TECHNICAL UNIVERSITY OF TALLINN

Faculty of Engineering

Department of Computer Systems

IAX0584 Programming II

Files and structures

Homework I

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: 12.03.2025

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:

Variant K-6:

Write an algorithm and matching code for a program with the following functional requirements:

1. Data is read from a plain text file „F1.txt“ and stored as a structure. The data file must contain the given attributes:

· Name - string

· Student code - string [10]

· Grade - array of integers

2. Students whose grades are all 5, will be assigned 100 € scholarship.

3. Students whose grades are either 5 or 4 (but not all 5), will be assigned 75 € scholarship.

4. Program will output to file „F2.txt“ the students who receive the scholarship in descending order of the sum. When the sum of the scholarship is equal, then in alphabetical order.

1.2 Understanding task

The given task requires developing a program that processes student data from a text file and determines scholarship eligibility based on their grades. The key functional requirements of the program include:

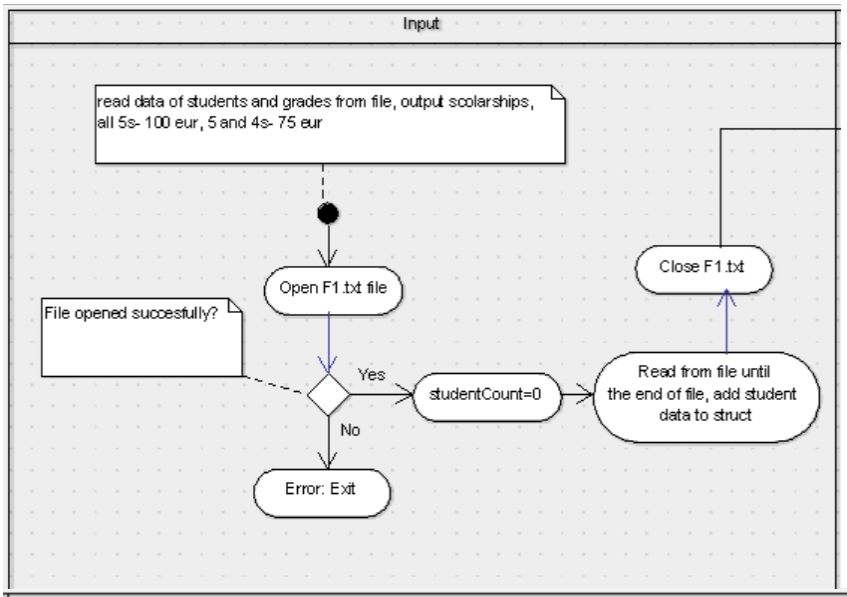
1. **Reading Data from a File**
    a. The program must read student information from a file named "F1.txt."

   b. Each student entry includes a name (string), student code (string of length 10), and an array of integer grades.
   c. The data must be stored in an appropriate structure to facilitate processing.

