|  |  |
| --- | --- |
| Student Name | : TAN ZHI LIN |
| Matric | : 206730 |
| Link to GitHub | : <https://github.com/ravenCrown0627/calculate_cgpa> |
| Total Points (20 pts) | : |
| **Due date: 7th Nov, 2023 before 9pm** | |

**Assignment 1**

**ECC4802**

**Calculating CGPA**

**Analysis of Problem**

To record multiple students results, the Java program should read multiple CSV files that contain information about students' results. All the CSV files had been stored at the directory named as ./*csv*. The category of the CSV files can be classified as shown at the table below.

*Table 1 CSV Files Classification*

|  |  |
| --- | --- |
| **File name** | **Description** |
| *course\_info.csv* | This file including the course code taken by student for each semester and their corresponding credit hour. Assume that the first line is 0, the even line number is showing the course code and the odd line number is showing the course credit hour. |
| *<student\_name>\_result.csv* | This file including the result of each course taken by student for each semester. The first row is the name of student followed by each row of the grade value representing the result of courses taken in a semester.  The test cases prepared for the program is listed below:   * zhilin\_result.csv * yewy\_result.csv * tabina\_result.csv * hasif\_result.csv * shisilia\_result.csv |

Once the CSV files had been parsed, the student’s Grade Point Average (GPA) for each semester will be calculated using the formula below as stated by the School of Graduate Studies Universiti Putra Malaysia.

*Figure 1 Formula to calculate student’s GPA for a semester.*

To determine a student's Cumulative Grade Point Average (CGPA), it can obtain by dividing the total sum of grade values for all eight semesters by the combined credit hours completed during those semesters as shown in Figure 2.

*Figure 2 Formula to calculate student’s CGPA throughout 8 semesters.*

While the GPA and CGPA for a student had been calculated, their result had been printed on console and saved as a text file in the meantime. The name of text file is in the format of *<student\_name>\_semester\_detail\_output.txt* under the directory of ./*csv/output*.

The pseudo code for the program had been shown in Figure 3.

A white paper with black text

Description automatically generated

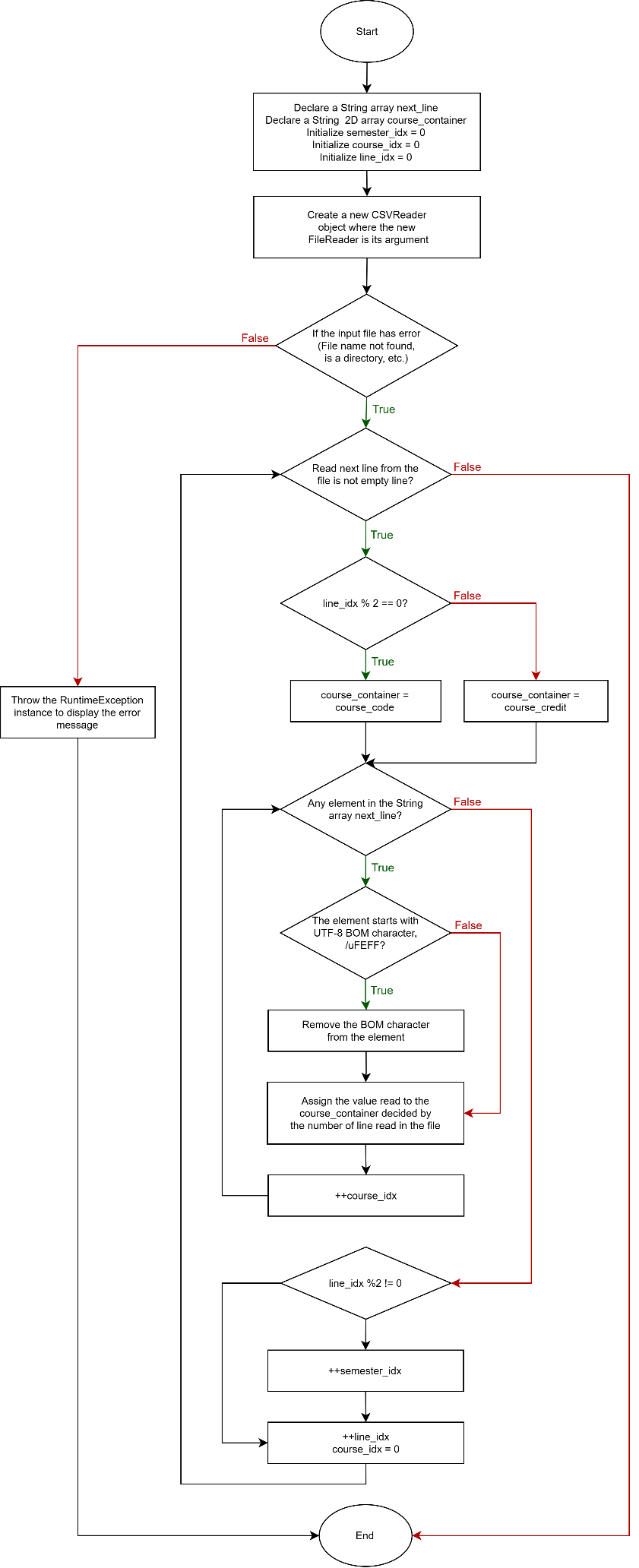
*Figure 3 Pseudo code for CalculateCGPA*

