ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Prog102: Procedural Programming		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Nguyen Van Truong	**Student ID**	GCD201597
Class	GCD0905	**Assessor name**	Phan Thanh Tra

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	

Grading grid

P1	P2	P3	M1	M2	D1

❐ Summative Feedback:		❐ Resubmission Feedback:
Grade:	Assessor Signature:	Date:
Lecturer Signature:		

Table of contents

Table of Contents

Table of contents

Table of figures

I. INTRODUCTION (P1, M1)

1. INTRODUCTION TO PROGRAMMING LANGUAGES

2.1. What are programming languages?

Figure 1: Programming Languages [1]

A programming language is a standardized form of computer language that follows a set of rules and consists of a set of instructions that produce various types of output. In computer programming, programming languages are used to build software and implement algorithms. It is a tool that developers use to communicate with computers, thereby creating technology products for human life.

Programming languages have a total of 2 types which are:

- High-level programming languages: Easy to understand and friendly because they are close to human language, so they are easy to maintain and debug but they have poor hardware management and slow program execution time. For example: Python, JavaScript, PHP, ...

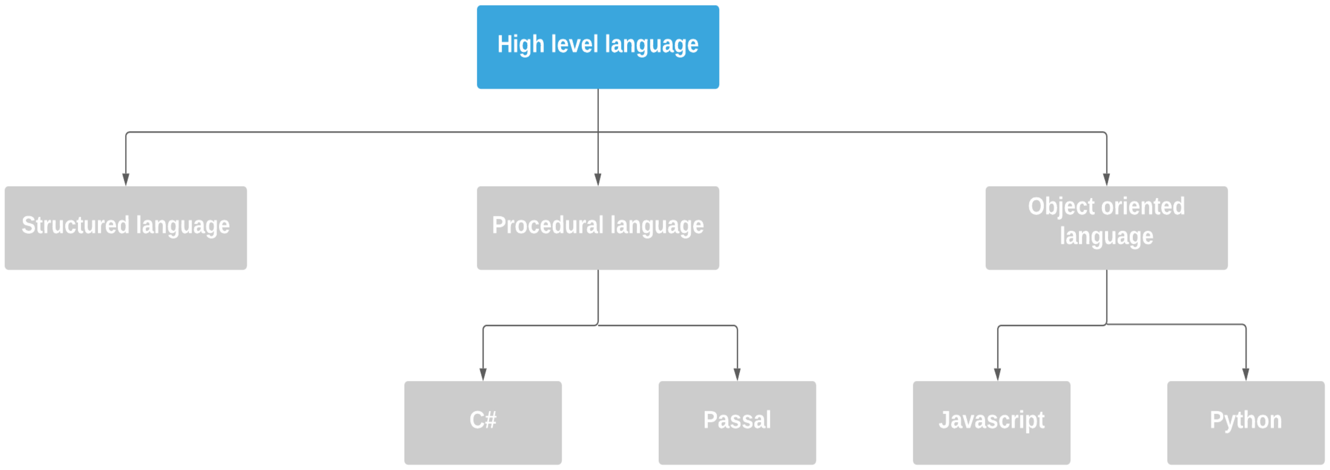


Figure 2: High-level language [2]

- Low-level programming languages: A programming language with a deep relationship to computer hardware. As a result, the computer can quickly capture and perform interactions from the user and can manipulate the server directly. On the contrary, they are difficult to read, debug, and maintain. Such as C, …

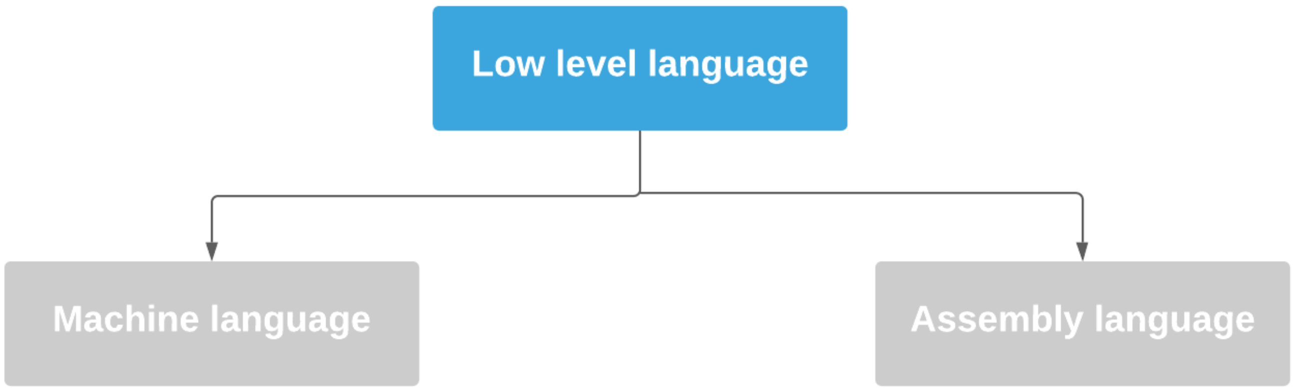


Figure 3: Low-level languages [3]

2.2. Development history of programming languages:

Programming languages have been around since 1883, with the dedication of Ada Lovelace, she wrote the first programming language for computational analysis for mechanical computers. Niklaus Wirth invented the Pascal programming language, which is also used in the Skype application, in 1970. After two centuries, there have been 700 programming languages in existence, with just 50 in popular application today.

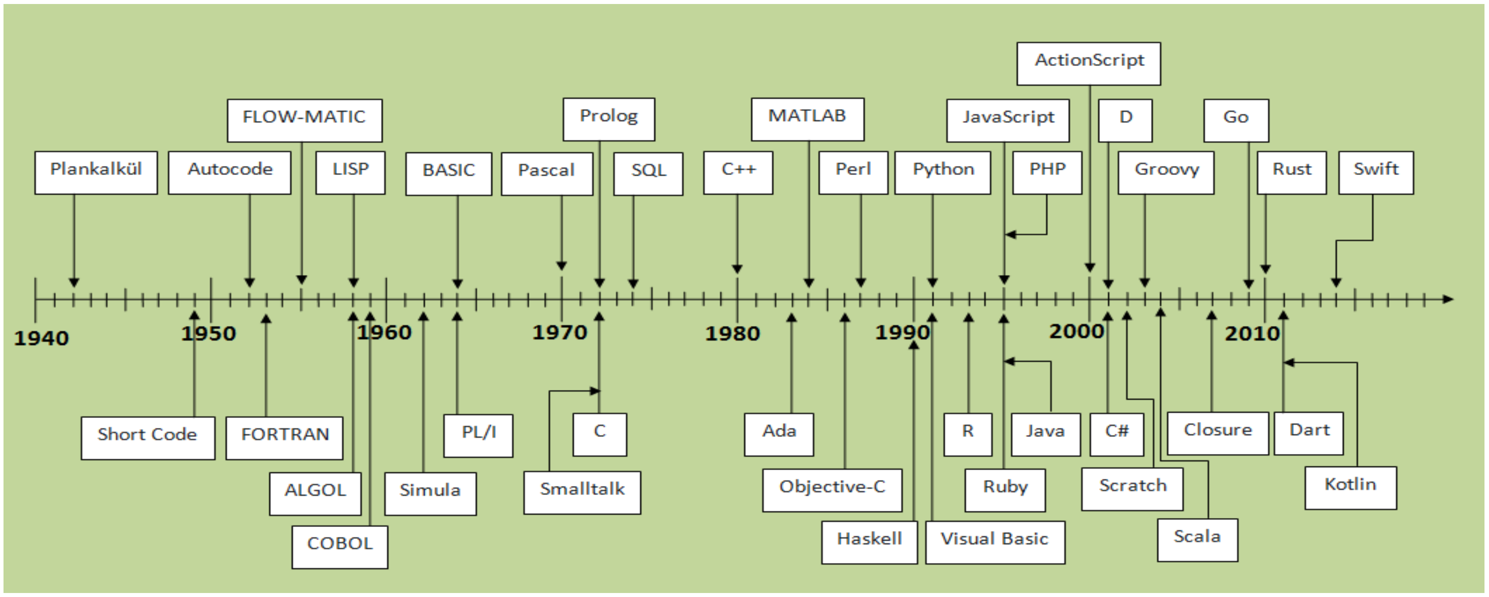


Figure 4: History of programming language [4]

2.3. Visual statistics about programming languages:

Below are specific statistics about the popularity and demand for using the most programming languages according to regularly updated statistics from reputable sources.