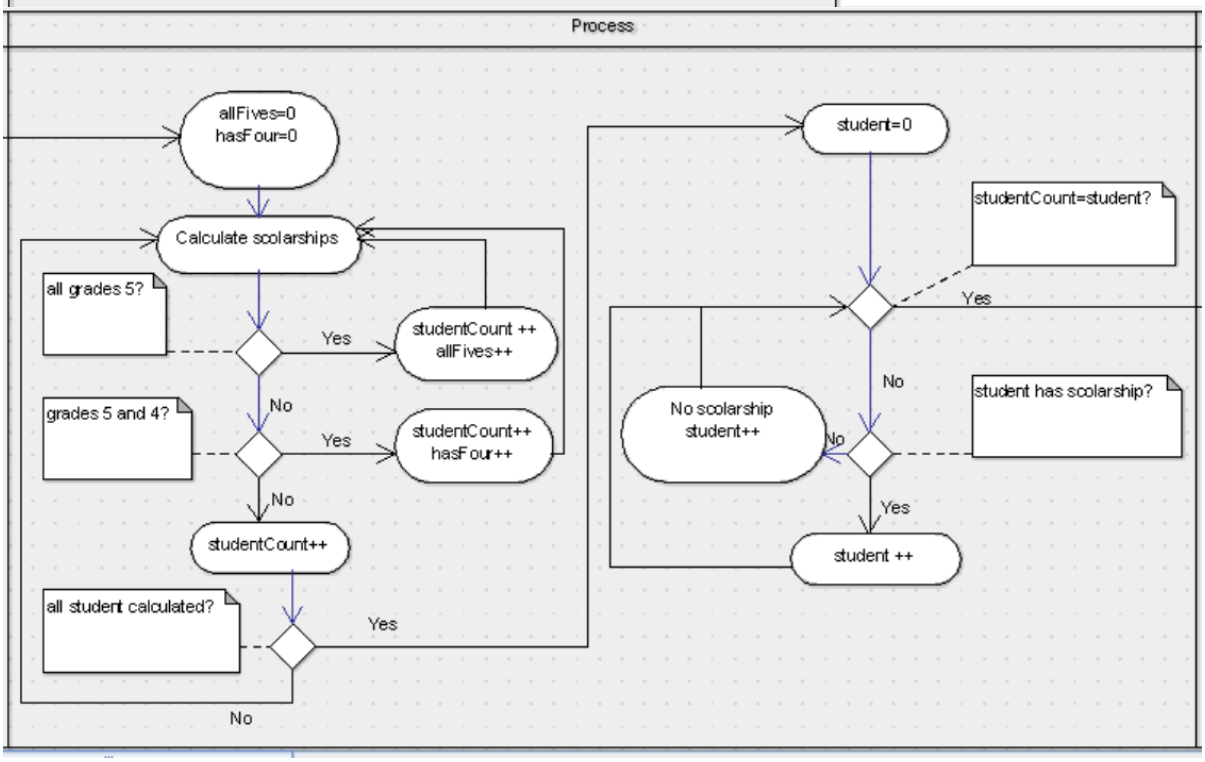
2. **Determining Scholarship Eligibility**
   a. Students who have all grades equal to 5 will receive a scholarship of 100 €.
   b. Students whose grades consist only of 4s and 5s (but not exclusively 5s) will receive a scholarship of 75 €.
   c. Students with grades lower than 4 will not receive any scholarship.
3. **Sorting and Writing Data to a File**
   a. The students who receive scholarships must be written to a file named "F2.txt."
   b. The list should be sorted in descending order of scholarship amount.
   c. If two students have the same scholarship amount, they should be sorted alphabetically by name.
4. **Program Structure**
   a. The program defines a structure (Student) to store the necessary student attributes.
   b. Functions are implemented to read from the input file, process scholarship eligibility, sort the students, and write the results to the output file.
   c. An error-handling mechanism ensures proper file handling and program execution.
5. **Output and Display**
   a. The program also displays the number of students who received scholarships.
   b. If any errors occur, they are logged into an error file and displayed on the console.

