MINISTRY OF EDUCATION, CULTURE AND RESEARCH
OF THE REPUBLIC OF MOLDOVA
Technical University of Moldova
Faculty of Computers, Informatics and Microelectronics
Department of Software and Automation Engineering

Postica Valeria student

Group: FAF-243

Report

Laboratory Work No.5

of the *"Data Structures and Algorithms" course*

Checked:

Burlacu Natalia, *PhD, associate professor*

Department of Software and Automatic Engineering,

FCIM Faculty, TUM

Chisinau – 2025

The Purpose:

Develop a procedural-style program in C / C++, using own written functions. Data processing in your program should be organized according to a given length of input records based on memory allocation functions.

Your solution should:

A. use the pointers;
B. have to be presented in your report, emerging from the content of the problem statement.
C. To draw the block diagram corresponding to the solved problem.

----------Program 5.1----------

Problem Condition:

Write a C program able to perform the following operations:

- ✓ Create an input text file (input.txt) (in the location of the source code);
- ✓ Opens the given file (see pt.1) and allows the input of **a string** *(string should be read from the keyboard),* storing the entered string in the working file;

Additionally, write the following C functions (later to be called in the main routine) which should do:

- ✓ *Identifying the presence of double spaces combinations in the string from the work file;*
- ✓ *Replacing all double spaces with a single space;*
- ✓ *Calculating the length of the string in the input file after doing the preview pts.;*
- ✓ *Displaying on the screen the output information about:*
  - *How many double spaces combinations were replaced by single spaces in the string in the given input file;*
  - *What is the strings length after doing the preview pts.;*
- ✓ *Writing in the output file* (output.txt) *the earlier displayed on the screen data about how many spaces have been found in the string in the given input file and what is the string length after modifications.*

Figure 1.1 - Problem Condition

1. The program code, including relevant comments within it.

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

// function to identify and replace double spaces and return count
int processDoubleSpaces(char *str) {
    int count = 0;
    int len = strlen(str);
    int j = 0;

    // Temporary string to store result
    char temp[1000];

    for(int i = 0; i < len; i++) {
        // Check for double spaces
        if(str[i] == ' ' && str[i+1] == ' ') {
            count++;
            // Skip the second space
            continue;
        }
        temp[j++] = str[i];
    }
    temp[j] = '\0';

    // Copy processed string back to original
    strcpy(str, temp);

    return count;
}

int main() {
    // Create and open input file
    FILE *inputFile = fopen("input.txt", "w+");
    if(inputFile == NULL) {
        printf("Error creating input file!\n");
        return 1;
    }

    // Get string from keyboard
    char inputString[1000];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);

    //remove trailing newline if present
    if(inputString[strlen(inputString)-1] == '\n') {
        inputString[strlen(inputString)-1] = '\0';
    }

    //write string to input file
    fprintf(inputFile, "%s", inputString);
    fclose(inputFile);

    // Reopen file for reading
    inputFile = fopen("input.txt", "r");
    if(inputFile == NULL) {
        printf("Error opening input file!\n");
        return 1;
```

```c
    }

    // Read string from file
    char fileString[1000];
    fgets(fileString, sizeof(fileString), inputFile);
    fclose(inputFile);

    // Process the string
    int doubleSpacesCount = processDoubleSpaces(fileString);
    int finalLength = strlen(fileString);

    // Display results
    printf("\nResults:\n");
    printf("Number of double spaces replaced: %d\n", doubleSpacesCount);
    printf("Final string length: %d\n", finalLength);
    printf("Formated string: %s\n",fileString);

    // Write results to output file
    FILE *outputFile = fopen("output.txt", "w");
    if(outputFile == NULL) {
        printf("Error creating output file!\n");
        return 1;
    }

    fprintf(outputFile, "Number of double spaces replaced: %d\n",
doubleSpacesCount);
    fprintf(outputFile, "Final string length: %d\n", finalLength);
    fprintf(outputFile, "Formated string: %s\n",fileString);
    fclose(outputFile);