**Flowchart**

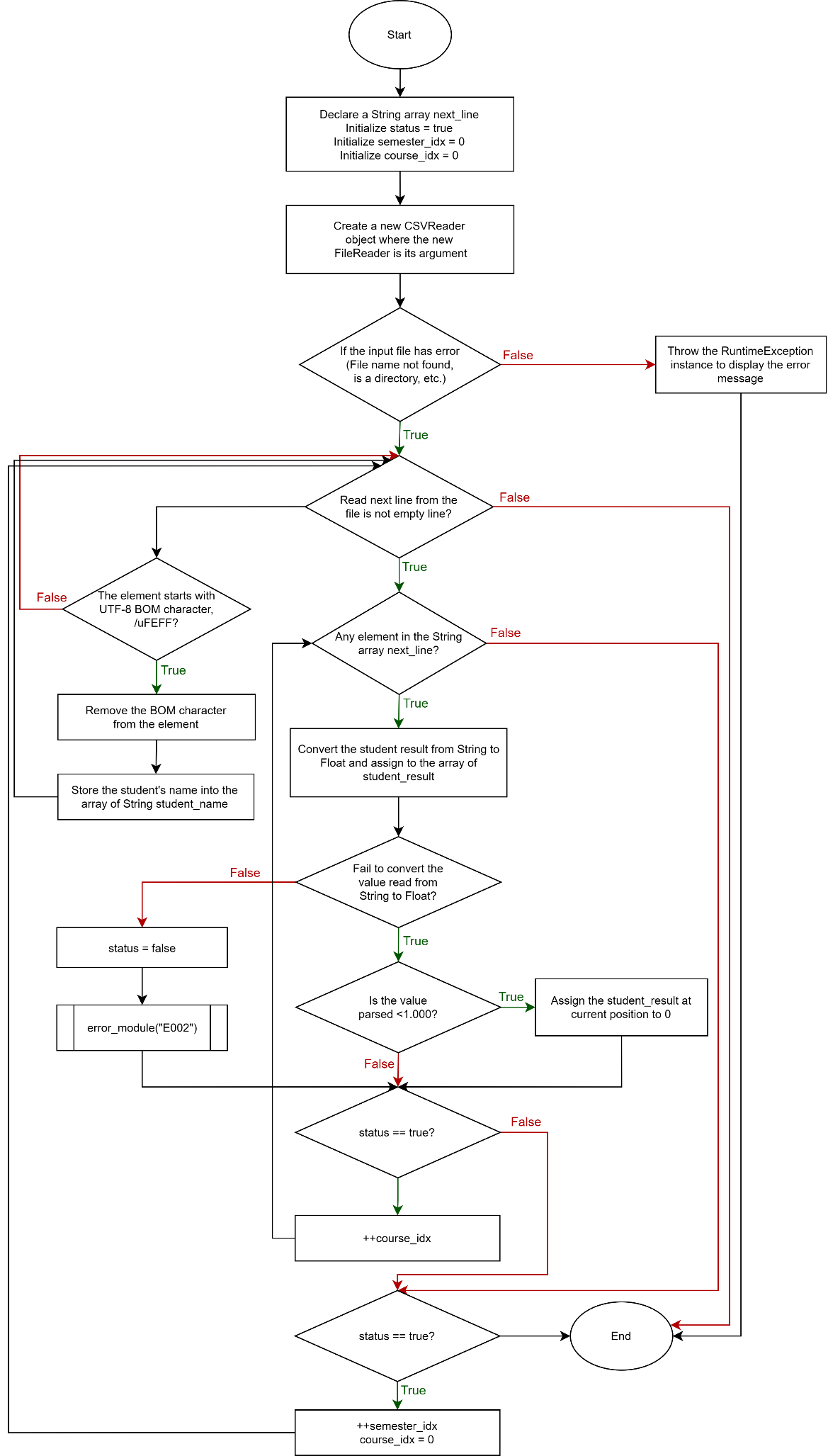
A screenshot of a computer

Description automatically generated

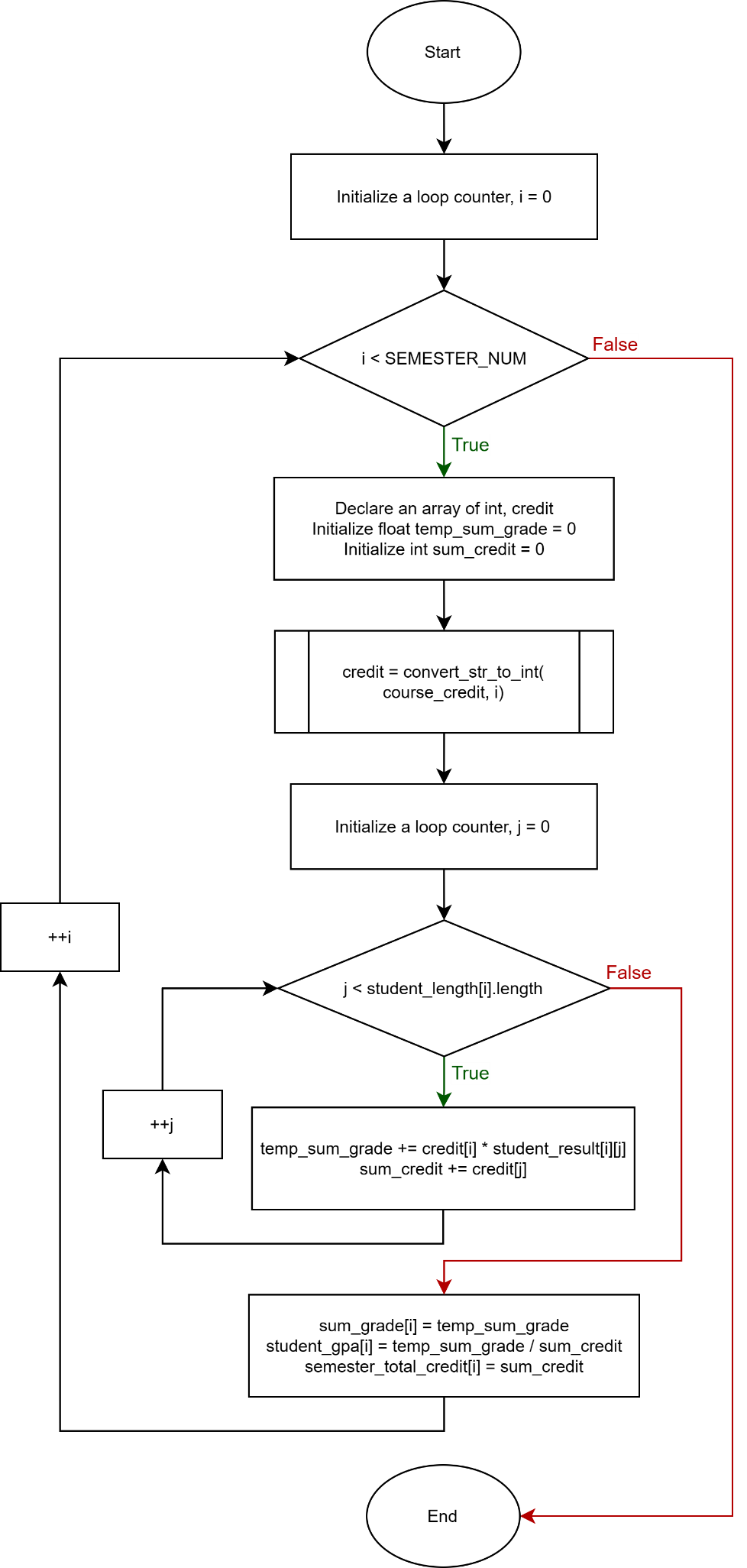
*Figure 4 Flowchart for main()*