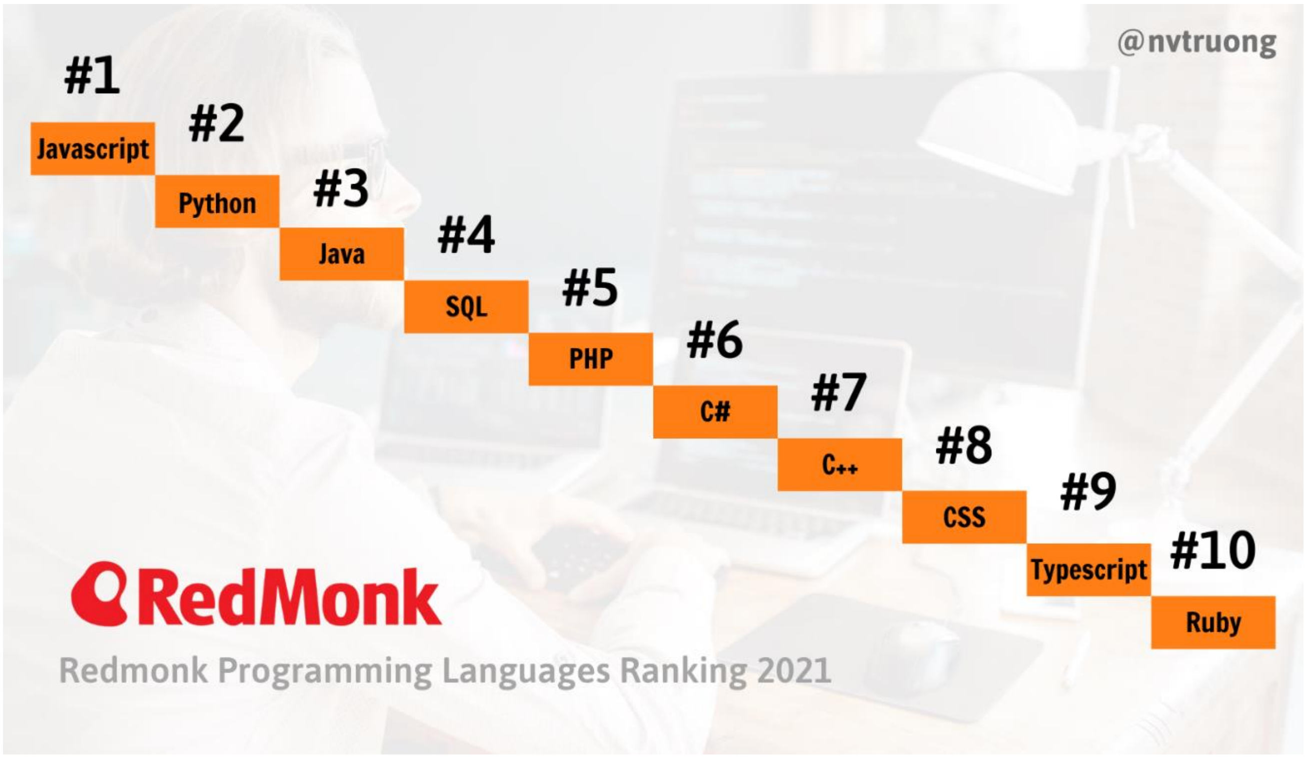


Figure 5: RedMonk programming languages ranking 2021[5]

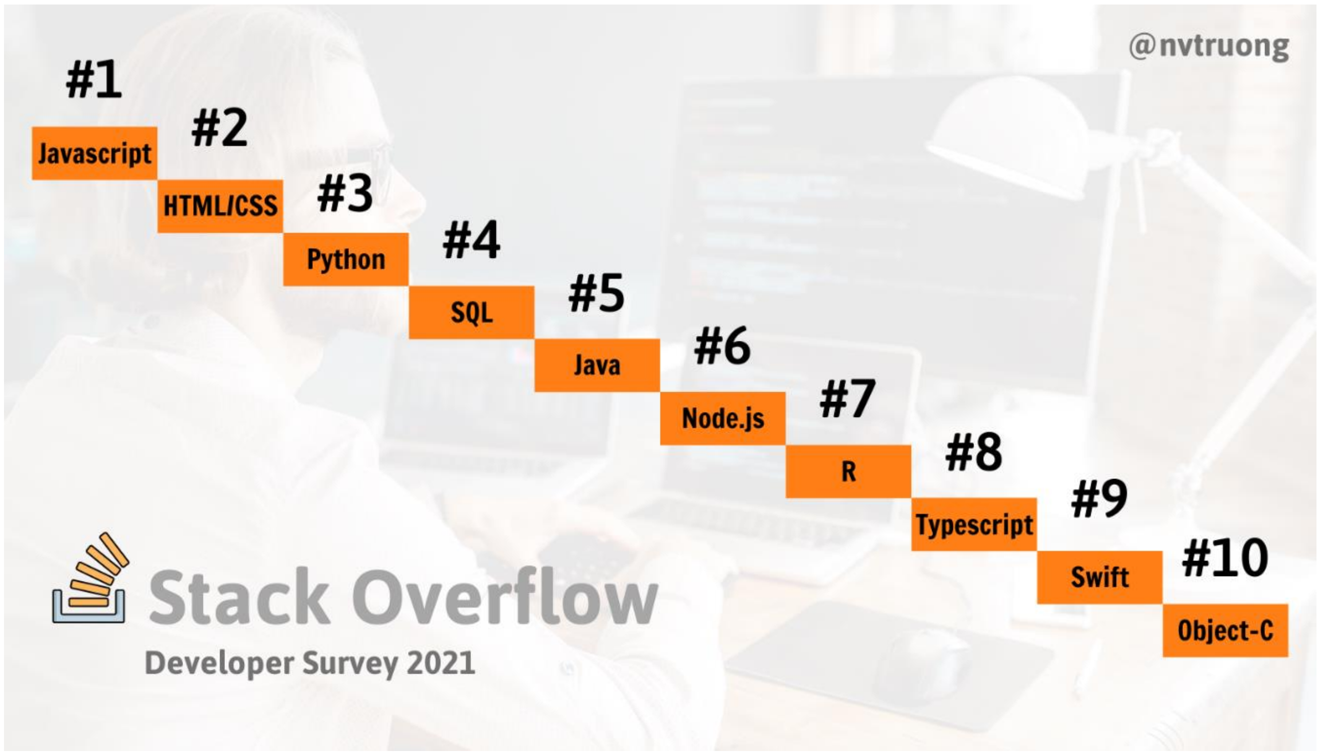


Figure 6: Stack Overflow survey of programming languages of the year 2021 [6]

2. C PROGRAMMING LANGUAGE OVERVIEW:

2.1. C Overview:

C is a general-purpose, imperative programming language that supports structured programming. Use statements to change the state of a program. C makes compact and efficient programs, it has most of the basic control structures and features of modern languages, and it is one of the important programming languages in the present and future...

The C programming language was born at the AT&T BELL lab. Developed by Kernighan and Dennis Ritchie in the early 1970s and completed in 1972. It was created to overcome the problems of languages like B, BCPL, ... And it is used in the system operating UNIX.

Figure 7: C logo [7]

2.2. C - Program Structure:

The specified structure when writing a C program is:

- Do not use keywords to name variables and all keywords must be in lower case.
- Do not use key names to name variables.
- Use curly braces to mark the start and end of a block of code.
- A statement ends with a semicolon.
- Comments are often written to describe the work of a particular command and function or an entire program. Comments start with /* and end with */.

2.3. Advantage and disadvantage of C programming

- **Advantages**:
    - Fast and efficient because the compiled language closely resembles the machine's hardware language.
    - There are a lot of tools and open-source code that make programming easier.
    - we can have direct access to system memory and low-level system functionality.

- It allows storing elements in an array - supports multidimensional arrays. As well as storing the number of elements in it.

- **Disadvantages**:
    - The size in the array must be fixed
    - The memory bytes allocated to the array are sorted sequentially. And we must manage the memory ourselves
    - It's very easy to get in trouble. For example, direct access to memory and pointers.
    - Sometimes the code is longer than in high-level scripting languages like Python, JavaScript. It also has no concept of checking at runtime.

2.3. PROCEDURAL PROGRAMMING OVERVIEW

3.1. What is procedural programming?

The first programming paradigm a developer must learn is procedural programming, also known as subroutines or functions. It is basically a method of building a program from a sequence of commands to be executed step by step to develop an application. It will break down a program into small instructions called procedures or functions.

In procedural programming, there are two data types, global and local. The data contained in the function is called local data and vice versa is called global. Local data has only declarative access, so each function can access its own local data as well as global data. Global data sets the values that are accessed by two or more functions. Help secure and maintain and deploy compilers.

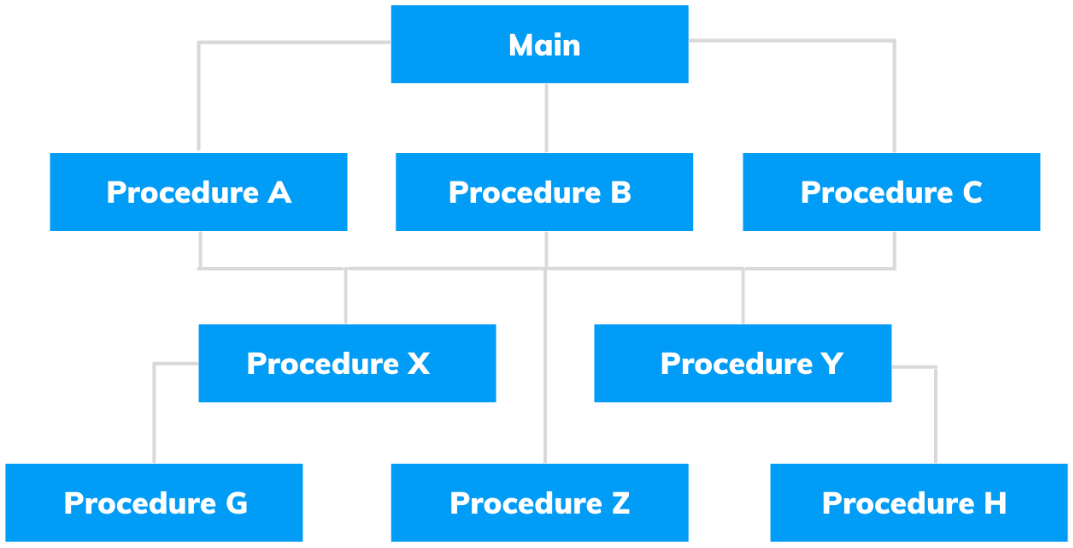


Figure 8: Procedural Programming Paradigms [8]

3.2. Characteristic of Procedural Programming

- A large program is divided into small blocks of code called functions, each of which performs specific tasks: *Help manage and maintain the code easily, avoid code duplication.*
- Procedural programming using top-down methodology.
- Most functions are shared data from global data.
- An appropriate and effective technique for protecting the data of a function from others.

3.3. Feature of Procedural Programming

- **Modularity**: is a software method that demonstrates the separation of functionality into distinct. each of which allows it to accomplish the unique task for which it was created. These all work together as distinct activities to attain a common aim.