    return 0;
}
```

----------Block Diagrams----------

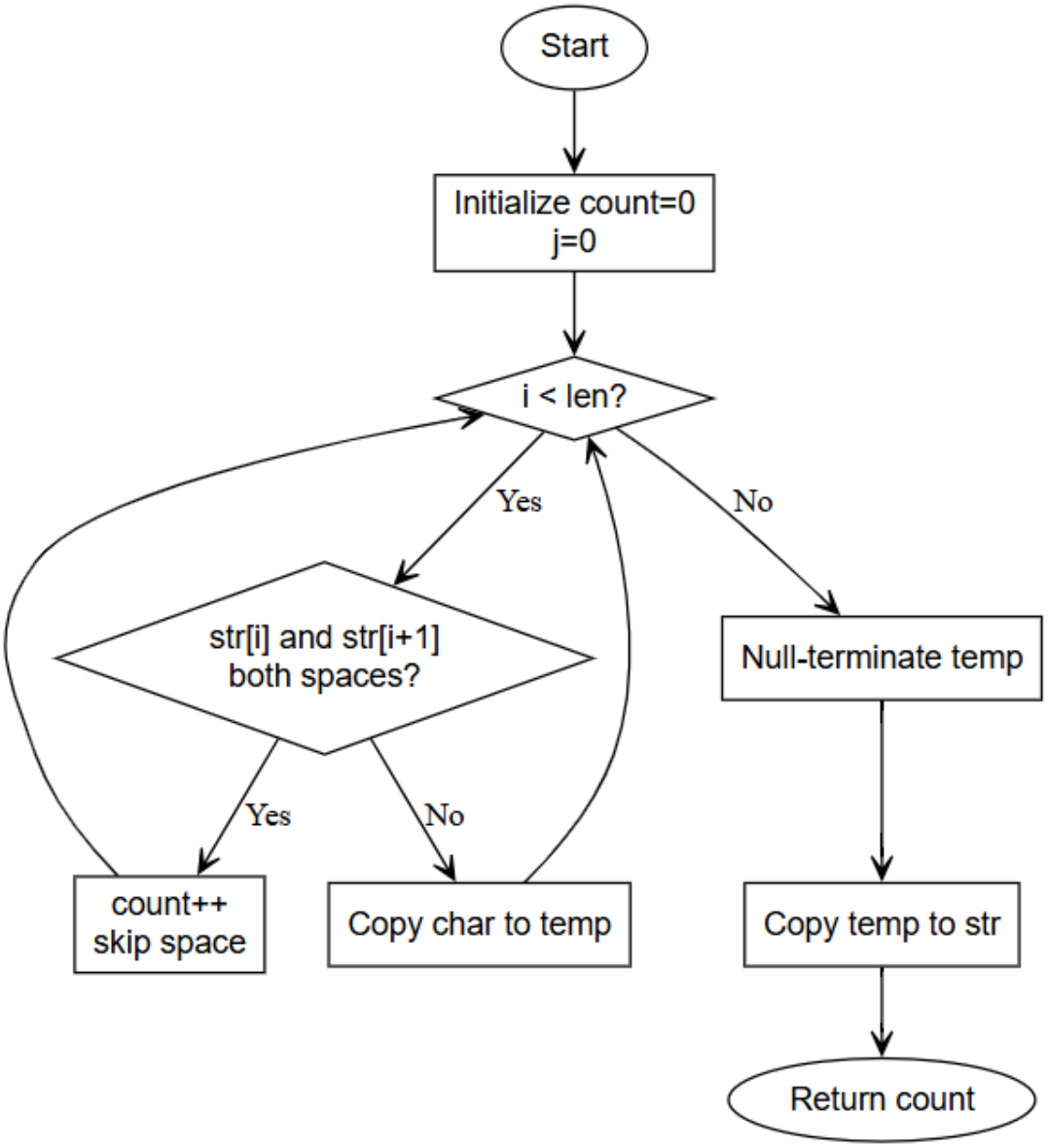

Figure 2.1 processDoubleSpaces()

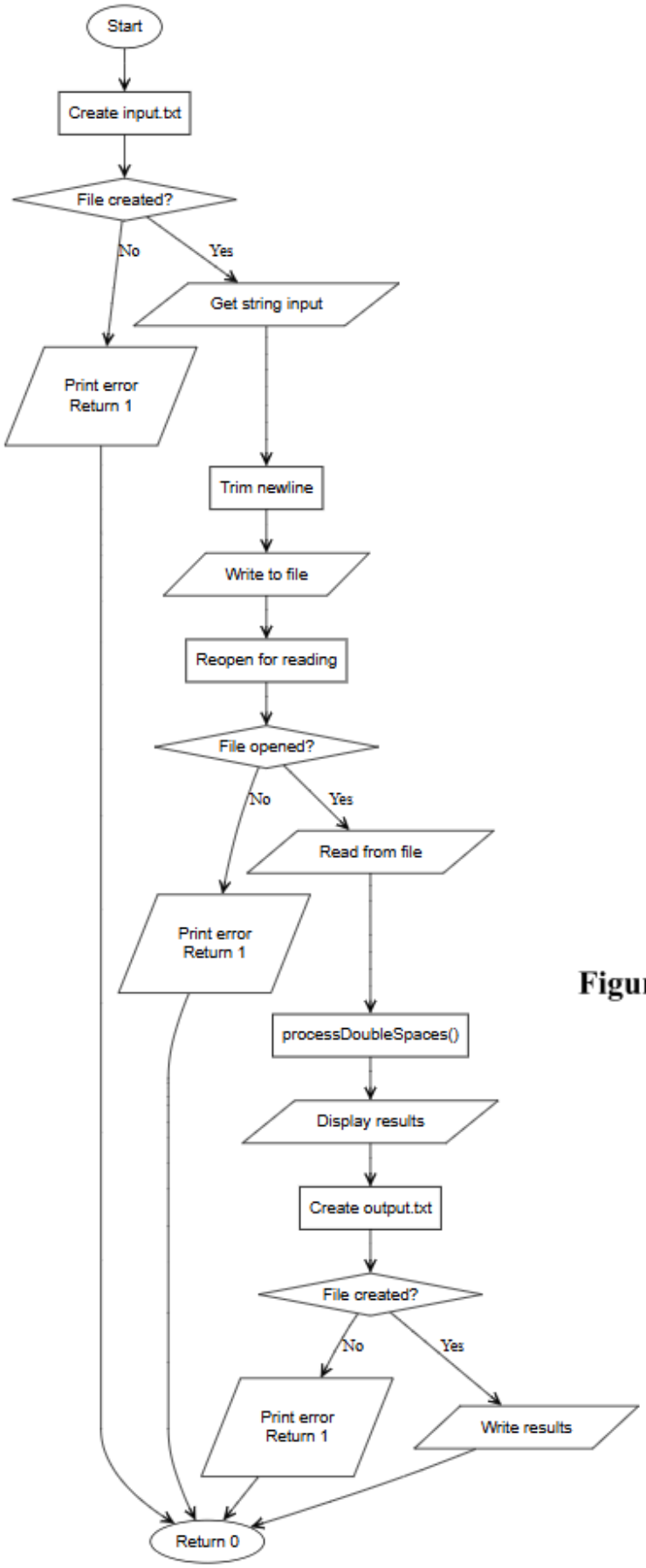

Figure 2.2 main()

Output:

```
Enter a string:This s  tr ing co    ntain many   dou   ble spa ce    s

Results:
Number of double spaces replaced: 11
Final string length: 44
Formated string: This s tr ing co ntain many dou ble spa ce s

Process finished with exit code 0
```

Figure 3.1 - The output window

```
CMakeLists.txt   main.c   input.txt   output.txt
1   This s  tr ing co    ntain many   dou   ble spa ce    s
```

Figure 3.2 - The input.txt

```
CMakeLists.txt   main.c   input.txt   output.txt
1   Number of double spaces replaced: 11
2   Final string length: 44
3   Formated string: This s tr ing co ntain many dou ble spa ce s
4
```

Figure 3.3 - The output.txt

Conclusion:

In this laboratory, I deepened my understanding of files, how to work with them combined with a struct, all of them related throw functions based on pointers. This programming exercise provided valuable hands-on experience with fundamental C concepts, particularly string manipulation and file handling. The task of identifying and removing double spaces helped reinforce my understanding of character arrays, string functions, and basic text processing.

----------Program 5.2----------

Problem Condition:

It is given a structure **The registry of the citizens** should contain the following components: **name, surname, date of birth (date, month, year), gender (m, f), home address, work address.**
Solve the problem in C that performs the following operations:

- ✓ Create a text file *(experiment.txt)* (in the source code save location);
- ✓ Opens the textual file with the given name, allows entering the data determined by the given structure, storing them in the working file;
- ✓ Displays on the screen the structure data recently entered in the given file.

Your program should be compound from the following C functions (with subsequent calls from the main):

1. Your program will calculate the quantity of citizens of different gender (m & f), the data about the quantity of citizens of each gender is stored in *output.txt.*

2. Your program will write the citizens' records by gender into two different files: *male.txt & female.txt.*
3. The second version of your program will copy all the data from *output.txt* to the beginning of the *experiment.txt* file, without affecting the previously entered data.
4. *male.txt & female.txt* files should have the opportunity to be opened in the read mode.

1. **The program code, including relevant comments within it.**

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

#define LIMIT 100
```

```c
// Citizen structure definition
typedef struct {
    char name[50];
    char surname[50];
    struct {
        int day;
        int month;
        int year;
    } date_of_birth;
    char gender;
    char home_address[LIMIT];
    char work_address[LIMIT];
} Citizen;

// Input citizen data from user
void inputCitizen(Citizen *c) {
    printf("Enter name: ");
    scanf("%s", c->name);
    printf("Enter surname: ");
    scanf("%s", c->surname);
    printf("Enter date of birth (day month year): ");
    scanf("%d %d %d", &c->date_of_birth.day, &c->date_of_birth.month,
&c->date_of_birth.year);
    printf("Enter gender (m/f): ");
    scanf(" %c", &c->gender);
    getchar(); // Clear input buffer
    printf("Enter home address: ");
    fgets(c->home_address, LIMIT, stdin);
    c->home_address[strcspn(c->home_address, "\n")] = '\0'; // Remove newline
    printf("Enter work address: ");
    fgets(c->work_address, LIMIT, stdin);
    c->work_address[strcspn(c->work_address, "\n")] = '\0';
}

// Display citizen information
void displayCitizen(const Citizen *c) {
    printf("\nName: %s %s\n", c->name, c->surname);
    printf("Date of Birth: %02d/%02d/%04d\n",
           c->date_of_birth.day, c->date_of_birth.month, c->date_of_birth.year);
    printf("Gender: %c\n", c->gender);
    printf("Home Address: %s\n", c->home_address);
    printf("Work Address: %s\n", c->work_address);
}

// Write all citizens to experiment.txt
void writeToExperimentFile(Citizen *citizens, int n) {
    FILE *file = fopen("experiment.txt", "w");
    if (file == NULL) {
        printf("Error opening experiment.txt for writing!\n");
        return;
    }

    for (int i = 0; i < n; i++) {
        fprintf(file, "%s %s %d %d %d %c %s %s\n",
                citizens[i].name, citizens[i].surname,
                citizens[i].date_of_birth.day,
                citizens[i].date_of_birth.month,
                citizens[i].date_of_birth.year,
                citizens[i].gender,
```

```
                citizens[i].home_address,
                citizens[i].work_address);
    }
    fclose(file);
    printf("Data written to experiment.txt\n");
}

// Count and write gender statistics to output.txt
void countAndWriteGenders(Citizen *citizens, int n) {
    int maleCount = 0, femaleCount = 0;

    for (int i = 0; i < n; i++) {
        if (citizens[i].gender == 'm' || citizens[i].gender == 'M') maleCount++;
        else if (citizens[i].gender == 'f' || citizens[i].gender == 'F')
