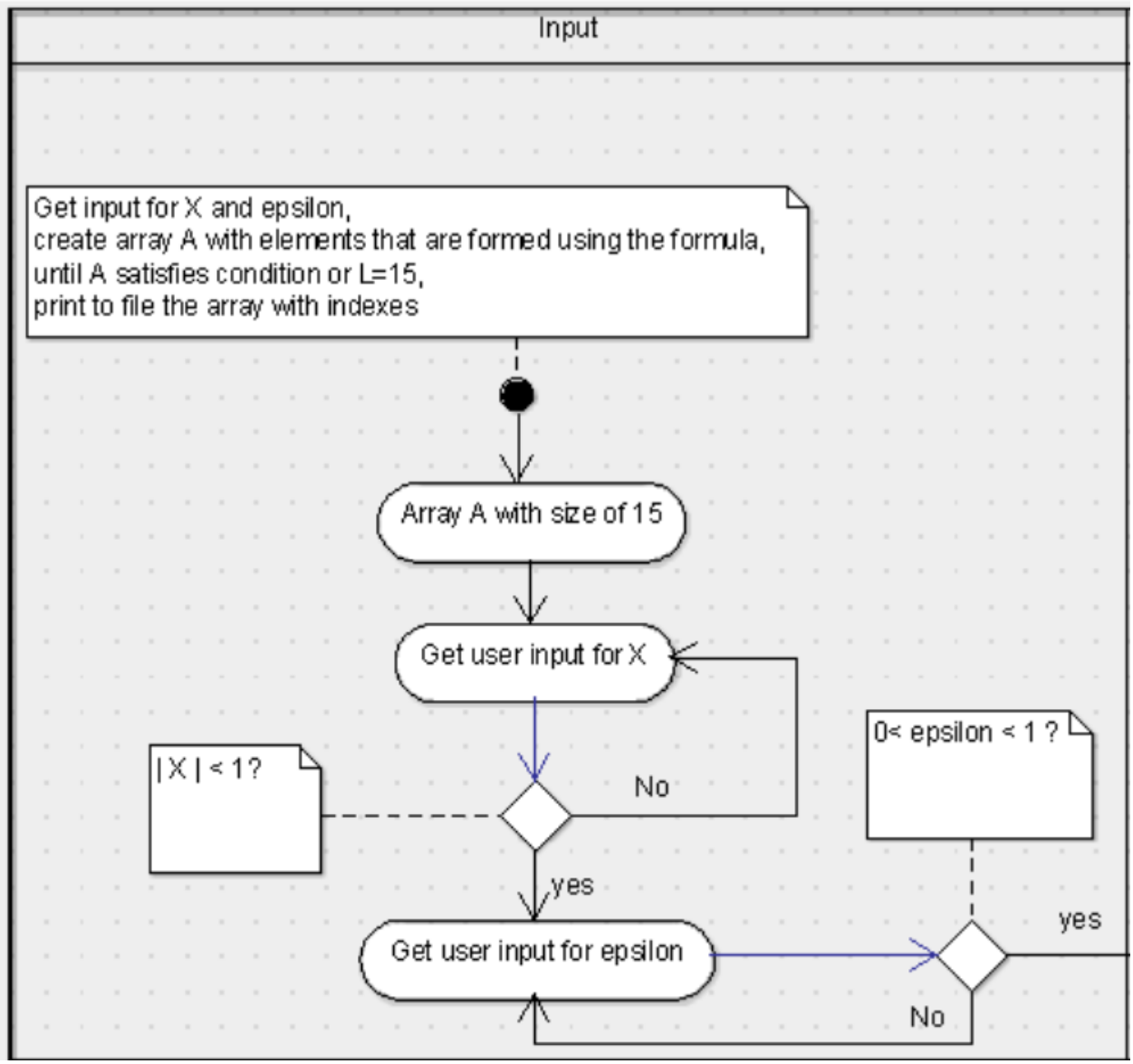
- The program prints the generated sequence with indices.
- The sequence is also written to a file named "F.txt" for future reference.
- The execution time of the program is measured and displayed.