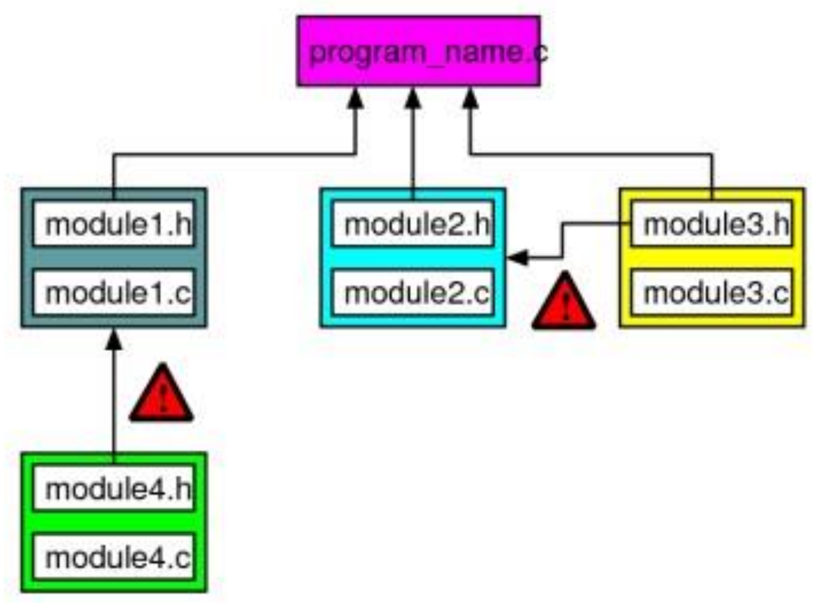
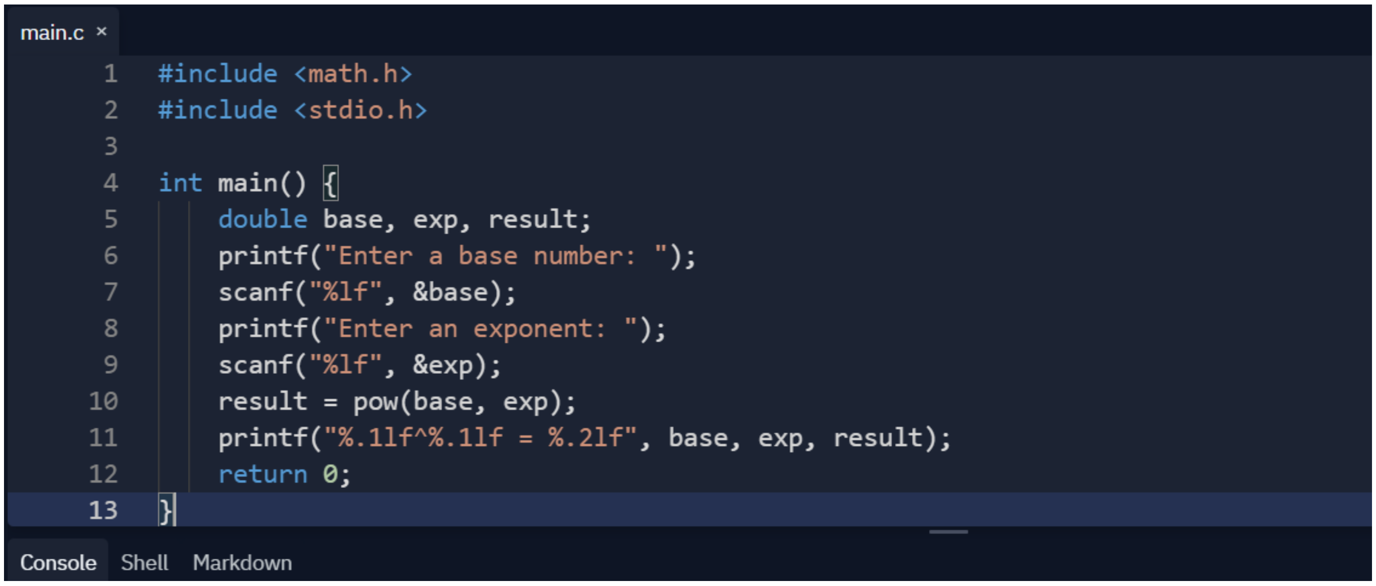


Figure 9: Module example [9]

- **A predefined function**: is often a named instruction. Predefined functions are common in high-level programming languages, although they are obtained from the library or registry rather than the application.



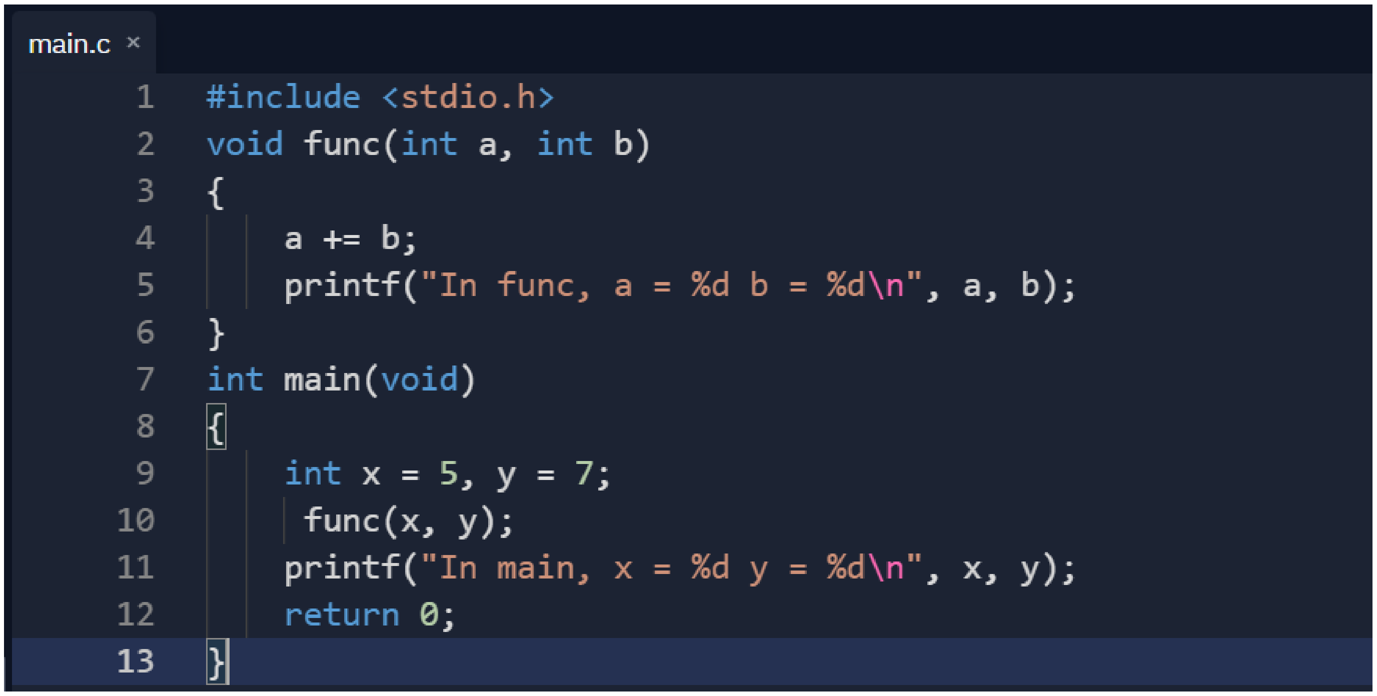
main.c ×

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

int main() {
    double base, exp, result;
    printf("Enter a base number: ");
    scanf("%lf", &base);
    printf("Enter an exponent: ");
    scanf("%lf", &exp);
    result = pow(base, exp);
    printf("%.1lf^%.1lf = %.2lf", base, exp, result);
    return 0;
}
```

Figure 10: A predefined function example [10]

- **Parameter passing:** Parameters are passed to a procedure (subroutine) or function using this technique. Passing the value of the real argument (call by value) or the address of the memory location where the actual parameter is stored are the most frequent approaches (call by reference).

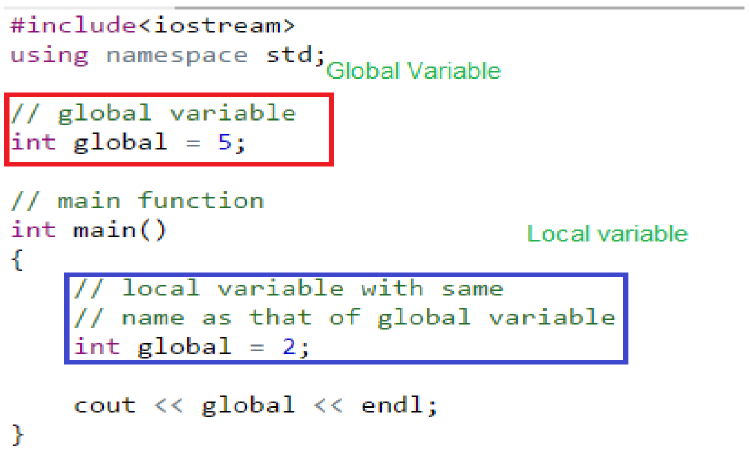


main.c ×

```c
#include <stdio.h>
void func(int a, int b)
{
    a += b;
    printf("In func, a = %d b = %d\n", a, b);
}
int main(void)
{
    int x = 5, y = 7;
     func(x, y);
    printf("In main, x = %d y = %d\n", x, y);
    return 0;
}
```

Figure 11: Parameter passing example [11]

- **Local variables:** are variables that can be accessible just within the single chunk/block of code in which they were written, rather than across the full script of code. A local variable is defined to override a variable with the same name in the wider scope.
- **Global variable:** is a variable that may be read by any other process running in the program, and it can also be accessed by any other task operating in the program. Most of the time, a global variable is a static variable with a scope that spans the whole program's execution.