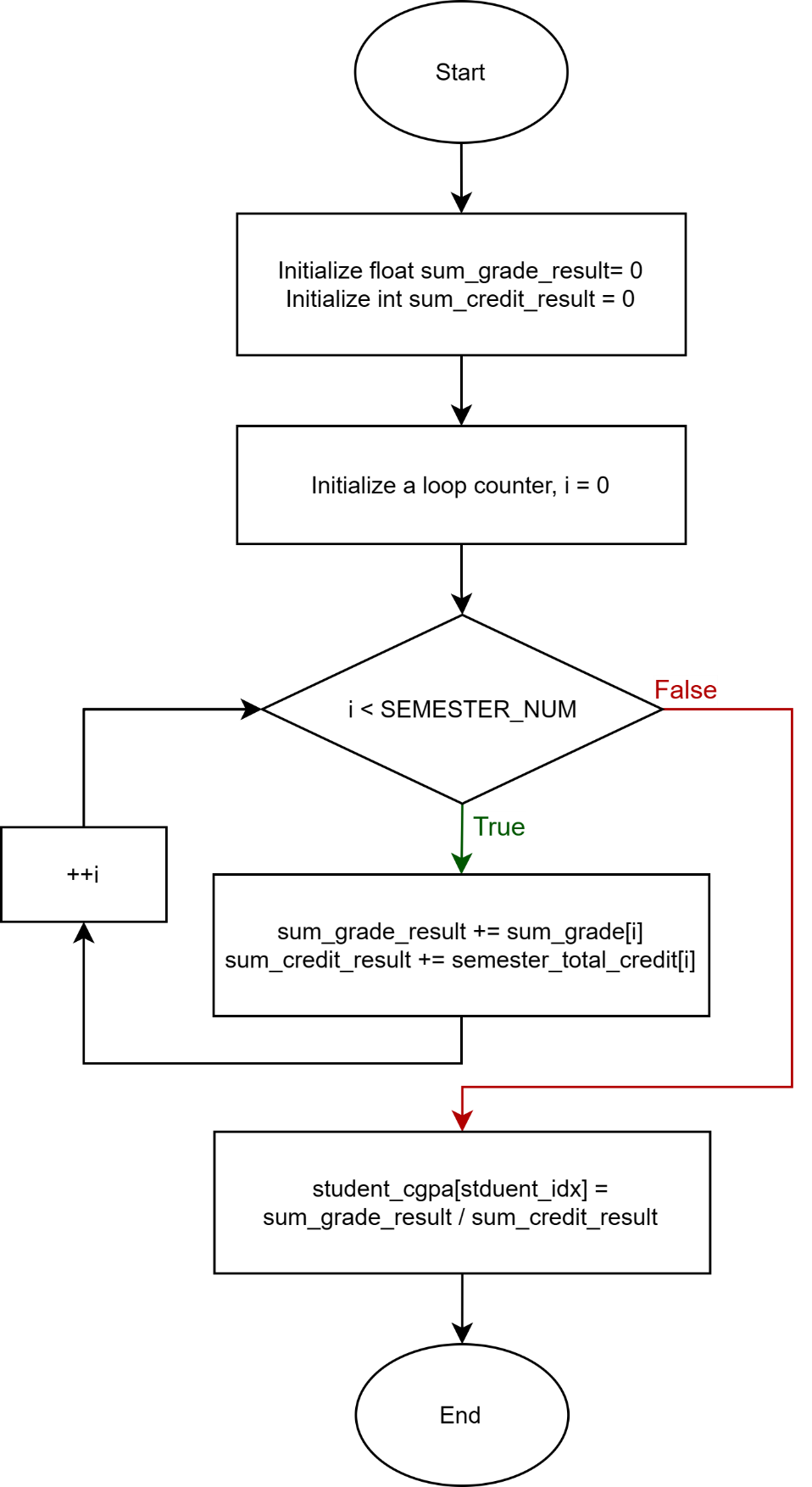
*Figure 5 Flowchart for parse\_course\_info()*



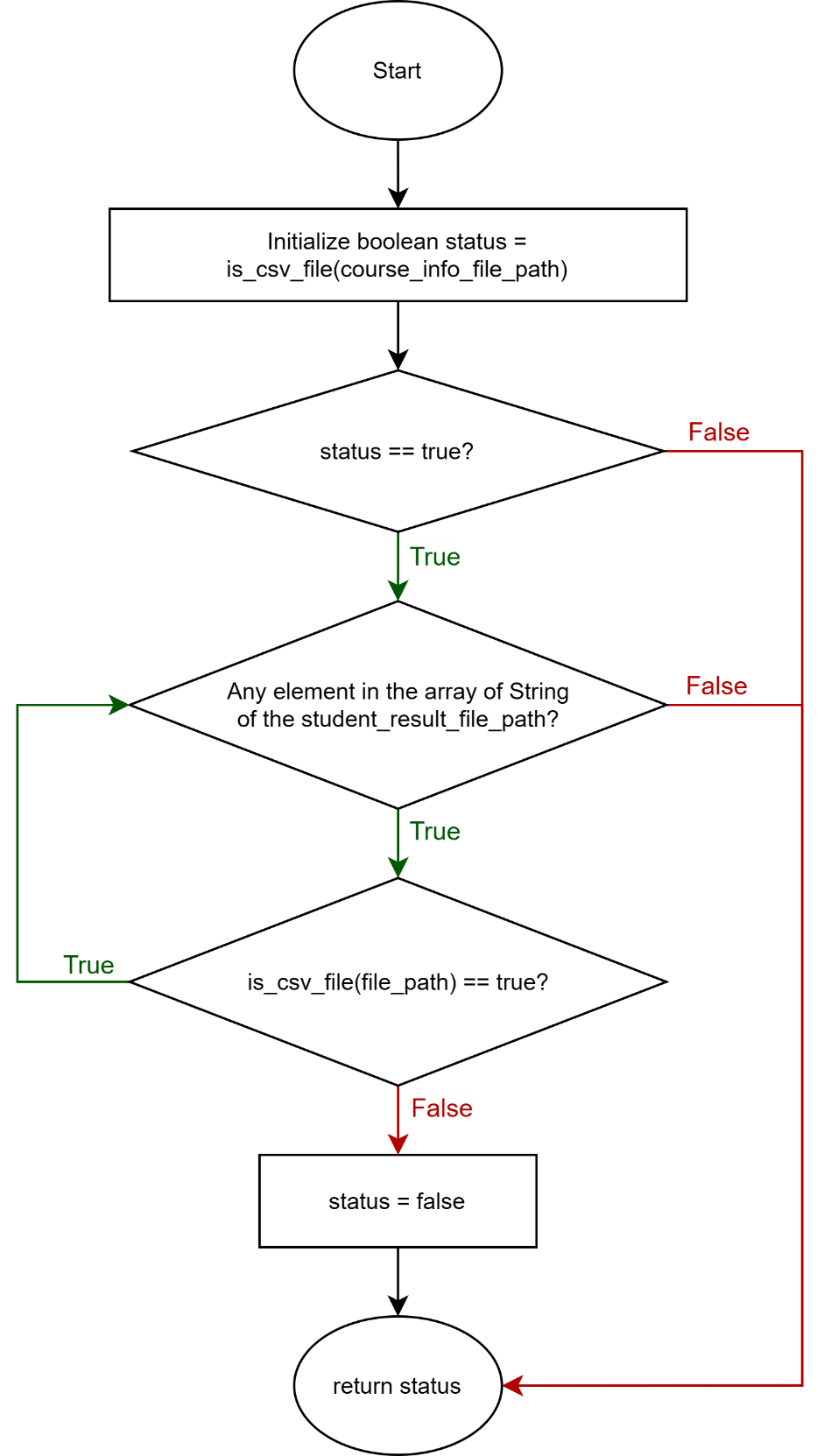
*Figure 6 Flowchart for parse\_student\_result()*