femaleCount++;
    }

    FILE *file = fopen("output.txt", "w");
    if (file == NULL) {
        printf("Error opening output.txt for writing!\n");
        return;
    }

    fprintf(file, "Male citizens: %d\nFemale citizens: %d\n", maleCount,
femaleCount);
    fclose(file);
    printf("Gender counts written to output.txt\n");
}

// Separate citizens by gender into different files
void separateByGender(Citizen *citizens, int n) {
    FILE *maleFile = fopen("male.txt", "w+");
    FILE *femaleFile = fopen("female.txt", "w+");

    if (maleFile == NULL || femaleFile == NULL) {
        printf("Error opening gender files!\n");
        if (maleFile) fclose(maleFile);
        if (femaleFile) fclose(femaleFile);
        return;
    }

    // Write to appropriate files based on gender
    for (int i = 0; i < n; i++) {
        if (citizens[i].gender == 'm' || citizens[i].gender == 'M') {
            fprintf(maleFile, "%s %s %d %d %d %s %s\n",
                    citizens[i].name, citizens[i].surname,
                    citizens[i].date_of_birth.day,
                    citizens[i].date_of_birth.month,
                    citizens[i].date_of_birth.year,
                    citizens[i].home_address,
                    citizens[i].work_address);
        } else if (citizens[i].gender == 'f' || citizens[i].gender == 'F') {
            fprintf(femaleFile, "%s %s %d %d %d %s %s\n",
                    citizens[i].name, citizens[i].surname,
                    citizens[i].date_of_birth.day,
                    citizens[i].date_of_birth.month,
                    citizens[i].date_of_birth.year,
                    citizens[i].home_address,
                    citizens[i].work_address);
```

```c
        }
    }

    // Display contents of created files
    rewind(maleFile);
    rewind(femaleFile);

    printf("\nContents of male.txt:\n");
    char line[256];
    while (fgets(line, sizeof(line), maleFile)) printf("%s", line);

    printf("\nContents of female.txt:\n");
    while (fgets(line, sizeof(line), femaleFile)) printf("%s", line);

    fclose(maleFile);
    fclose(femaleFile);
    printf("\nCitizens separated into male.txt and female.txt\n");
}
// ASCII Art Header with simple symbols
void printHeader() {
    printf("\n");
    printf("  /\\_/\\    /----------------\\\n");
    printf(" ( o.o )   | FILE MERGER 3000 |\n");
    printf("  > ^ <    \\----------------/\n");
    printf("\n");
}

// Progress bar remains the same
void showProgress(int step, int total) {
    printf("\rMerging: [");
    for (int i = 0; i < 20; i++) {
        printf(i < (step * 20 / total) ? "#" : " ");
    }
    printf("] %d%%", step * 100 / total);
    fflush(stdout);
}

void copyOutputToExperiment() {
    // Cute header
    printf("\n");
    printf("  /\\_/\\    File Merger 9000!\n");
    printf(" ( o.o )  \n");
    printf("  > ^ <    Merging files...\n");
    printf("\n");

    FILE *outputFile = fopen("output.txt", "r");
    if (outputFile == NULL) {
        printf("(╯°□°)╯︵ ┻━┻ Error opening output.txt!\n");
        return;
    }

    FILE *experimentFile = fopen("experiment.txt", "r");
    if (experimentFile == NULL) {
        printf("(╯°□°)╯︵ ┻━┻ Error opening experiment.txt!\n");
        fclose(outputFile);
        return;
    }
```

```c
    FILE *tempFile = fopen("temp.txt", "w");
    if (tempFile == NULL) {
        printf("(ノ°□°)ノ︵ ┻━┻ Error creating temp file!\n");
        fclose(outputFile);
        fclose(experimentFile);
        return;
    }

    // Show progress
    printf("Copying: [");
    int dotCount = 0;

    char ch;
    while ((ch = fgetc(outputFile)) != EOF) {
        fputc(ch, tempFile);
        if (dotCount++ % 1000 == 0) {
            printf(".");
            fflush(stdout);
        }
    }

    while ((ch = fgetc(experimentFile)) != EOF) {
        fputc(ch, tempFile);
        if (dotCount++ % 1000 == 0) {
            printf(".");
            fflush(stdout);
        }
    }

    printf("] Done!\n");

    fclose(outputFile);
    fclose(experimentFile);
    fclose(tempFile);

    remove("experiment.txt");
    rename("temp.txt", "experiment.txt");

    // Success message with cute art
    printf("\n");
    printf("  /\\_/\\    Success!\n");
    printf(" ( ^.^ )  \n");
    printf("  > _ <   output.txt + experiment.txt merged!\n");
    printf("\n");
}
int main() {
    int n;
    printf("Enter number of citizens: ");
    scanf("%d", &n);

    Citizen *citizens = malloc(n * sizeof(Citizen));

    // Input citizen data
    for (int i = 0; i < n; i++) {
        printf("\nCitizen %d:\n", i + 1);
        inputCitizen(&citizens[i]);
    }

    // Perform file operations
```

```
    writeToExperimentFile(citizens, n);
    countAndWriteGenders(citizens, n);
    separateByGender(citizens, n);
    copyOutputToExperiment();

    // Display all citizens
    printf("\nAll citizens:\n");
    for (int i = 0; i < n; i++) {
        displayCitizen(&citizens[i]);
    }

    free(citizens);
    return 0;
}
```