```
#include<iostream>
using namespace std;

// global variable
int global = 5;

// main function
int main()
{
    // local variable with same
    // name as that of global variable
    int global = 2;

    cout << global << endl;
}
```

Figure 11: Example of global variable & local variable [11]

- **Procedural programming paradigm:** It comes from structured programming. Procedure, also known as procedure or subprocess. Essentially a set of instructions to perform a set of calculations. Procedures can be run at any time in the program, and other procedures can sometimes call another process during its destructive cycle.

3.4. Structure Diagram and Levels:

A hierarchy is a diagram that shows how the tasks of each level are broken down. For example, to develop a function that is responsible for sending row notifications, we need to divide a large problem into small problems as follows.

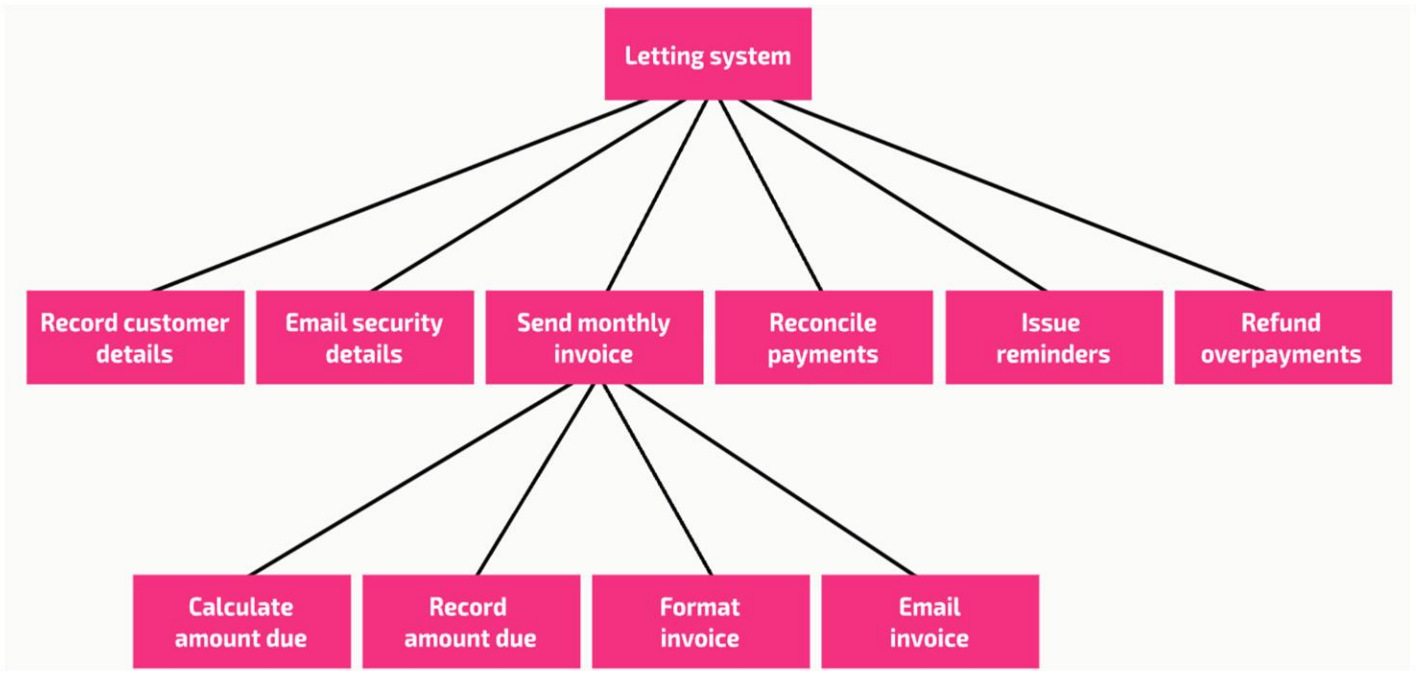


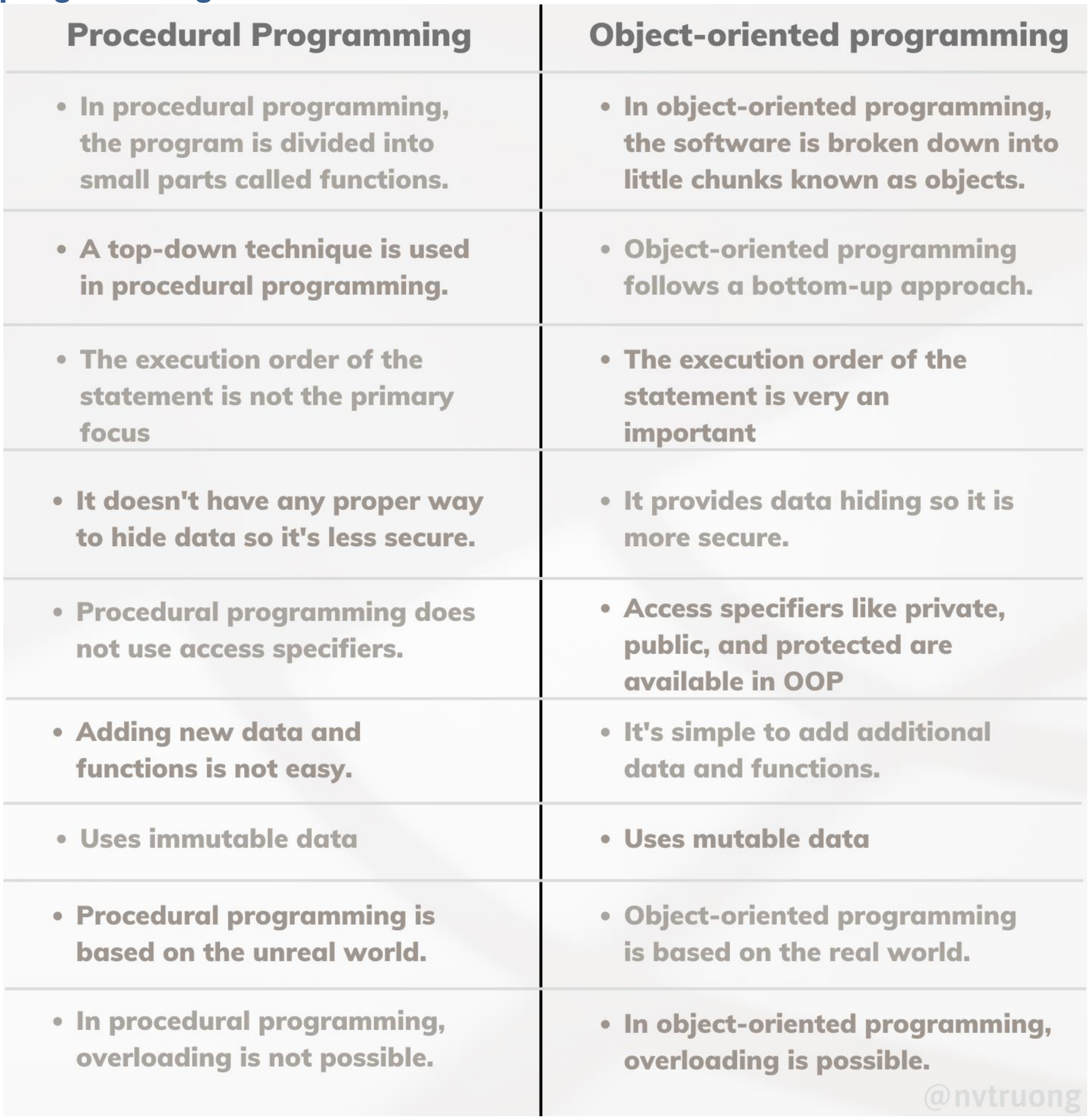
Figure 12: Practical example of procedural programming model [12]

3.5. Advantage and disadvantage of procedural programming:

- **Advantages**:
  - It is a simple language that easily implements compilers and interpreters
  - It uses less memory and thanks to the Procedural Programming approach
  - It uses the code in different places in the program without having to rewrite or copy it
  - We can create the structure of a program based on this method. Produce a cleaner and more maintainable code
  - Since the source code is portable, it may also be used to target another CPU.
- **Disadvantages**:
  - The data is visible to the entire program at once, so they have no security over the data.
  - Difficulty matching real-world objects
  - Since procedural code is usually not reusable, it may require re-creation so they will require reuse in another application.
  - Functional programming is inefficient compared to other programming languages.

- Pure Functional Languages are rarely used in commercial software development.

3.6. Difference between procedural programming & Object-oriented programming:



Procedural Programming	Object-oriented programming
• In procedural programming, the program is divided into small parts called functions.	• In object-oriented programming, the software is broken down into little chunks known as objects.
• A top-down technique is used in procedural programming.	• Object-oriented programming follows a bottom-up approach.
• The execution order of the statement is not the primary focus	• The execution order of the statement is very an important
• It doesn't have any proper way to hide data so it's less secure.	• It provides data hiding so it is more secure.
• Procedural programming does not use access specifiers.	• Access specifiers like private, public, and protected are available in OOP
• Adding new data and functions is not easy.	• It's simple to add additional data and functions.
• Uses immutable data	• Uses mutable data
• Procedural programming is based on the unreal world.	• Object-oriented programming is based on the real world.
• In procedural programming, overloading is not possible.	• In object-oriented programming, overloading is possible.

@nvtruong

Figure 13: Comparison table between procedural programming and OOP [13]

II. ANALYSIS (P2)

1. PROPOSAL:

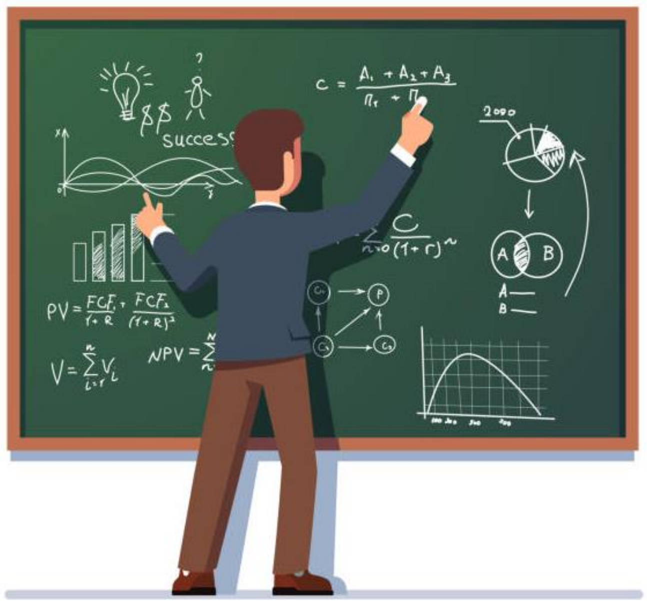


Figure 14: Proposal [14]