Program Structure

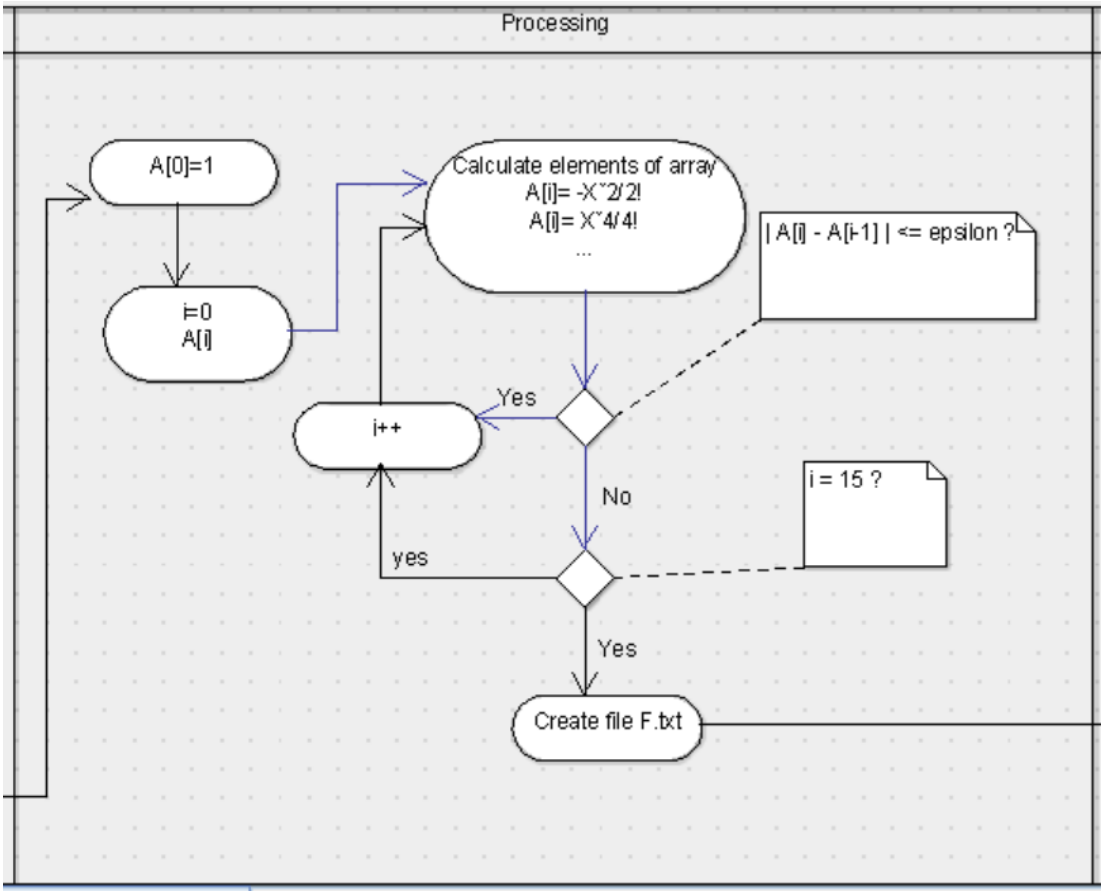
- The program is structured using multiple functions:
- `calculateTerm`() computes individual terms of the sequence.
- `generateArray`() recursively builds the sequence.
- `printArray`() displays the sequence in the console.
- `printArrayToFile()` writes the results to "F.txt".
- `calculateElapsedTime`() measures execution time.
- Error-handling mechanisms ensure valid input and correct file operations.