2. Program description

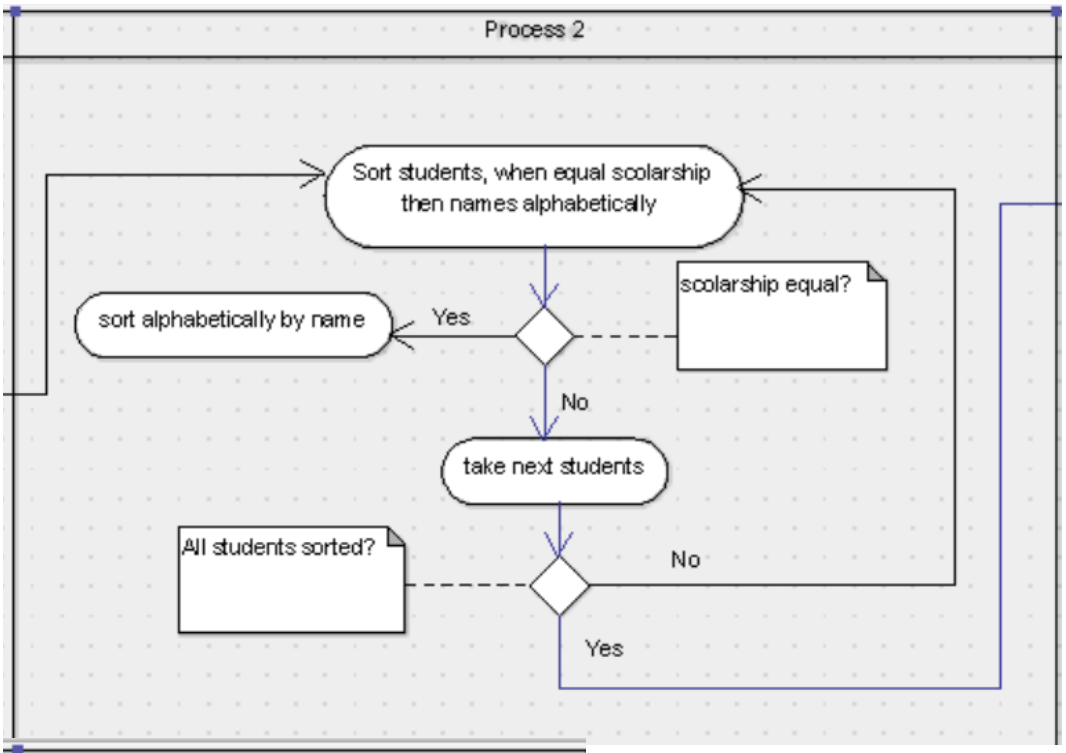
2.1 UML-chart



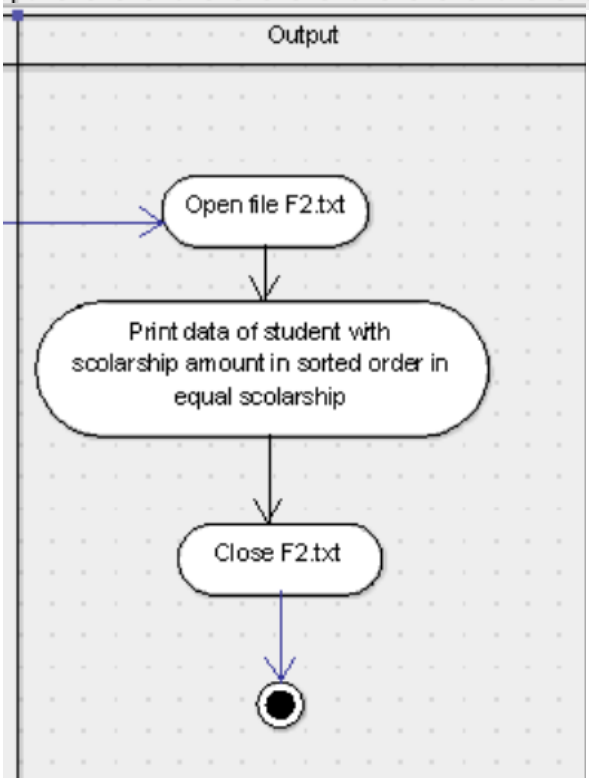
Picture 1. UML input



Picture 2. UML process part 1



Picture 3. UML process part 2



Picture 4. UML output

2.2 Code in C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100
#define MAX_NAME 50
#define MAX_GRADES 10

typedef struct {
    char name[MAX_NAME];
    char student_code[10];
    int grades[MAX_GRADES];
    int grade_count;
    int scholarship;
} Student;

void readFile1(const char *filename, Student *students, int *count);
void processStudents(Student *students, int count);
void writeFile2(const char *filename, Student *students, int count);
void displayScholarshipCount(Student *students, int count);
void Error(const char *message);
int compareStudents(const void *a, const void *b);

int main()
{
    Student students[MAX];
    int count = 0;