----------Block Diagrams----------

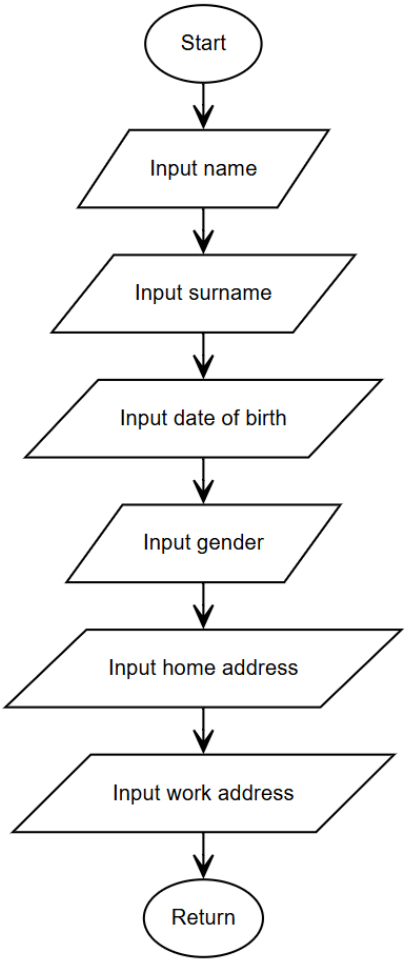

Figure 2.1 inputCitizen()

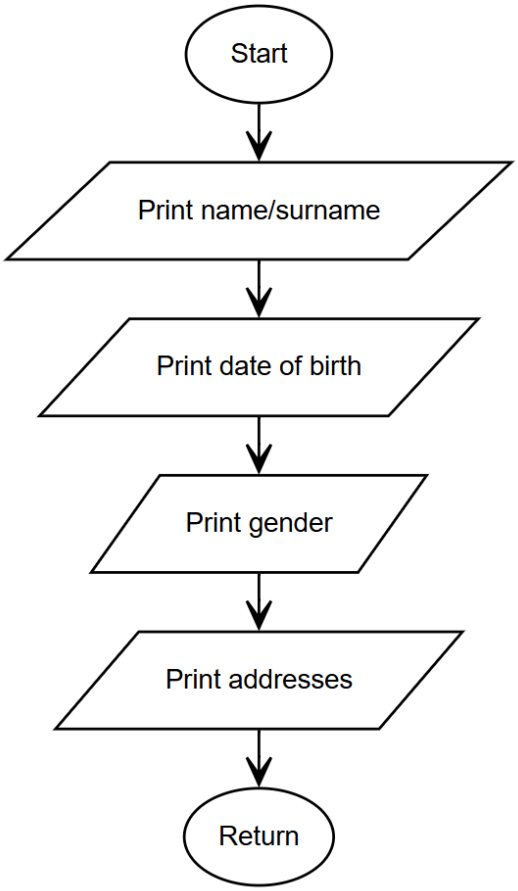

Figure 2.2 displayCitizen()