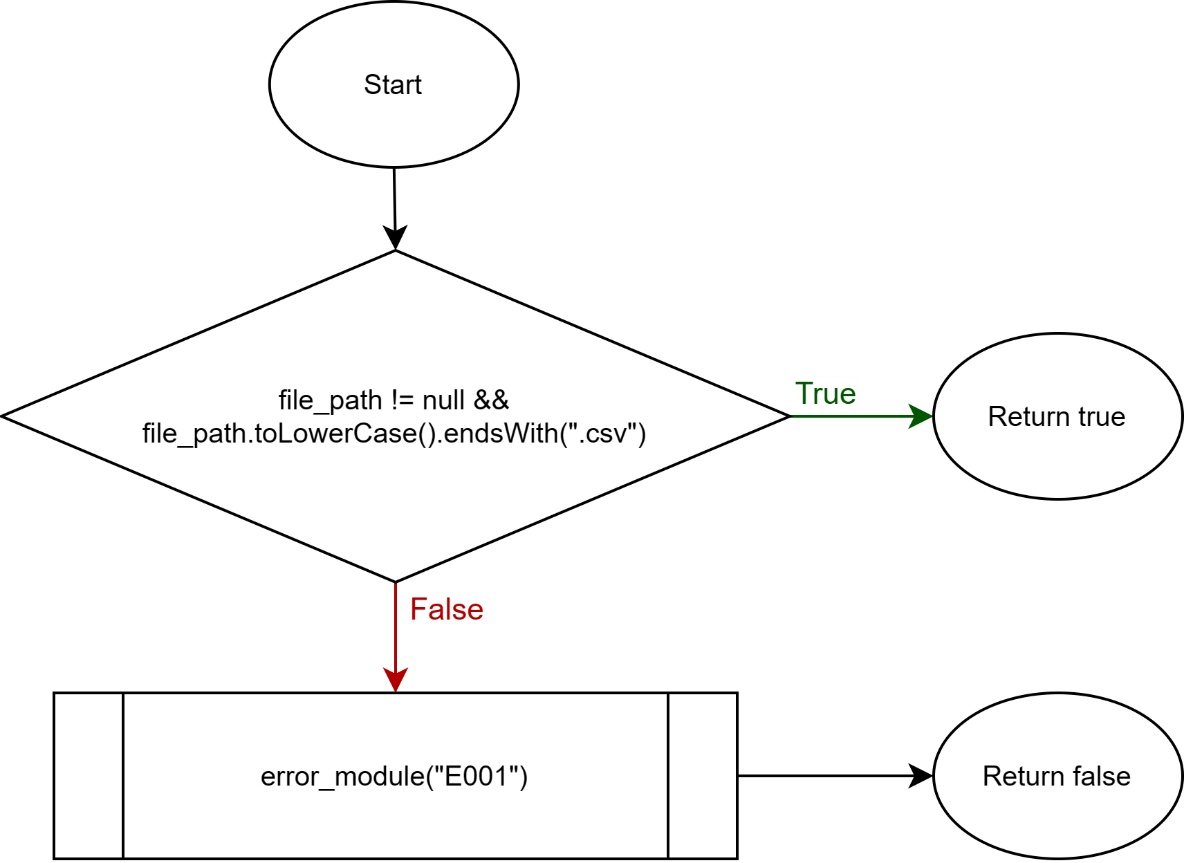
*Figure 7 Flowchart for calc\_gpa()*



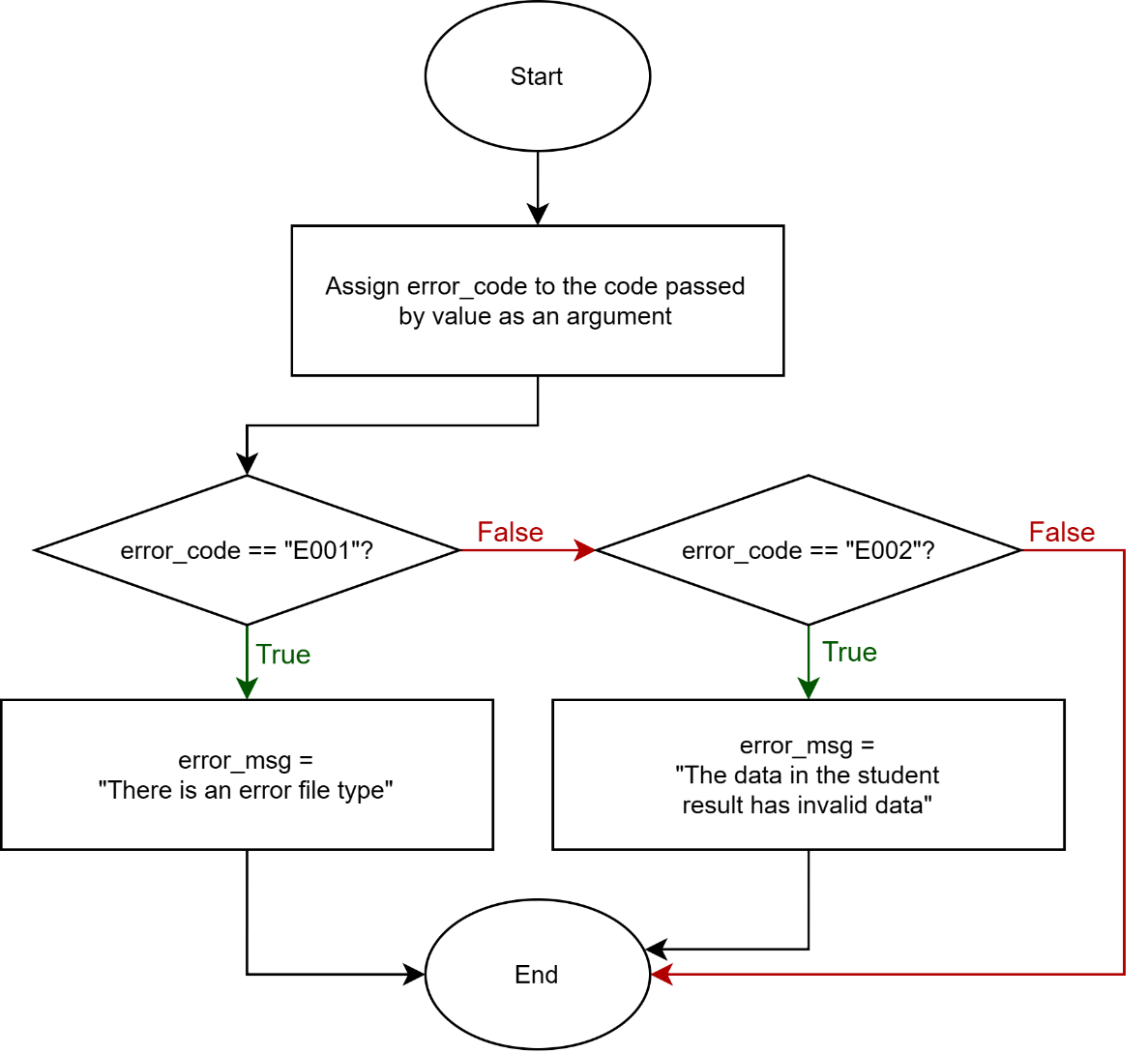
*Figure 8 Flowchart for calc\_cgpa()*



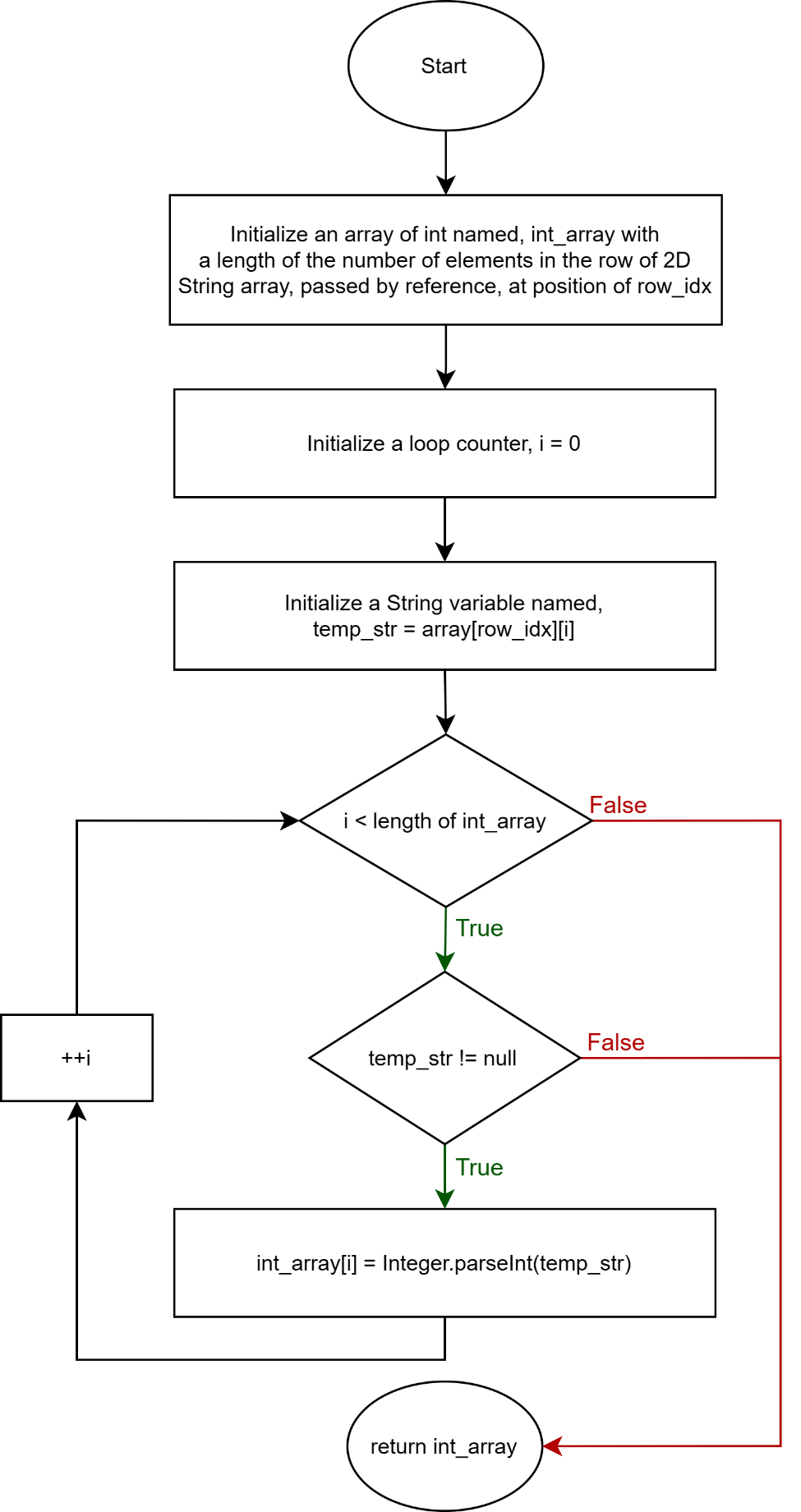
*Figure 9 Flowchart for check\_file\_path()*