2. Program description

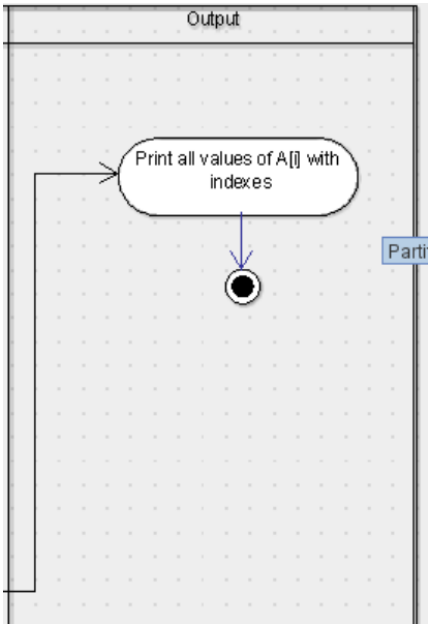
2.1 UML-chart



Picture 1. Input



Picture 2. Processing



Picture 3. Output

2.2 Code in C

```c
// Library
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

// Definitions
#define MAX_ARRAY_SIZE 15
#define OUTPUT_FILENAME "F.txt"

// Prototypes
double calculateTerm(double x, int n);
int generateArray(double x, double epsilon, double a[], int index);
double calculateElapsedTime(clock_t start, clock_t end);
void printArray(double a[], int length);
void printElapsedTime(double elapsed_time);
void printArrayToFile(double a[], int length);

// Main
int main() {
    // Variable declarations
    double x, epsilon;
    double a[MAX_ARRAY_SIZE];
    int length;
    clock_t start_time, end_time;
    double elapsed_time;

    // Calls to functions
    printf("Sisesta X (|X| < 1): ");
    scanf("%lf", &x);
    if (fabs(x) >= 1) {
        printf("Viga: |X| peab olema väiksem kui 1.\n");
        return 1;
    }

    printf("Sisesta epsilon (0 < epsilon < 1): ");
    scanf("%lf", &epsilon);
    if (epsilon <= 0 || epsilon >= 1) {
        printf("Viga: epsilon peab olema vahemikus (0, 1).\n");
        return 1;
    }
```

Picture 4. Code part 1

```
41          }
42
43          start_time = clock();
44          length = generateArray(x, epsilon, a, 0);
45          end_time = clock();
46          elapsed_time = calculateElapsedTime(start_time, end_time);
47
48          printArray(a, length);
49          printElapsedTime(elapsed_time);
50          printArrayToFile(a, length);
51
52          return 0;
53      }
54
55      // Functions
56      double calculateTerm(double x, int n) {
57          if (n == 0) {
58              return 1.0;
59          } else {
60              double numerator = 1.0;
61              for (int i = 0; i < 2 * n; i++) {
62                  numerator *= x;
63              }
64              double denominator = 1.0;
65              for (int i = 1; i <= 2 * n; i++) {
66                  denominator *= i;
67              }
68              if (n % 2 == 0) {
69                  return numerator / denominator;
70              } else {
71                  return -numerator / denominator;
72              }
73          }
74      }
75
76      int generateArray(double x, double epsilon, double a[], int index) {
77          if (index == 0) {
78              a[0] = 1.0;
79              return generateArray(x, epsilon, a, index + 1);
80          } else if (index >= MAX_ARRAY_SIZE) {
```

Picture 5. Code part 2

```
            return index;
        } else {
            a[index] = calculateTerm(x, index);
            if (fabs(a[index] - a[index - 1]) <= epsilon) {
                return index + 1;
            } else {
                return generateArray(x, epsilon, a, index + 1);
            }
        }
    }

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

    void printArray(double a[], int length) {
        printf("Massiivi elementide arv: %d\n", length);
        for (int i = 0; i < length; i++) {
            printf("A[%d] = %lf\n", i, a[i]);
        }
    }

    void printElapsedTime(double elapsed_time) {
        printf("Töö teostamiseks kulus %lf sekundit.\n", elapsed_time);
    }

    void printArrayToFile(double a[], int length) {
        FILE *file = fopen(OUTPUT_FILENAME, "w");
        if (file == NULL) {
            printf("Viga: faili avamine ebaõnnestus.\n");
            return;
        }

        fprintf(file, "Massiivi elementide arv: %d\n", length);
        for (int i = 0; i < length; i++) {
            fprintf(file, "A[%d] = %lf\n", i, a[i]);
        }

        fclose(file);
        printf("Massiiv on kirjutatud faili %s\n", OUTPUT_FILENAME);
```

Picture 6. Code part 3

3. Output of program

```
Sisesta X (|X| < 1): 0.9
Sisesta epsilon (0 < epsilon < 1): 0.0001
Massiivi elementide arv: 6
A[0] = 1.000000
A[1] = -0.405000
A[2] = 0.027338
A[3] = -0.000738
A[4] = 0.000011
A[5] = -0.000000
Töö teostamiseks kulus 0.000011 sekundit.
Massiiv on kirjutatud faili F.txt


------------------
(program exited with code: 0)
Press return to continue
```

Picture 7. Screenshot of output