    readFile1("F1.txt", students, &count);
    processStudents(students, count);
    writeFile2("scholarship.txt", students, count);
    displayScholarshipCount(students, count);

    return 0;
}

```

Picture 5. Code part 1

```c
void readFile1(const char *filename, Student *students, int *count)
{
    FILE *file1 = fopen("F1.txt", "r");
    if (file1 == NULL)
    {
        Error("Error opening file F1.txt");
    }

    while (fscanf(file1, "%49s %9s", students[*count].name, students[*count].student_code) == 2)
    {
        students[*count].grade_count = 0;

        while (students[*count].grade_count < MAX_GRADES && fscanf(file1, "%d", &students[*count].grades[students[*count].grade_count]) == 1)
        {
            students[*count].grade_count++;
        }

        (*count)++;
    }

    fclose(file1);
}
```

Picture 6. Code part 2

```c

void processStudents(Student *students, int count)
{
        int i;
        int j;

    for (i = 0; i < count; i++)
    {
        int all_five = 1, only_four_or_five = 1;
        for (j = 0; j < students[i].grade_count; j++)
        {
            if (students[i].grades[j] != 5) all_five = 0;
            if (students[i].grades[j] < 4)
            {
                only_four_or_five = 0;
                break;
            }
        }
        if (all_five)
            students[i].scholarship = 100;
        else if (only_four_or_five)
            students[i].scholarship = 75;
        else
            students[i].scholarship = 0;
    }
}

int compareStudents(const void *a, const void *b)
{
    Student *s1 = (Student *)a;
    Student *s2 = (Student *)b;

    if (s1->scholarship != s2->scholarship)
        return s2->scholarship - s1->scholarship;

    return strcmp(s1->name, s2->name);
```

Picture 7. Code part 3

```
}

void writeFile2(const char *filename, Student *students, int count)
{
    FILE *file2 = fopen("scholarship.txt", "w");
    if (!file2)
    {
        Error("Error opening file scholarship.txt");
    }

    qsort(students, count, sizeof(Student), compareStudents);

        int i;

    for (i = 0; i < count; i++)
    {
        fprintf(file2, "%s %s ", students[i].name, students[i].student_code);

        if (students[i].scholarship > 0)
        {
            fprintf(file2, "Scholarship: %d\n", students[i].scholarship);
        }
        else
        {
            fprintf(file2, "No scholarship\n");
        }
    }

    fclose(file2);
}

void displayScholarshipCount(Student *students, int count)
{
        int i;
    int scholarshipCount = 0;

```

Picture 8. Code part 4

```

    for (i = 0; i < count; i++)
    {
        if (students[i].scholarship > 0)
        {
            scholarshipCount++;
        }
    }

    printf("%d students received a scholarship.\n", scholarshipCount);
}

void Error(const char *message)
{
    FILE *errorFile = fopen("error.txt", "w");
    if (errorFile)
    {
        fprintf(errorFile, "%s\n", message);
        fclose(errorFile);
    }
    printf("%s\n", message);
    exit(1);
}

```

Picture 9. Code part 5

3. Output of program

Text file for process:

Märten 20230001 5 5 5 5 5

Kadri 20230002 4 5 5 5 4

Taavi 20230003 3 4 5 5 4

Liis 20230004 5 5 5 5 5

Rasmus 20230005 4 4 4 5 5

Anneli 20230006 5 4 5 5 4

Karl 20230007 3 3 5 4 4

Helena 20230008 5 5 5 5 5

Ott 20230009 5 5 4 4 5

Eerik 20230010 2 3 4 5 4

```
Helena 20230008 Scholarship: 100
Liis 20230004 Scholarship: 100
Märten 20230001 Scholarship: 100
Anneli 20230006 Scholarship: 75
Kadri 20230002 Scholarship: 75
Ott 20230009 Scholarship: 75
Rasmus 20230005 Scholarship: 75
Eerik 20230010 No scholarship
Karl 20230007 No scholarship
Taavi 20230003 No scholarship
```

Picture 10. Program output from created file F2.txt