*Figure 10 Flowchart for is\_csv\_file()*



*Figure 11 Flowchart for error\_module()*



*Figure 12 Flowchart for convert\_str\_to\_int()*

*A screenshot of a computer

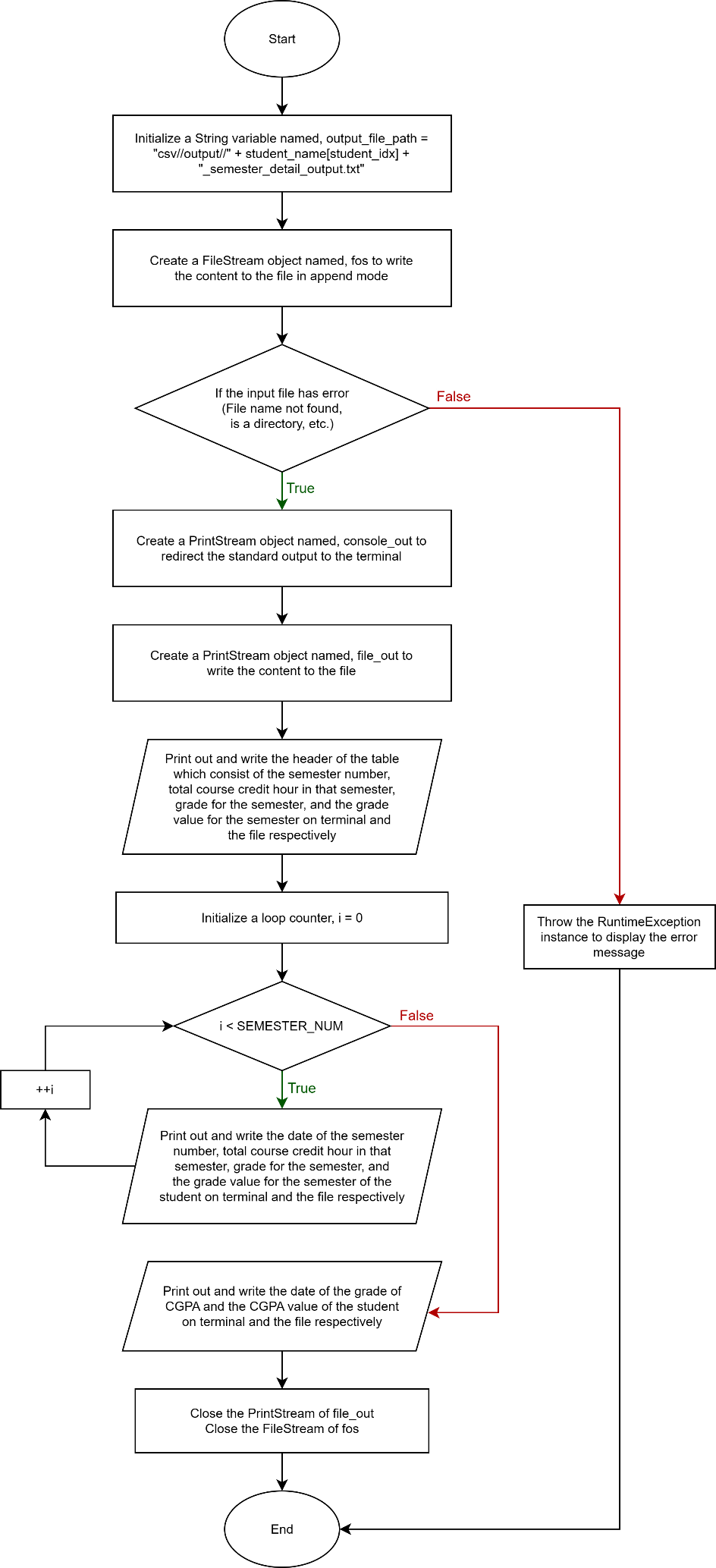
Description automatically generated*

*Figure 13 Flowchart for generate\_grade()*

A screenshot of a computer screen

Description automatically generated

*Figure 14 Flowchart for print\_result()*



*Figure 15 Flowchart for print\_summary\_result()*

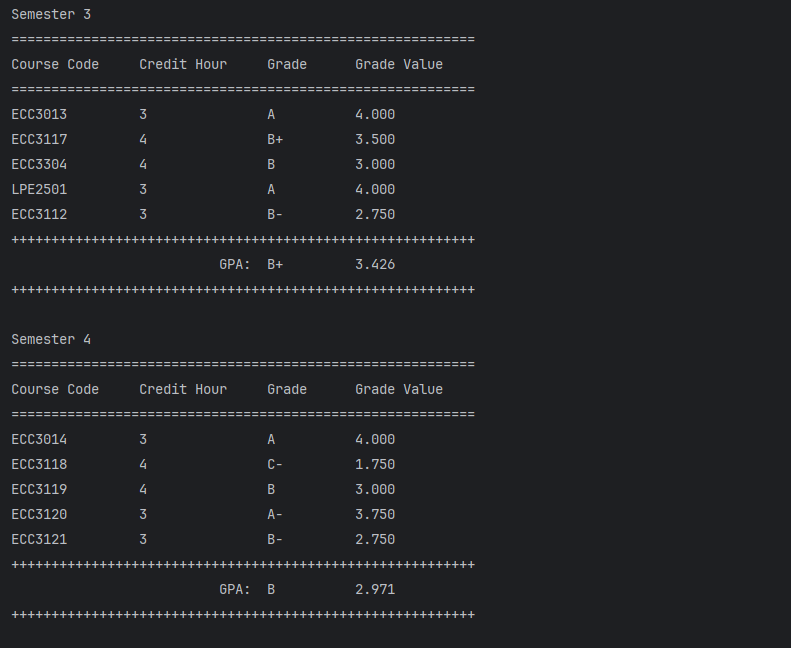
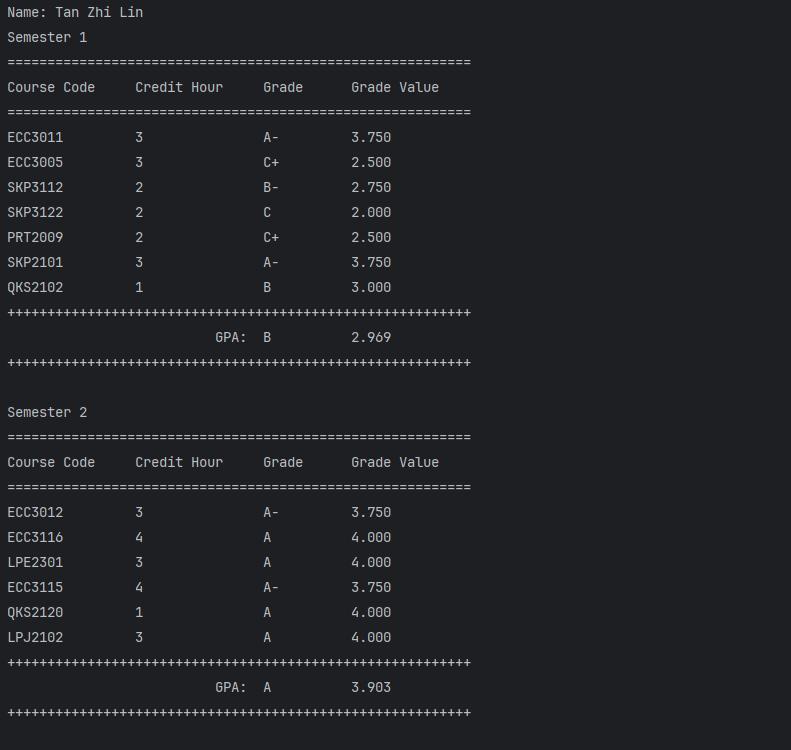
To summarize the function flowcharts stated in the figures shown above, Table 2 had stated the function name, function return type, function arguments, and the description of function.

*Table 2 Summary of functions*

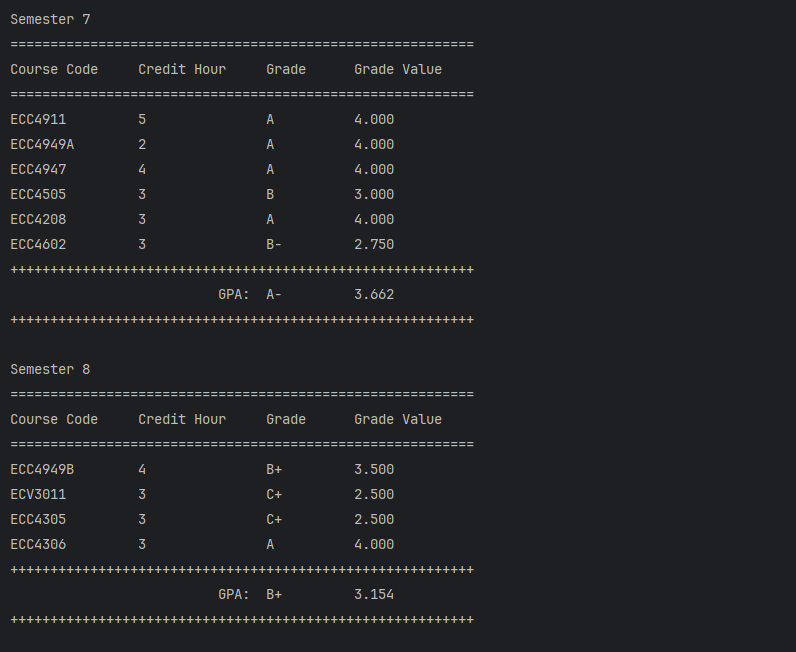
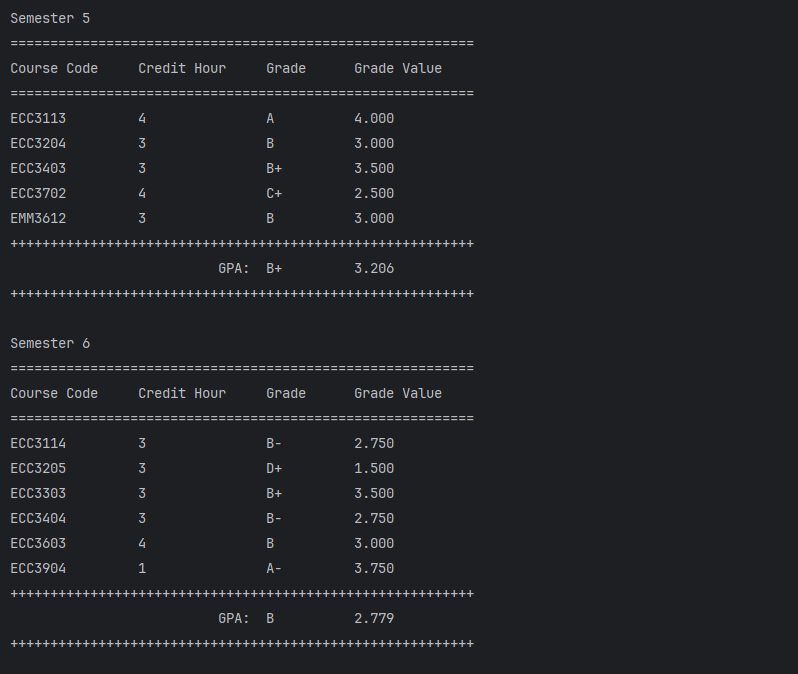
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Return type** | **Arguments** | **Description** |
| parse\_course\_info | void | String course\_info\_csv\_path | Extract the information of course code and course credit hour into 2 separate arrays |
| parse\_student\_result | void | String student\_result\_csv\_path | Extract the information of student name and the result of courses in all semesters |
| final int student\_idx |
| calc\_gpa | void | - | Calculate the GPA of each semester for a student |
| calc\_cgpa | void | Int student\_idx | Calculate the CGPA of all semesters for a student |
| check\_file\_path | boolean | String course\_info\_file\_path | To check whether the file’s path given is correct |
| String [] student\_result\_file\_path |
| is\_csv\_file | boolean | String file\_path | To check whether a file is CSV file |
| error\_module | void | String code | To handle the error message used to display when expected error is occurred |
| convert\_str\_to\_int | int [] | String [][] str\_array | To convert the String type element to Integer type |
| final int row\_idx |
| generate\_grade | String | float grade\_val | To generate the grade according to the value of result |
| print\_result | void | final int student\_idx | To display and save the table of results for all semesters on terminal and file respectively |
| print\_summary\_result | void | final int student\_idx | To display and save the table of summary results for all semesters on terminal and file respectively |

While developing this project, the ***opencsv*** library had been employed to facilitate the parsing of CSV files. This library has proven to be a valuable tool for efficiently reading and processing data from CSV files, simplifying the integration of external data sources into our program. By leveraging the capabilities of ***opencsv***, the program can streamline the parsing process, making it more robust and reliable. The full source code and repository for this project can be accessed via the GitHub link provided on the first page of this report. The use of ***opencsv*** not only enhances the flexibility of our program but also ensures that CSV data can be effortlessly ingested and utilized as needed.