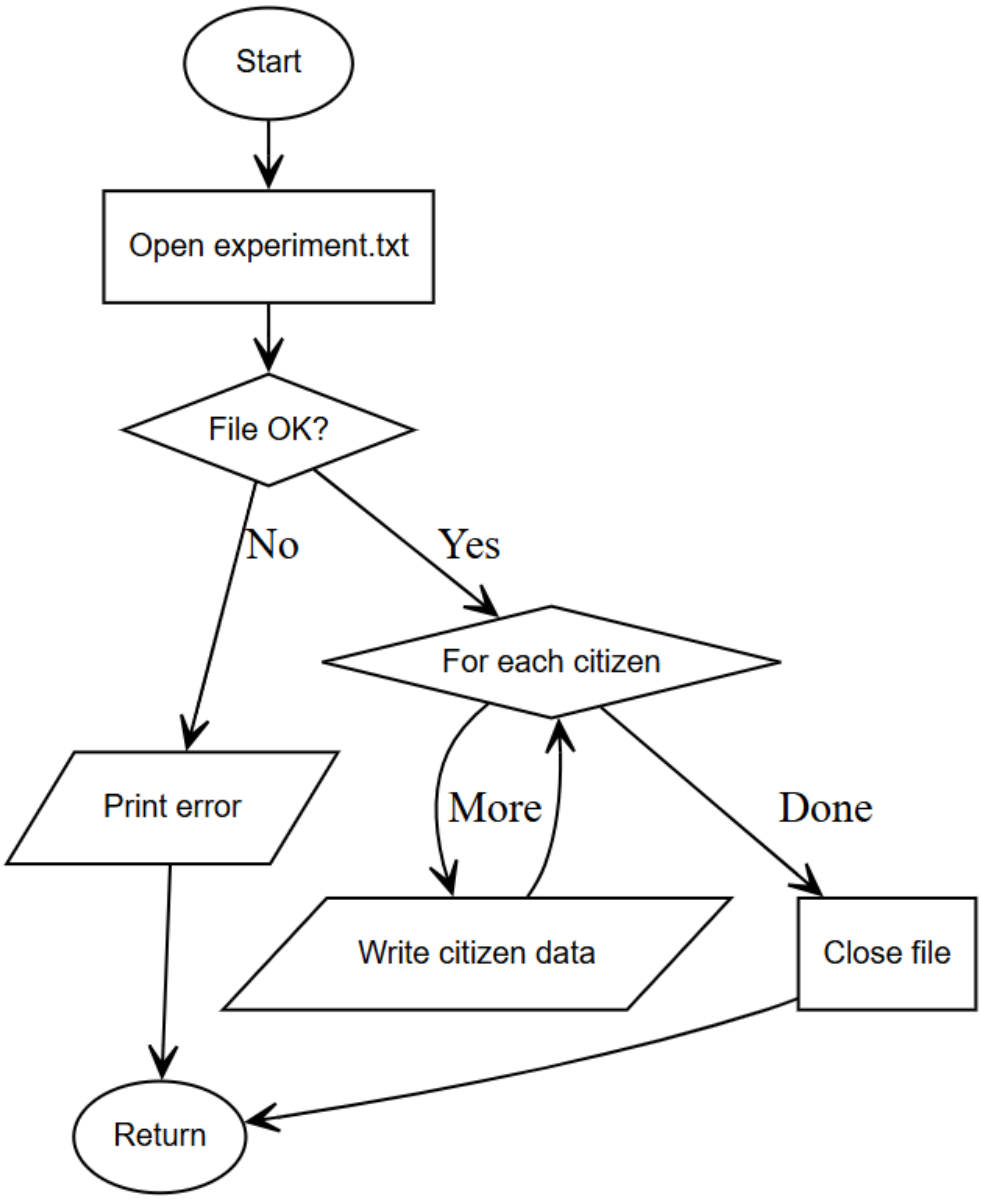

Figure 2.3 writeToExperimentFile()

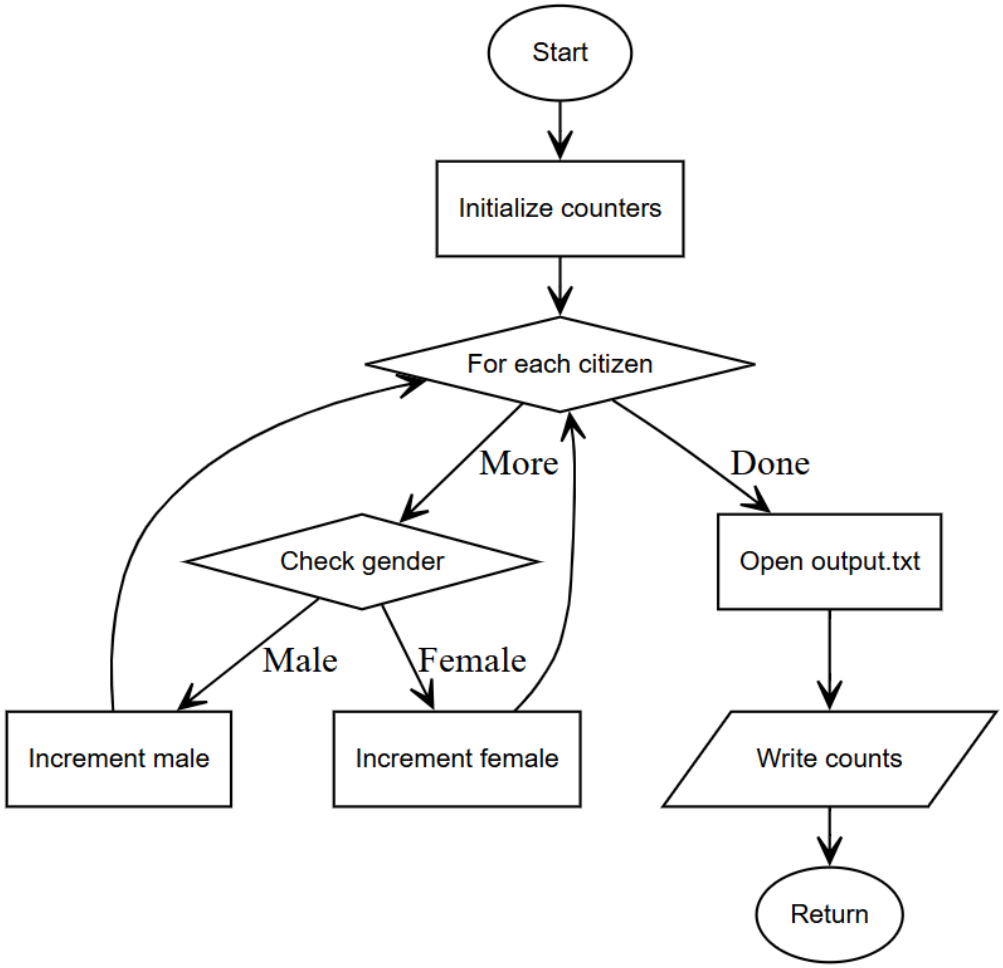

Figure 2.4 countAndWriteGenders()

