Tallinn University of Technology

School of Engineering

Integrated Engineering

Emma Hansson

243487MVEB

**FILES AND STRUCTURES**

First homework in course IAX0584

Supervisor: Vladimir Viies

Tallinn 2025

# AUTHORS DECLARATION

I confirm that I have prepared this homework independently and that it has not been previously submitted for defense by someone else. All the works of other authors used in the preparation of the work, important points of view, data from literary sources and elsewhere are cited in the work.

Author: Emma Hansson

Date: 12.03.2025

**CONTENTS**

[AUTHORS DECLARATION 2](#_Toc192724188)

[1. TASK STATEMENT 4](#_Toc192724189)

[1.1. Specific task 4](#_Toc192724190)

[Specific task is option K-1: 4](#_Toc192724191)

[1.2. Workflow 4](#_Toc192724192)

[1.3. The algorithm 5](#_Toc192724193)

[1.1.1. File content: 6](#_Toc192724194)

[2. THE CODE IN C 7](#_Toc192724195)

[REFERENCES 10](#_Toc192724196)

# TASK STATEMENT

The task is to construct the algorithm of the task and the corresponding program in C language. All raw data is read from .txt file and user-entered from the keyboard. The results are displayed on the screen.

## Specific task

# Specific task is option K-1:

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 „Student“. The data file must contain the given attributes:
   * Student code - *integer or string*
   * Name - *string*
   * ID code - *string*
2. From another file „F2.txt“ another structure „Resident“ will be read, with the attributes
   * ID code - *string*
   * City – *string*
3. User will be asked to input a string „Residence“ from keyboard.
4. Program outputs all students with the given residence.

## Workflow

Simple description of the workflow of programme:

1. **Data Ingestion:** It reads student records from "F1.txt" and resident records from "F2.txt", storing them in memory.
2. **User Input:** It prompts the user to enter a city of residence.
3. **Relational Query:** It executes a join-like operation, matching student IDs with resident IDs, and filtering by the user-provided city.
4. **Result Presentation:** It displays the student codes, names, and IDs of students matching the search criteria.

## The algorithm

Based on the workflow, I conducted an algorithm, using “yEd” graph editor. Alrogithm has 3 separate parts: input, process and output, which are shown visually by columns.

**A diagram of a flowchart

AI-generated content may be incorrect.**

Breakdown of the algorithm:

 **Get the Student List:**

* Open the file "F1.txt".
* Read each line and put the student's code, name, and ID into a list.
* If you can't open the file, tell the user there's an error and stop.

 **Get the Resident List:**

* Open the file "F2.txt".
* Read each line and put the resident's ID and city into a list.
* If you can't open the file, tell the user there's an error and stop.

 **Ask for the City:**

* Ask the user, "Which city are you looking for?"
* Get the city name from the user.

 **Find the Matching Students:**

* Make a new empty list to hold the students who live in the city.
* For each student in the student list:
  + For each resident in the resident list:
    - If the student's ID matches the resident's ID AND the resident's city matches the city the user entered:
      * Add the student's information to the new list of matching students.

 **Show the Results:**

* For each student in the list of matching students:
  + Show the student's code, name, and ID.
* If the list of matching students is empty, tell the user that no students were found in that city.

 **End:**

* The program is done

### File content:

F1.txt:

123456 John Doe 11111111111

654321 Jane Smith 22222222222

987654 Peter Pan 33333333333

246810 Alice Wonderland 44444444444

135791 Bob The Builder 55555555555

F2.txt:

1111111111 Springfield

2222222222 Hill Valley

3333333333 Springfield

4444444444 Wonderland

5555555555 Bobville

# THE CODE IN C

A screenshot of a computer

AI-generated content may be incorrect.

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**EXPLANATION**

**The Program's Structure:**

* **The Main Program (int main()):** This is where the program starts. It's like the boss that tells the other parts what to do.
* **The File Checker (inputControl()):** This part is responsible for checking if the files the program needs are there and can be opened.
* **The Data Processor (process()):** This part does the main work of the program: reading the data, finding the students, and showing the results.

**How Your Program Works:**

1. **Checking the Files (inputControl()):**
   * The main() program first asks the inputControl() part to check the files "F1.txt" and "F2.txt".
   * The inputControl() part tries to open these files.
   * If inputControl() can open both files, it signals "success" (like returning 0 in the example).
   * If inputControl() fails to open a file, it signals "failure" (like returning -1 in the example).
   * If inputControl() signals "failure," the main() program doesn't do anything else and the program stops. It's like if you don't have all the ingredients, you can't cook the meal!
2. **Processing the Data (process()):**
   * If inputControl() signaled "success," the main() program tells the process() part to do its job.
   * The process() part does several things:
     + It opens the files "F1.txt" and "F2.txt" again (to actually read the data).
     + It reads the student information from "F1.txt" and the resident information from "F2.txt" and stores them in lists.
     + It gets the city name from the user.
     + It compares the student and resident data to find the students who live in the city the user entered.
     + It shows the information of the matching students.
3. **The Main Program's Role (int main()):**
   * The main() program is in charge of the overall flow.
   * It starts by using inputControl() to make sure the files are okay.
   * If the files are okay, it then uses process() to do the main job.
   * After process() finishes (if it worked correctly), the program ends.

# REFERENCES

Vladimir Viies. (2022). Homework formatting requirements. <https://www.tud.ttu.ee/im/Vladimir.Viies/materials/IAX0583_Programmeerimine%20I/pR0583en/Homeworks%20formatting.pdf> (12.03.2025).

<https://www.tud.ttu.ee/im/Vladimir.Viies/materials/IAX0584PROGRAMMEERIMINE%20II/PR2_iax0584/PR2en/PR2_en25/help3_25/kodu1_pr2_ex.docx> (12.03.2025)

Jason J. Sosa. (2019). <https://www.blackbox.ai/> (12.03.2025).