**Screenshot of Results**



*Figure 16.1 Table result of a student from semester 1 to 4 printed at the terminal*



*Figure 16.2 Table result of a student from semester 5 to 8 printed at the terminal*

A screenshot of a computer

Description automatically generated

*Figure 16.3 Table result of a student summarized  
 from semester 1 to semester 8 printed at the terminal*

A screenshot of a computer

Description automatically generated

*Figure 17* *Table result of a student shown in   
terminal had been written into a text file*

**Source Code**

import com.opencsv.CSVReader;

import com.opencsv.exceptions.CsvValidationException;

import java.io.\*;

import java.util.Objects;

public class CalculateCGPA {

    static String error\_code;

    static String error\_msg;

    static final int SEMESTER\_NUM   = 8;

    static final int MAX\_COURSE\_NUM = 10;

    static final int STUDENT\_NUM    = 5;

    static String [][] course\_code = new String[SEMESTER\_NUM][MAX\_COURSE\_NUM];

    static String [][] course\_credit = new String[SEMESTER\_NUM][MAX\_COURSE\_NUM];

    static float [][] student\_result = new float[SEMESTER\_NUM][MAX\_COURSE\_NUM];

    static String [] student\_name = new String[STUDENT\_NUM];

    static float [] student\_gpa = new float[SEMESTER\_NUM];

    static float [] sum\_grade = new float[SEMESTER\_NUM];

    static int [] semester\_total\_credit = new int[SEMESTER\_NUM];

    static float [] student\_cgpa = new float [STUDENT\_NUM];

    public static void main(String [] args) {

        String course\_info\_file\_path = "csv\\course\_info.csv";

        String [] student\_result\_file\_path = {

                "csv\\zhilin\_result.csv",

                "csv\\yewy\_result.csv",

                "csv\\tabina\_result.csv",

                "csv\\shisilia\_result.csv",

                "csv\\hasif\_result.csv"

        };

        if (check\_file\_path(course\_info\_file\_path, student\_result\_file\_path)) {

            parse\_course\_info(course\_info\_file\_path);

            for (int student\_idx = 0; student\_idx < STUDENT\_NUM; student\_idx++) {

                if (parse\_student\_result(student\_result\_file\_path[student\_idx], student\_idx)) {

                    calc\_gpa();

                    calc\_cgpa(student\_idx);

                    print\_result(student\_idx);

                    print\_summary\_result(student\_idx);

                }

                else

                    break;

            }

        }

        if (error\_code != null)

            System.out.println(error\_msg);

        else

            System.out.println("Success");

    }

    private static boolean is\_csv\_file(String file\_path) {

        boolean status = file\_path != null && file\_path.toLowerCase().endsWith(".csv");

        if (!status)

            error\_module("E001");

        return status;

    }

private static boolean check\_file\_path(String course\_info\_file\_path, String [] student\_result\_file\_path) {

        boolean status = is\_csv\_file(course\_info\_file\_path);

        if (status) {

            for (String file\_path : student\_result\_file\_path) {

                if (!is\_csv\_file(file\_path)) {

                    status = false;

                    break;

                }

            }

        }

        return status;

    }

    private static int [] convert\_str\_to\_int(String [][] str\_array, final int row\_idx) {

        int [] int\_array = new int[str\_array[row\_idx].length];

        // Convert String to int

        for (int i = 0; i < int\_array.length; ++i) {

            String temp\_str = str\_array[row\_idx][i];

            if (temp\_str != null)

                int\_array[i] = Integer.parseInt(temp\_str);

            else

                break;

        }

        return int\_array;

    }

    private static String generate\_grade(float grade\_val) {

        String grade;

        if (grade\_val > 3.750 && grade\_val <= 4.000) {

            grade = "A";

        } else if (grade\_val > 3.500 && grade\_val <= 3.750) {

            grade = "A-";

        } else if (grade\_val > 3.000 && grade\_val <= 3.500) {

            grade = "B+";

        } else if (grade\_val > 2.750 && grade\_val <= 3.00) {

            grade = "B";

        } else if (grade\_val > 2.500 && grade\_val <= 2.750) {

            grade = "B-";

        } else if (grade\_val > 2.000 && grade\_val <= 2.500) {

            grade = "C+";

        } else if (grade\_val > 1.750 && grade\_val <= 2.000) {

            grade = "C";

        } else if (grade\_val > 1.500 && grade\_val <= 1.750) {

            grade = "C-";

        } else if (grade\_val > 1.000 && grade\_val <= 1.500) {

            grade = "D+";

        } else if (grade\_val == 1.000) {

            grade = "D";

        } else {

            grade = "F";

        }

        return grade;

}

    private static void error\_module(String code) {

        error\_code = code;

        if (Objects.equals(error\_code, "E001")) {

            error\_msg = "There is an error file type";

        }

        else if (Objects.equals(error\_code, "E002")) {

            error\_msg = "The data in the student result has invalid data”;

        }

    }

public static void parse\_course\_info(String course\_info\_csv\_path) {

        String [] next\_line;

        String [][] course\_container;

        int semester\_idx = 0;

        int course\_idx = 0;

        int line\_idx = 0;

        try (CSVReader reader = new CSVReader(new FileReader(course\_info\_csv\_path))) {

            try {

                while ((next\_line = reader.readNext()) != null) {

                    // switching the container to store the course code and course credit interleave

                    if (line\_idx % 2 == 0) {

                        course\_container = course\_code;

                    }

                    else {

                        course\_container = course\_credit;

                    }

                    // next\_line is an array of values from the line

                    for (String val : next\_line) {

                        if (val.startsWith("\uFEFF")) {

                            val = val.substring(1); // Remove the BOM

                        }

                        course\_container[semester\_idx][course\_idx] = val;

                        ++course\_idx;

                    }

                    if (line\_idx % 2 != 0)

                        ++semester\_idx;

                    ++line\_idx;

                    course\_idx = 0;

                }

            } catch (IOException | CsvValidationException e) {

                throw new RuntimeException(e);

            }

        } catch (IOException e) {

            throw new RuntimeException(e);

        }

    }

    public static boolean parse\_student\_result(String student\_result\_csv\_path, final int student\_idx) {

        boolean status = true;

        String[] next\_line;

        int semester\_idx = 0;

        int course\_idx = 0;

        try (CSVReader reader = new CSVReader(new FileReader(student\_result\_csv\_path))) {

            try {

                while ((next\_line = reader.readNext()) != null) {

                    // Only the first line has BOM char and contain the student name

                    if (next\_line[0].startsWith("\uFEFF")) {

                        student\_name[student\_idx] = next\_line[0].substring(1);