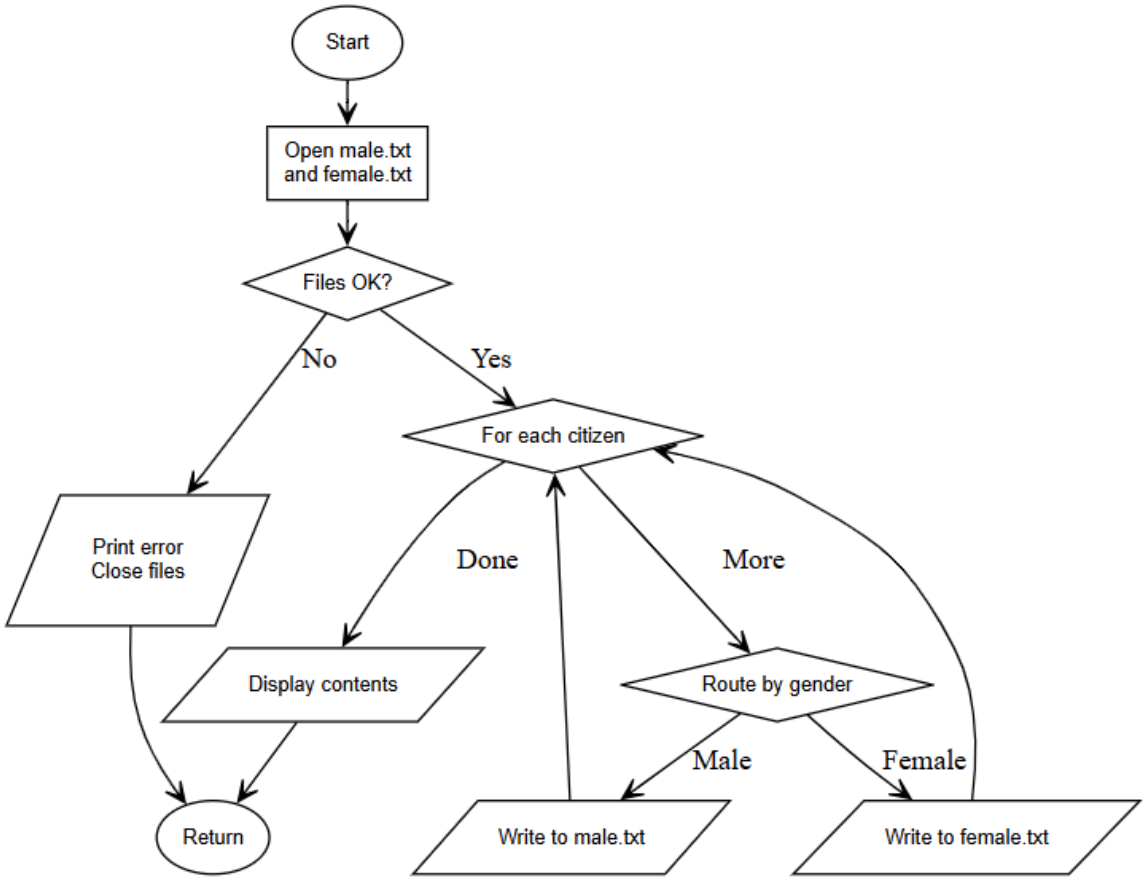

Figure 2.5 separateByGenders()

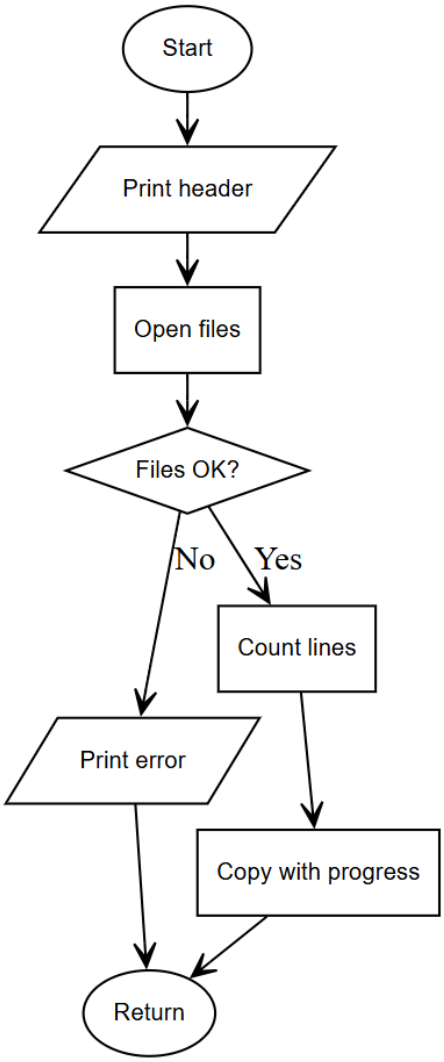

Figure 2.6 copyOutpuToExperiment()

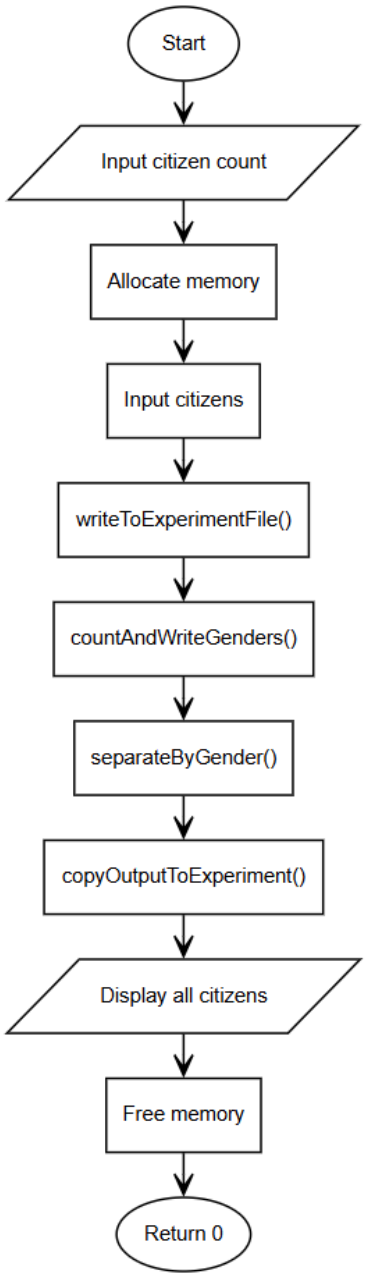

Figure 2.7 main()

Output:

```
Data written to experiment.txt
Gender counts written to output.txt

Contents of male.txt:
Victor Sirbu 23 8 1993 Alecsandri 44 Eminescu 12

Contents of female.txt:
Ecaterina Mare 12 3 2000 Moscova 7 Centrala 9
Mariana Stoianova 4 2 2004 Miorita 13 Centrala 42

Citizens separated into male.txt and female.txt

  /\_/\    File Merger 9000!
 ( o.o )
  > ^ <    Merging files...

Copying: [.] Done!

  /\_/\    Success!
 ( ^.^ )
  > _ <   output.txt + experiment.txt merged!
```

```
All citizens:

Name: Ionel Barliga
Date of Birth: 24/05/2001
Gender: m
Home Address: Visinilor 42
Work Address: Centrala 2

Name: Ecaterina Mare
Date of Birth: 12/03/2000
Gender: f
Home Address: Moscova 7
Work Address: Centrala 9

Name: Victor Sirbu
Date of Birth: 23/08/1993
Gender: m
Home Address: Alecsandri 44
Work Address: Eminescu 12

Name: Mariana Stoianova
Date of Birth: 04/02/2004
Gender: f
Home Address: Miorita 13
Work Address: Centrala 42

Process finished with exit code 0
```

Figure 3.1 - The output window

CMakeLists.txt | main.c | experiment.txt | female.txt | male.txt | output.txt

```
Male citizens: 2
Female citizens: 2
Ionel Barliga 24 5 2001 m sinilor 42 Centrala 2
Ecaterina Mare 12 3 2000 f Moscova 7 Centrala 9
Victor Sirbu 23 8 1993 m Alecsandri 44 Eminescu 12
Mariana Stoianova 4 2 2004 f Miorita 13 Centrala 42
```

Figure 3.2 - The experiment.txt file

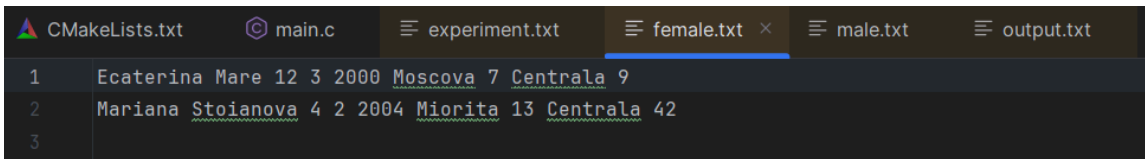

Figure 3.3 - The female.txt file

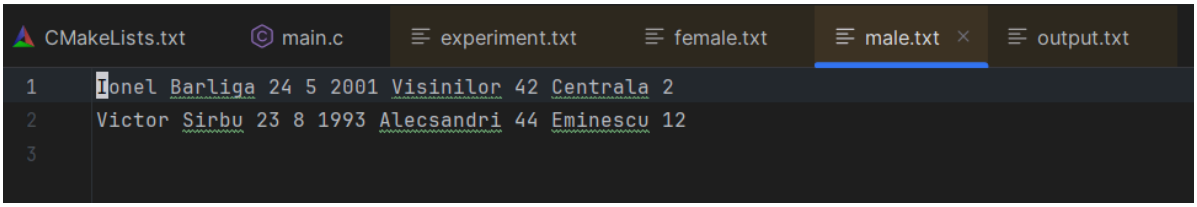

Figure 3.4 - The male.txt file

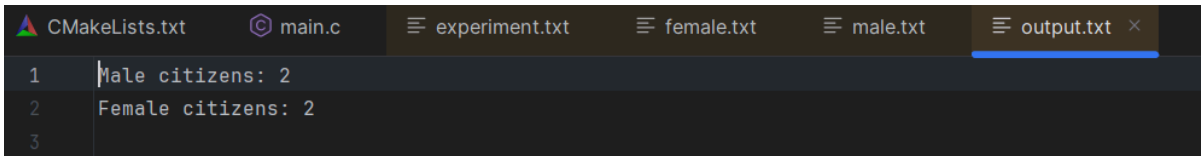

Figure 3.4 - The output.txt file

Conclusion:

In this laboratory I worked mostly with structures and files in C by a new approach, working with several files at the same time. I created a program to store citizen data, including their names, birth dates, and addresses. While the program works well, I could improve it by adding better input checks. Overall, this was good practice for managing structured data in C and would help me to work with bigger and more complex projects.