                        continue;

                    }

                    // next\_line is an array of values from the line

                    for (String val : next\_line) {

                        try {

                            student\_result[semester\_idx][course\_idx] = Float.parseFloat(val);

                            // Modify the point to 0.0 if it is less than 1.000

                            if (student\_result[semester\_idx][course\_idx] < 1.000)

                                student\_result[semester\_idx][course\_idx] = 0;

                        } catch (NumberFormatException e ) {

                            status = false;

                            error\_module("E002");

                        }

                    if (status) {

                        ++semester\_idx;

                        course\_idx = 0;

                    }

                    else

                        break;

                }

            } catch (IOException | CsvValidationException e) {

                throw new RuntimeException(e);

            }

        } catch (IOException e) {

            throw new RuntimeException(e);

        }

        return status;

    }

public static void calc\_gpa() {

        for (int i = 0; i < SEMESTER\_NUM; ++i) {

            int [] credit;

            float temp\_sum\_grade = 0;

            int sum\_credit = 0;

            credit = convert\_str\_to\_int(course\_credit, i);

            for (int j = 0; j < student\_result[i].length; ++j) {

                temp\_sum\_grade += credit[j] \* student\_result[i][j];

                sum\_credit += credit[j];

            }

            sum\_grade[i] = temp\_sum\_grade;

            student\_gpa[i] =  temp\_sum\_grade / sum\_credit;

            semester\_total\_credit[i] = sum\_credit;

        }

    }

    public static void calc\_cgpa(final int student\_idx) {

        float sum\_grade\_result = 0;

        int sum\_credit\_result = 0;

        for (int i = 0; i < SEMESTER\_NUM; i++) {

            sum\_grade\_result += sum\_grade[i];

            sum\_credit\_result += semester\_total\_credit[i];

        }

        student\_cgpa[student\_idx] = sum\_grade\_result / sum\_credit\_result;

    }

    public static void print\_result(final int student\_idx) {

        String output\_file\_path = "csv//output//" + student\_name[student\_idx] + "\_semester\_detail\_output.txt";

        try {

            // Create a FileOutputStream to write to the file

            FileOutputStream fos = new FileOutputStream(output\_file\_path);

            // Redirect standard output to the terminal

            PrintStream console\_out = System.out;

            // Write the content to the file

            PrintStream file\_out = new PrintStream(output\_file\_path);

            System.setOut(file\_out);

            // Display on console

            console\_out.println("Name: " + student\_name[student\_idx]);

            // Print to file

            file\_out.println("Name: " + student\_name[student\_idx]);

            for (int i = 0; i < SEMESTER\_NUM; ++i) {

                // Display on console

                console\_out.println("Semester " + (i + 1));

                console\_out.println("==========================================================");

                console\_out.printf("%-15s %-15s %-10s %-15s%n", "Course Code", "Credit Hour", "Grade", "Grade Value");

                console\_out.println("==========================================================");

                for (int j = 0; j < course\_code.length; ++j) {

                    if (course\_code[i][j] != null) {

                        // Display on console

                        console\_out.printf("%-15s %-15s %-10s %.3f%n", course\_code[i][j], course\_credit[i][j], generate\_grade(student\_result[i][j]), student\_result[i][j]);

                        // Print to file

                        file\_out.printf("%-15s %-15s %-10s %.3f%n", course\_code[i][j], course\_credit[i][j], generate\_grade(student\_result[i][j]), student\_result[i][j]);

                    }

                }

                // Display on console

                console\_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");

                console\_out.printf("%-22s    GPA:  %-10s %.3f%n", "" , generate\_grade(student\_gpa[i]), student\_gpa[i]);

                console\_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");

                // Print to file

                file\_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");

                file\_out.printf("%-22s    GPA:  %-10s %.3f%n", "", generate\_grade(student\_gpa[i]), student\_gpa[i]);

                file\_out.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");

            }

            // Close the file of FileStream

            fos.close();

            // Close the file of PrintStream

            file\_out.close();

            // Reset the System.out ro original PrintStream for terminal

            System.setOut(console\_out);

        } catch (IOException e) {

            throw new RuntimeException(e);

        }

    }

    public static void print\_summary\_result(final int student\_idx) {

        String output\_file\_path = "csv//output//" + student\_name[student\_idx] + "\_semester\_detail\_output.txt";

        try {

            // Create a FileOutputStream to write to the file in append mode

            FileOutputStream fos = new FileOutputStream(output\_file\_path, true);

            // Redirect standard output to the file

            PrintStream console\_out = System.out;

            // Write the content to the file

            PrintStream file\_out = new PrintStream(fos);

            // Display on console

            console\_out.println("==============================================");

            console\_out.printf("%-12s %-12s %-10s %-10s%n", "Semester", "Credit Hour", "Grade", "GPA");

            console\_out.println("==============================================");

            // Print to file

            file\_out.println("==============================================");

            file\_out.printf("%-12s %-12s %-10s %-10s%n", "Semester", "Credit Hour", "Grade", "GPA");

            file\_out.println("==============================================");

            for (int i = 0; i < SEMESTER\_NUM; i++) {

                // Display on console

                console\_out.printf("%-12s %-12s %-10s %.3f%n", (i + 1), semester\_total\_credit[i], generate\_grade(student\_gpa[i]), student\_gpa[i]);

                // Print to file

                file\_out.printf("%-12s %-12s %-10s %.3f%n", (i + 1), semester\_total\_credit[i], generate\_grade(student\_gpa[i]), student\_gpa[i]);

            }

**Program Testing**

            // Display on console

            console\_out.println("++++++++++++++++++++++++++++++++++++++++++++++");

            console\_out.printf("%-16s   CGPA:  %-10s %.3f%n", "", generate\_grade(student\_cgpa[student\_idx]), student\_cgpa[student\_idx]);

            console\_out.println("++++++++++++++++++++++++++++++++++++++++++++++\n");

            // Print to file

            file\_out.println("++++++++++++++++++++++++++++++++++++++++++++++");

            file\_out.printf("%-16s   CGPA:  %-10s %.3f%n", "", generate\_grade(student\_cgpa[student\_idx]), student\_cgpa[student\_idx]);

            file\_out.println("++++++++++++++++++++++++++++++++++++++++++++++");

            // Close the file of FileStream

            fos.close();

            // Close the file of PrintStream

            file\_out.close();

            // Reset the System.out ro original PrintStream for terminal

            System.setOut(console\_out);

        } catch (IOException e) {

            throw new RuntimeException(e);

        }

    }

}

In the program, the input is primarily comprised of hard-coded file paths, student information stored in an array of String, and course information stored in a String variable. Proper handling of these inputs, particularly the parsing process, plays a vital role in ensuring the robustness and reliability of the software. The exception handling had been done for the program in 2 situations, which is when the file path given is not CSV file and the program fail to parse a String value to Float value which assigned to error code *E001* and *E002* respectively. When the error is trigged, the program will halt immediately and printing the designed error message onto the terminal to give a clue for user on when the program is halt.

A screenshot of a computer program

Description automatically generated

*Figure 18 Error message displayed on terminal   
when the type of input file is not CSV*

A screenshot of a computer

Description automatically generated

*Figure 19 Error message displayed on terminal when the parser fails to convert the read value from String to Float*