A math teacher asks me to help him to write a small application that manages grades of a class and store this information into 2 separate arrays. He needs to print all student's ID with their grades, show the highest grade and lowest grade among all students. Finally, he must determine which student gets the highest and lowest grade. With the parameters listed above, your software should be menu-based. When an option has been completed, the application should return him to the main menu, where he can select another choice. There should be a way to exit the program.

- **REQUIREMENT:**
  - Create a Grade Management App for a Class.
  - Print student information, including ID information, grades, and store information into two separate arrays: integers and real numbers.
  - Find out the students with the highest and lowest scores in the class
  - Software created must be menu-based

- When completing any of the options, the app will automatically return to the home page
- Exit the program.

Solution:

As required by the script, we will use C language to solve the user's request, and we will use variables and data types, conditional statements, loop statements and, functions to solve the problem subject.

2. THE FINALIZATION SECTION:

2.1. Variable:

- In C programming, for storing information in memory. Then we can refer to those pieces of information in memory by using the symbolic variable name, instead of having to use the raw address in memory. Variables can be used to store floating-point numbers, characters, and even pointers to other locations in memory.
- In C, there are some restrictions on variable names. Letters and numbers are used in the name, the first character must be a letter (not a digit). The underscore "_" is treated as a letter.

2.2. Data Types and Sizes:

Every variable in C has a corresponding data type. Each data type has a specific amount of memory, and it can be used to perform many different actions. There are four basic data types in C, their meanings and dimensions are as follows:

Type	Meaning	Size (bytes)	Size (bits)	Format Specifier
\`char\`	a single byte can store one character	1byte	8 bits	%c
\`int\`	An integer	4 bytes	32 bits	%d
\`float\`	a floating-point number with a single precision	4 bytes	32 bits	%f
\`double\`	floating point number with double precision	8 bytes	64 bits	%lf

Below is a list of different data types in C and format specifiers on the 32-bit GCC compiler

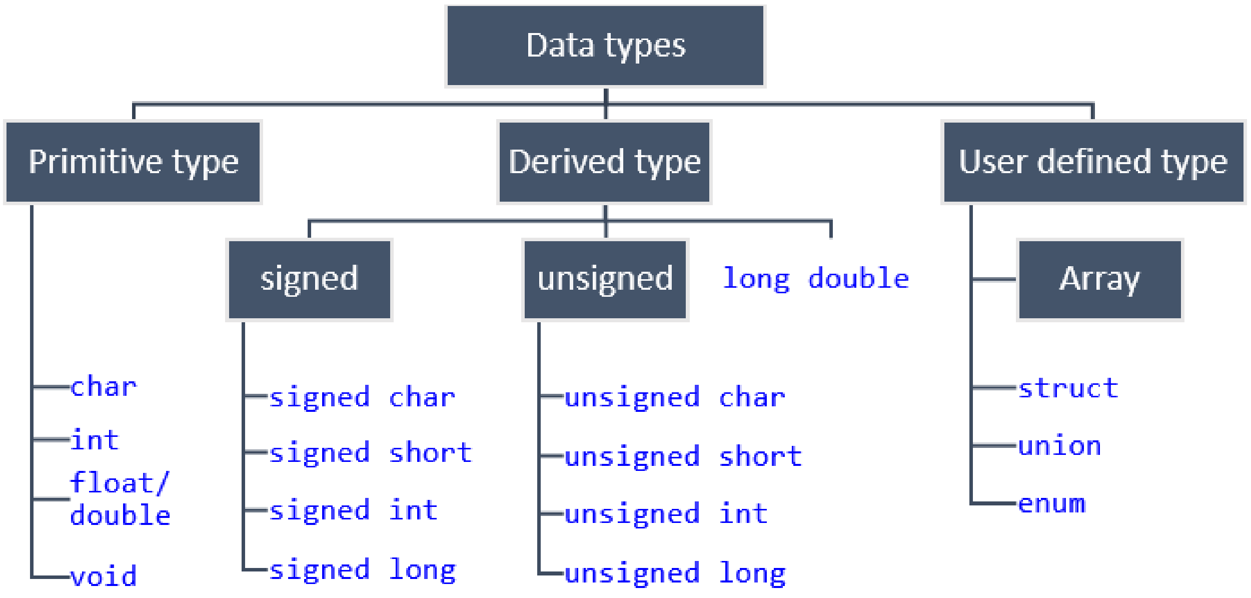


Figure 15: Data types [15]

Solution to the problem: In this problem, we will use float and integer data types to solve the problem. First, we use integers to perform functions such as declaring IDs (including the highest and lowest IDs) or menu selection. Second, float will be used to declare the student's lowest and highest score.

2.3. Conditional Statement:

- **If statement:**
  The if statement is used to examine a given condition and conduct actions based on whether that condition is true. It's most utilized in scenarios when we need to conduct several operations for various circumstances. The if statement's syntax is seen below.

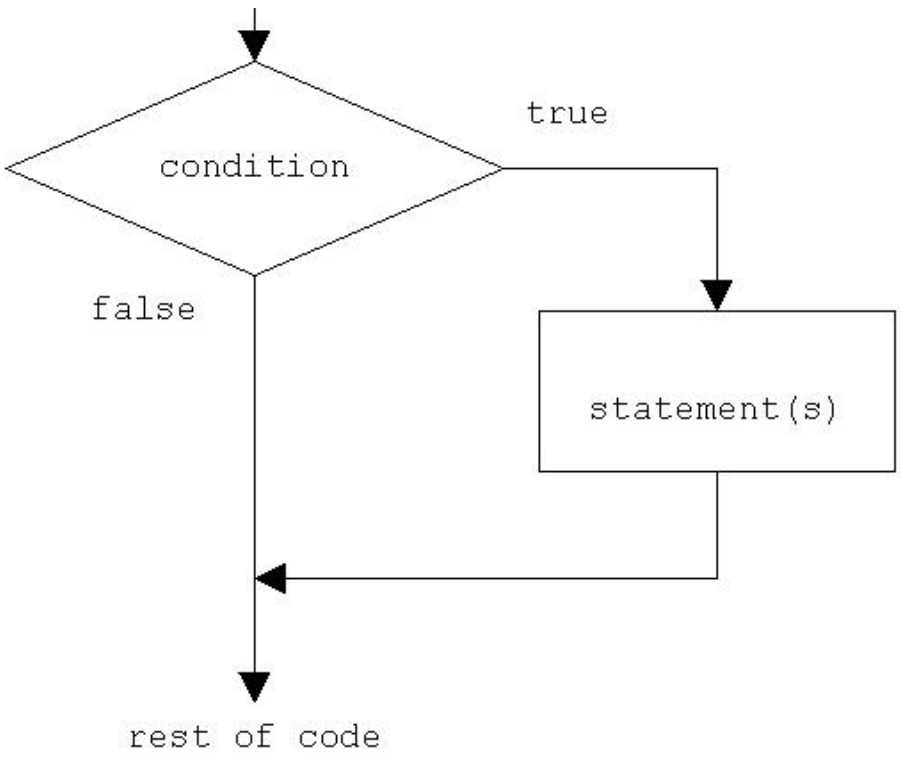


Figure 16: Flowchart of if statement [16]

Program_statements are executed only if **conditional_expression** returns a value other than 0, i.e., it returns TRUE and not FALSE.

- **If…else statement**

The if-else statement is used to execute the code. If the condition in the block returns true or false, the statement is also known as a two-way select statement.

How the if...else statement works:

- *If the expression is non-zero (true), the block statement(s) will be executed.*
- *Otherwise, the else block statement(s) is executed if the expression evaluates to 0 (false).*

Solution to the problem: We use the if-else statement to find the student with the highest and lowest scores. The most important thing is to use the compare condition command to find the student with the highest or lowest score, for example, studentGrade[i] > highestStudentGrade. If they are higher then I will print out the highest student's score and their ID, then we continue to do the search for the student with the highest score. And finding out who has the lowest score is similar.

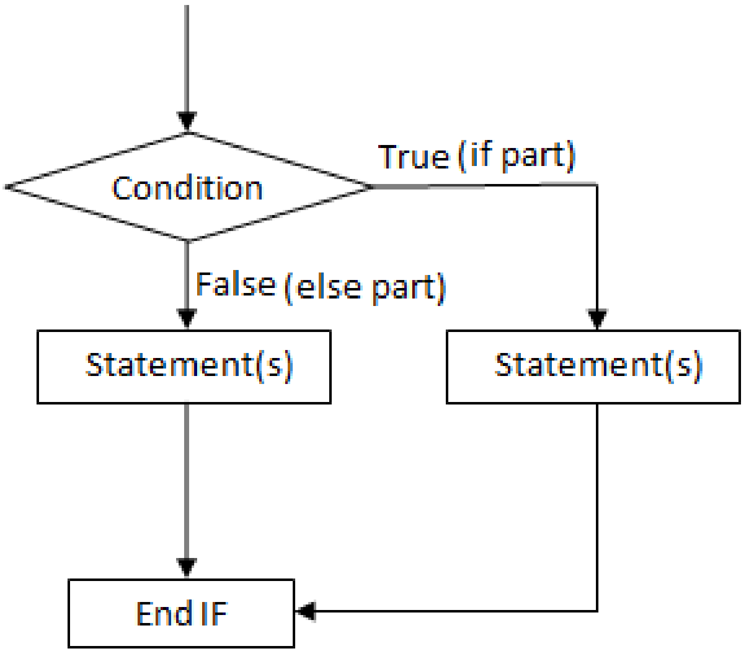


Figure 17: Flowchart of if…else statement [17]

- **Else…if statement**

The else-if statement is used to execute the code in case of multiple conditions. It is called a multi-decision statement and is followed by an else if statement and finally an else statement.

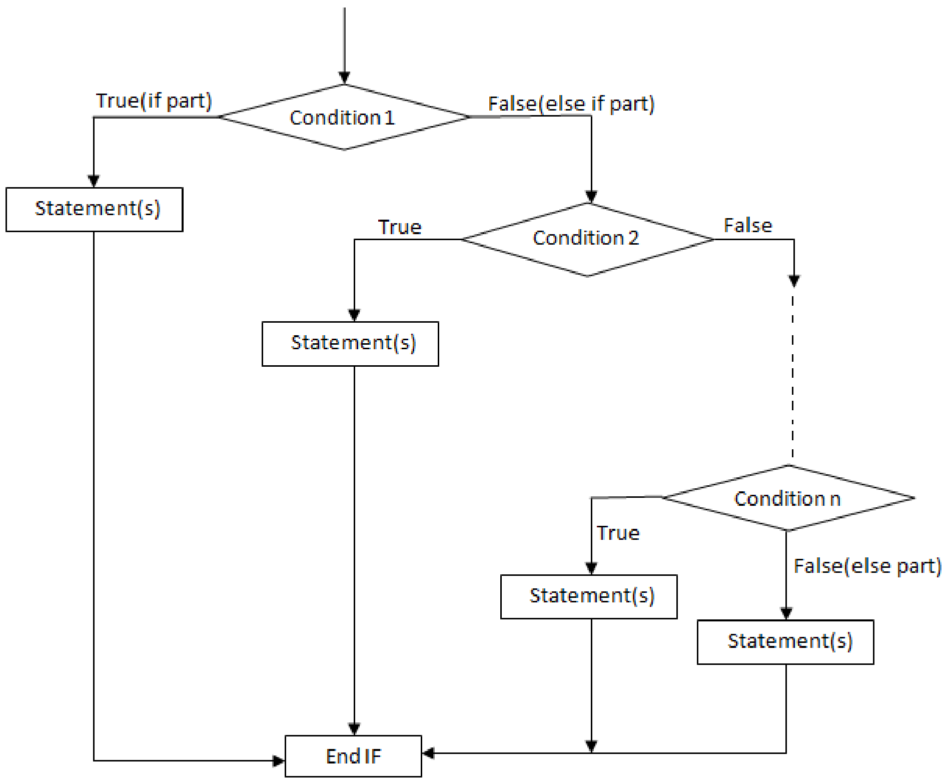


Figure 15: Flowchart of else…if statement [15]

- **Switch case statement**

A switch statement can be used to test a list of conditional instead of a lengthy if-else-if ladder. One or more case labels are evaluated against the switch expression in a statement. When the expression matches a case, the statements connected with that case are executed

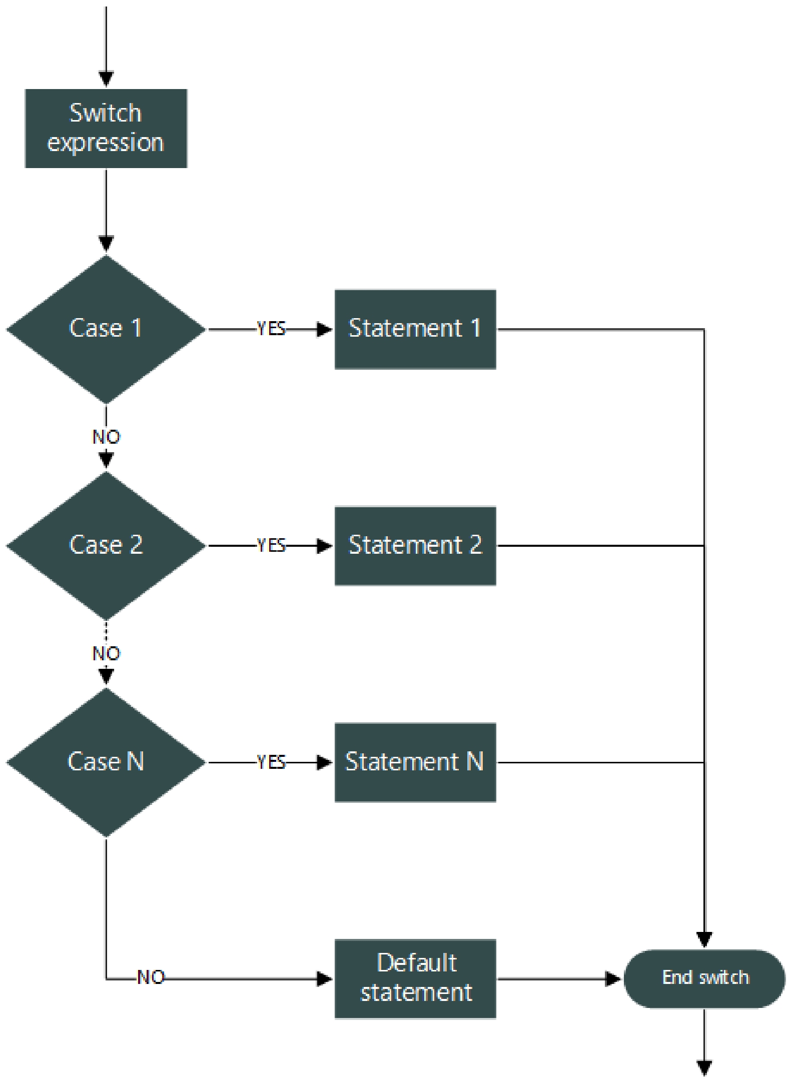


Figure 16: Flowchart of switch case statement [16]

Rules of working:

- Switch...case only works with integral constants, characters, or enumerations.
- We can arrange the cases in whatever order you like. They should, however, be ordered in ascending order. It improves the program's readability.
- Everything in the case must be different. For example, naming case 1 for label 2 is unreasonable
- The break statement is optional. It will stop the statement if there is a condition that satisfies the program.

Solution to the problem: We use a switch-case statement for creating options in the menu, they have 5 options in all and will be executed for each function, if the user enters more than 5 options, the process will exit and end the program.
Specifically, that is:
Function 1: Enter the student information.
Function 2: View all student information.
Function 3: Find out the highest grade.
Function 4: Find out the lowest grade
Function 5: Exit the program.

2.4. Loop Statement:

Repetition reduces complicated issues to simple ones. It enables us to modify the flow of our program so that instead of writing the same piece of code over and again, we may repeat it a finite amount of times.

- For loop:
  The for loop is used in case we need to execute some part of the code until the given condition is satisfied. The for loop is also known as the per-test loop. It is better to use a for loop if the number of iterations is known in advance.

Solution to the problem: We will implement a for loop to perform the entry of grades and student information. As in the article, I can use conditions like i < numberOfStudent to get all the student information as well as find the highest and lowest scores.

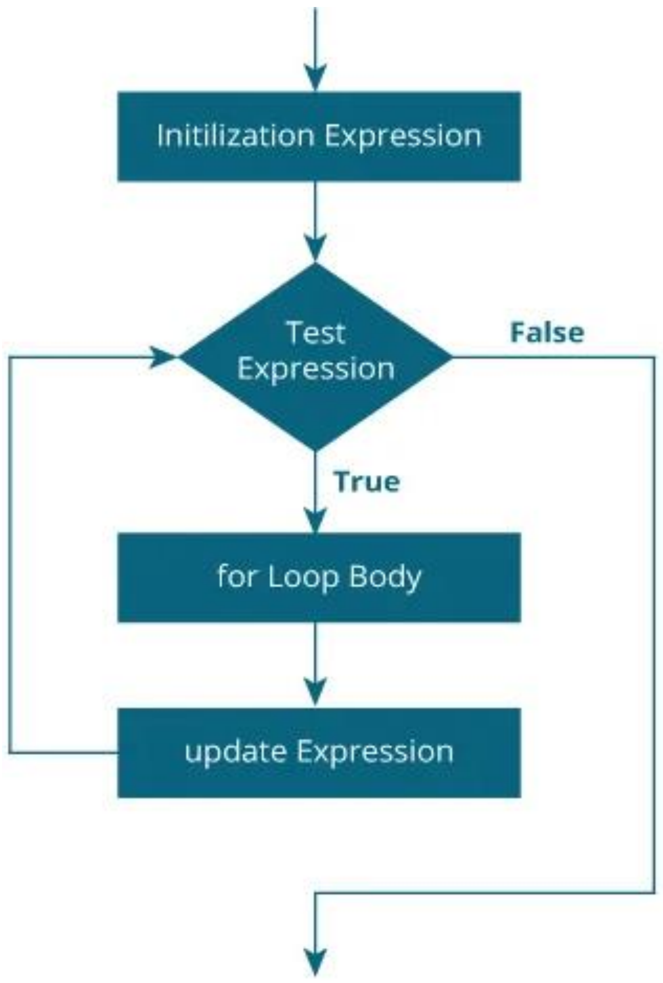


Figure 17: For loop flowchart [17]

How the for loop works:

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to be true, statements inside the body of the for loop are executed, and the update expression is updated.
- Again, the test expression is evaluated.

- While loop:

  While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given Boolean condition. It can be viewed as a repeating if statement. The while loop is mostly used in the case where the number of iterations is not known in advance.

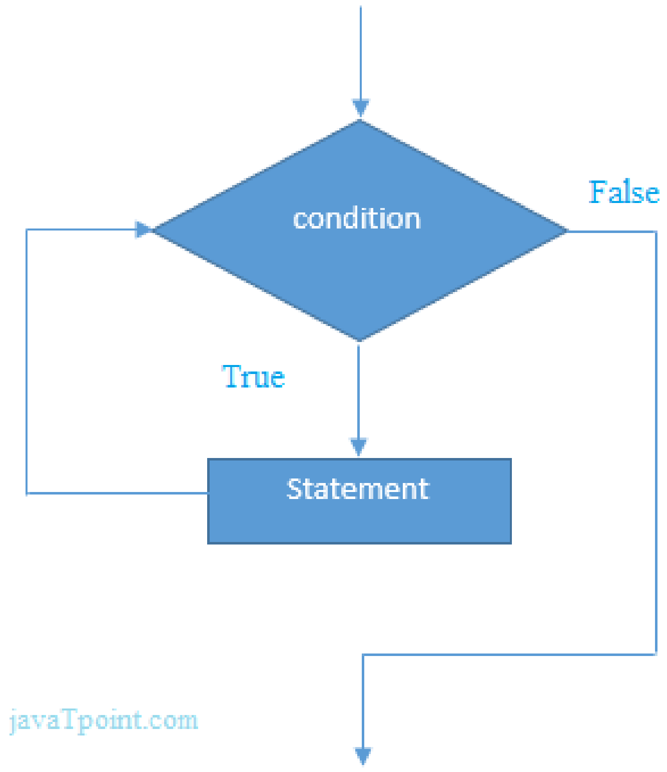


Figure 18: While loop flowchart [18]

How the for loop works: *[3]*

- The while statement first evaluates the expression. If the result is non-zero (true), the while statement executes the statement within its body. This cycle continues until the expression becomes zero (or false).
- The while statement re-checks the expression at the beginning of each iteration before executing the statement. Therefore, at some point, the expression needs to become zero (or false) to end the loop. Otherwise, you'll have an indefinite loop.

  - When the while statement is first encountered and the expression is zero (or false), the while statement won't execute the statement at all. The program passes the control to the statement that follows the while statement.

- Do…while loop:
  The while loop and the do... while loop is quite similar. The do... while loop, on the other hand, runs the body at least once regardless of whether the condition is true or not. Because the do...while loop, after completing the task,

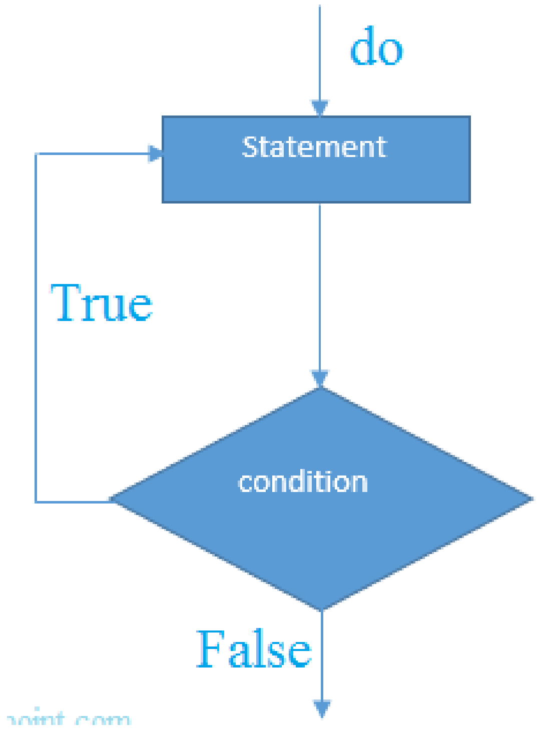


Figure 19: Do-while flowchart [19]

Solution to the problem: We will use a do while loop to create a control flow that executes the block at least once and then iterate to execute that block, these are used in finding the maximum and minimum points, and the result. The result will be determined by the Boolean condition at the end of the block

2.5. Function:

A function is a set of instructions used to process a function; they contain a set of programming statements surrounded by {}. A function can be called multiple times in a C program to provide reusability and modularity

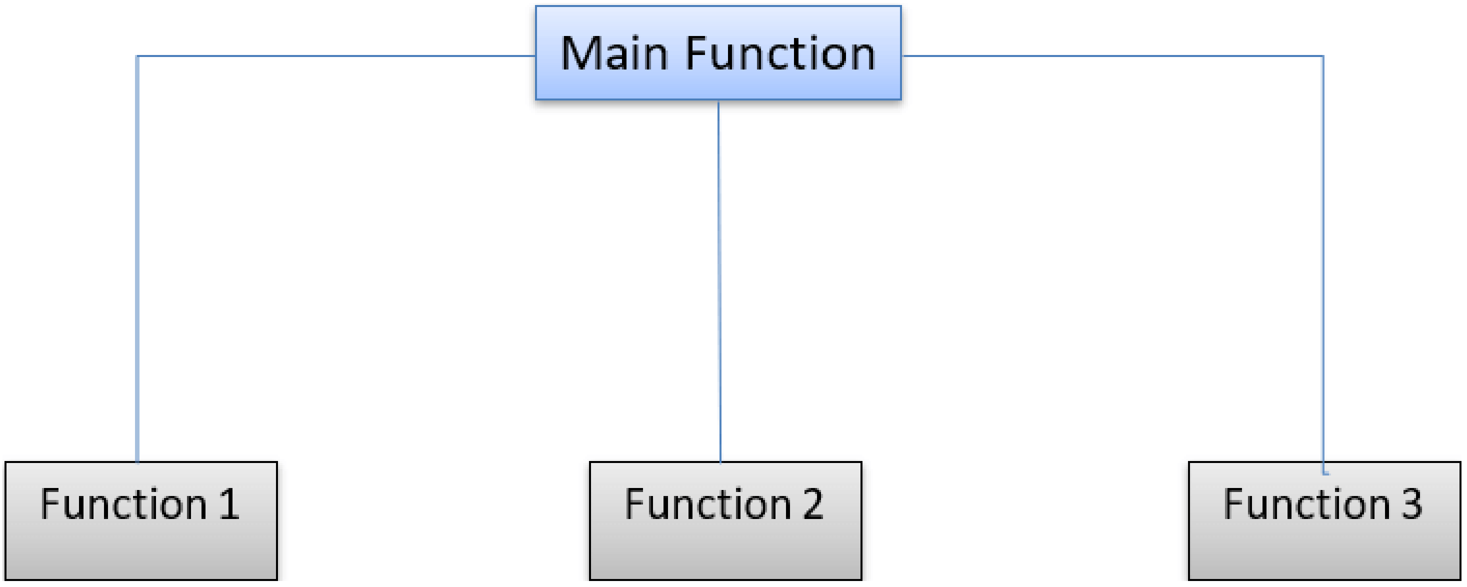


Figure 20: Function [20]

These are all the components of a function that may be found here:

- Function Name - The actual name of the function. The function name and the parameter list together form the signature of the function.
- Parameter - Same as placeholder. When we call a function, we send a value to the argument.
- Function Body - It contains a collection of statements that define the working function
- Return Type - A function will probably return a value within the function.

- Type of function:
    - Standard library functions are functions that are available. Header files specify these functions. Printf(), get(), and scanf() are examples of standard library functions (). For example, printf() is a standard library function for printing formatted output to the screen. They are specified in the header file stdio.h. To utilize the printf() function, we must include the stdio.h header file using #include.
    - User-defined functions: Developed by C programmers, users can reuse these functions many times. Aim to reduce the complexity of a large program and optimize the.

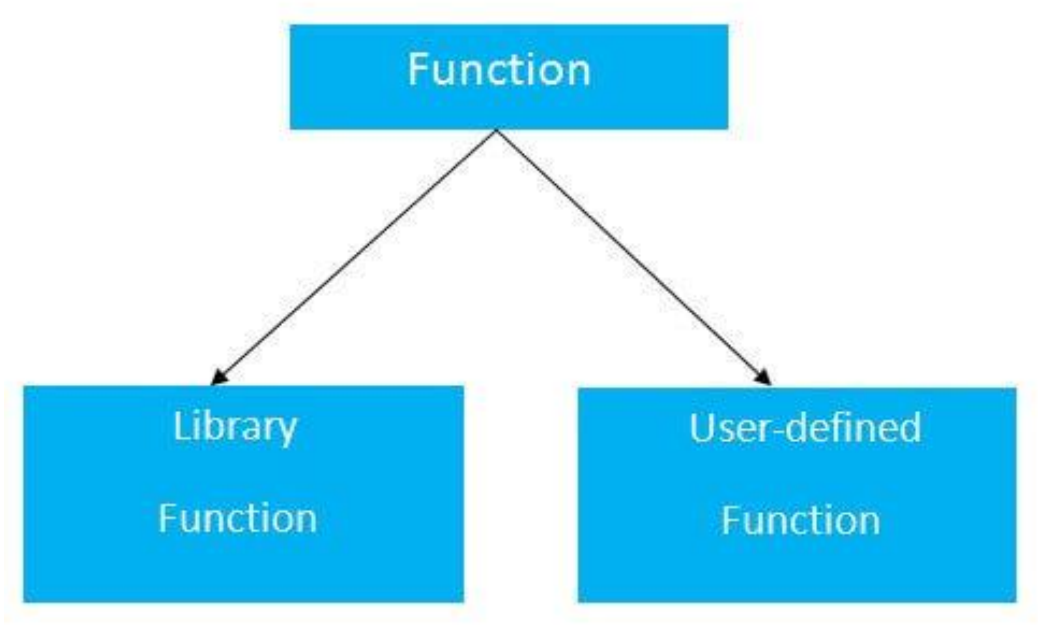


Figure 21: Type of function [21]

- Advantage of functions in C:
    - Use functions, to avoid rewriting the same piece of code in a program
    - We can call functions any number of times and from any location in a program.
    - We can follow a large program easily when it is divided into many parts, functions.

2.6. Array:

An array is a sort of data structure that is used to hold a fixed-length sequence of elements of the same type. Arrays are typically used to hold collections of data, but they may also be used to store a collection of variables of the same type.

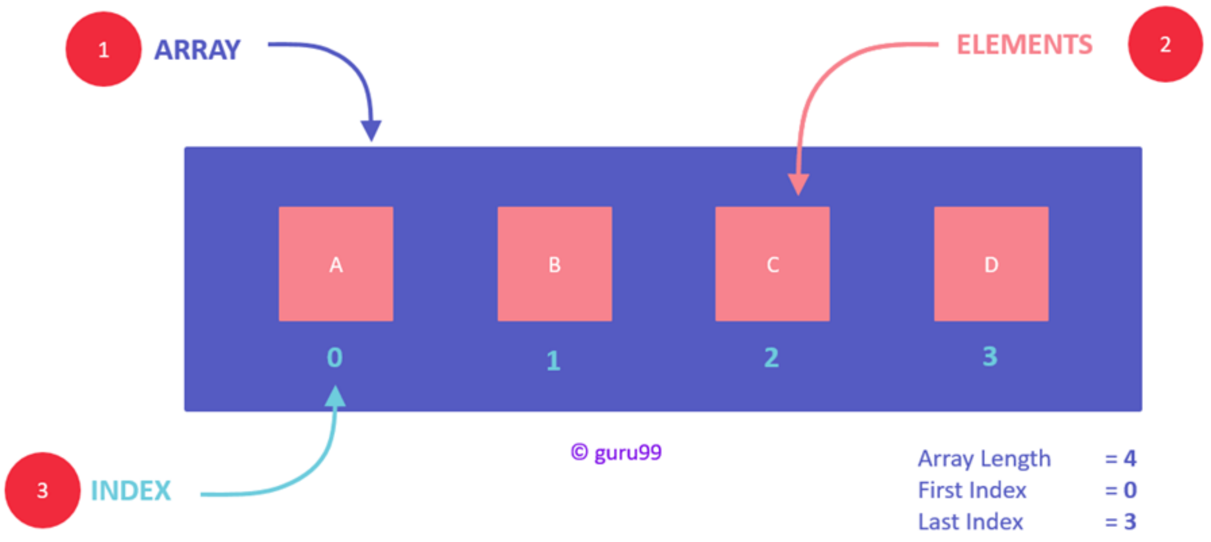


Figure 22: Array [22]

II. DESIGN SYSTEM MODEL (P3, M2)

1. USE CASE:

Use Case is a technique used to describe the interaction between a user and a system, in a particular environment and for a particular purpose. In the use case, the Actor (referring to the user or an external object that interacts with our system) will be the teacher with the role of system administrator and the use cases in the use case diagram will are functions, including 7 functions that have an include relationship (Must-have relationship between Use Cases):

- **Function 1 - View all student information:** The user chooses option 1 to execute the program => The program receives the request => The program displays all the student's information on the screen => The user can view the student's information, for example e.g: Name, age, ID, class... => After finishing the task, the program returns to the main menu.
- **Function 2 – Input student information:** User selects option 2 -> The program shows all information of the option -> User can enter student information, including grades, student ID, or both (-> If either end input is invalid -> the program returns an invalid message and asks the user to re-enter) -> Complete the task -> The program automatically returns to the main screen
- **Function 3 – Student with the highest grade:** choose option 3 -> the program displays the student with the highest score (subject option) -> go back to the main menu.
- **Function 4 – Student with lowest grade:** choose option 4 -> the program displays the student with the lowest score (subject option) -> go back to the main menu.
- **Function 5 – Quit the program:** Exit the program -> end the program.

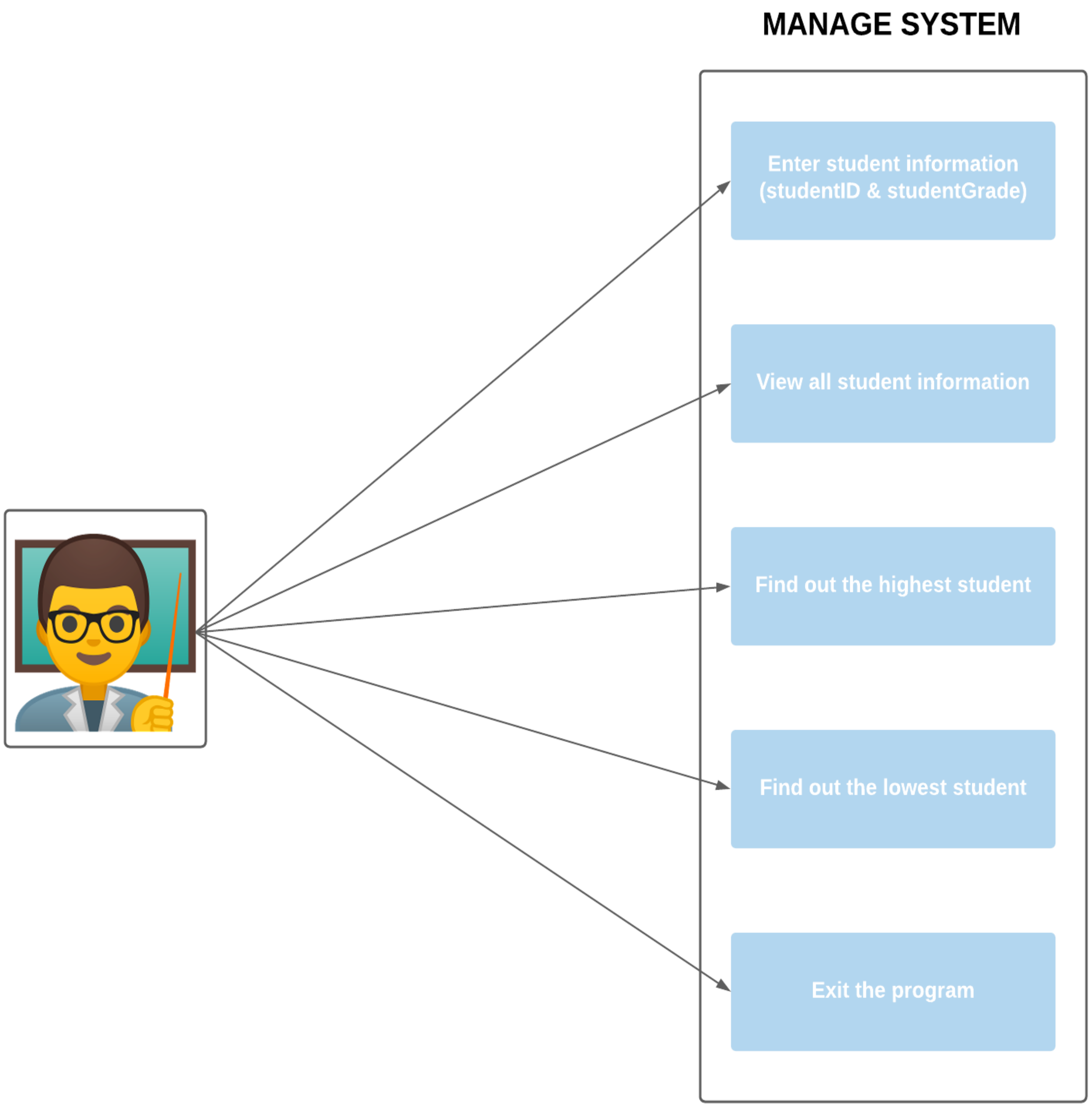

MANAGE SYSTEM
Enter student information (studentID & studentGrade)
View all student information
Find out the highest student
Find out the lowest student
Exit the program

Figure 23: Use Case

2. FLOWCHART:

2.1. View Menu flowchart:

- Step 1: Start:
- Step 2: View menu option.
- Step 3: Read the choice:
  - If option=1, execute the predefined function: “enter student information”, and execute the program.
  - If option=2, execute the predefined function: “view all student information”, and execute the program.
  - If option = 3, execute the predefined function: “find out the highest grade”, and execute the program.
  - If option=4, execute the predefined function: "find out the lowest grade", and execute the program.
  - If option = 5, exit the program.

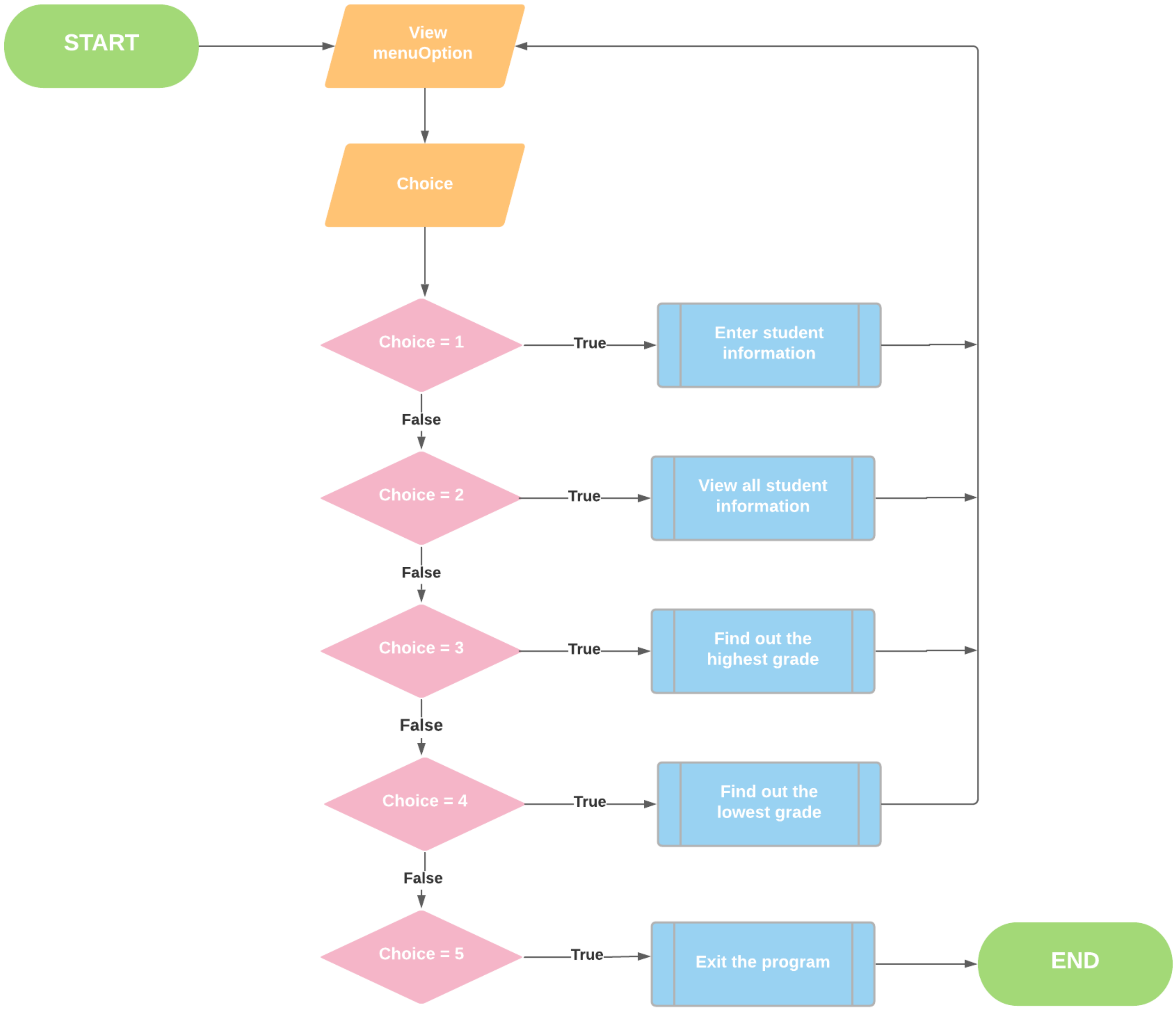


Figure 24: Menu flowchart

2.2. Enter student information flowchart:

- Step 1: Start:
- Step 2: assign the running variable i by 0 representing the first student value and execute the iterator starting from the next student index value: i <- 0.
- Step 3: Check the condition statement, decide whether the running variable i is greater than the numberOfStudent or not: I < numberOfstudent.
    - True: Perform step 4.
    - False: End program.
- Step 4: Enter studentID.
- Step 5: Enter studentGrade
- Step 7: check the condition “studentGrade[i] > 10 || studentGrade[i] < 0”.
    - True: Throws "invalid" and prints "Re-enter".
    - False: Perform step 8.

- Step 8: Increase the running variable i by 1 and go back to step 3 as a do-while loop.

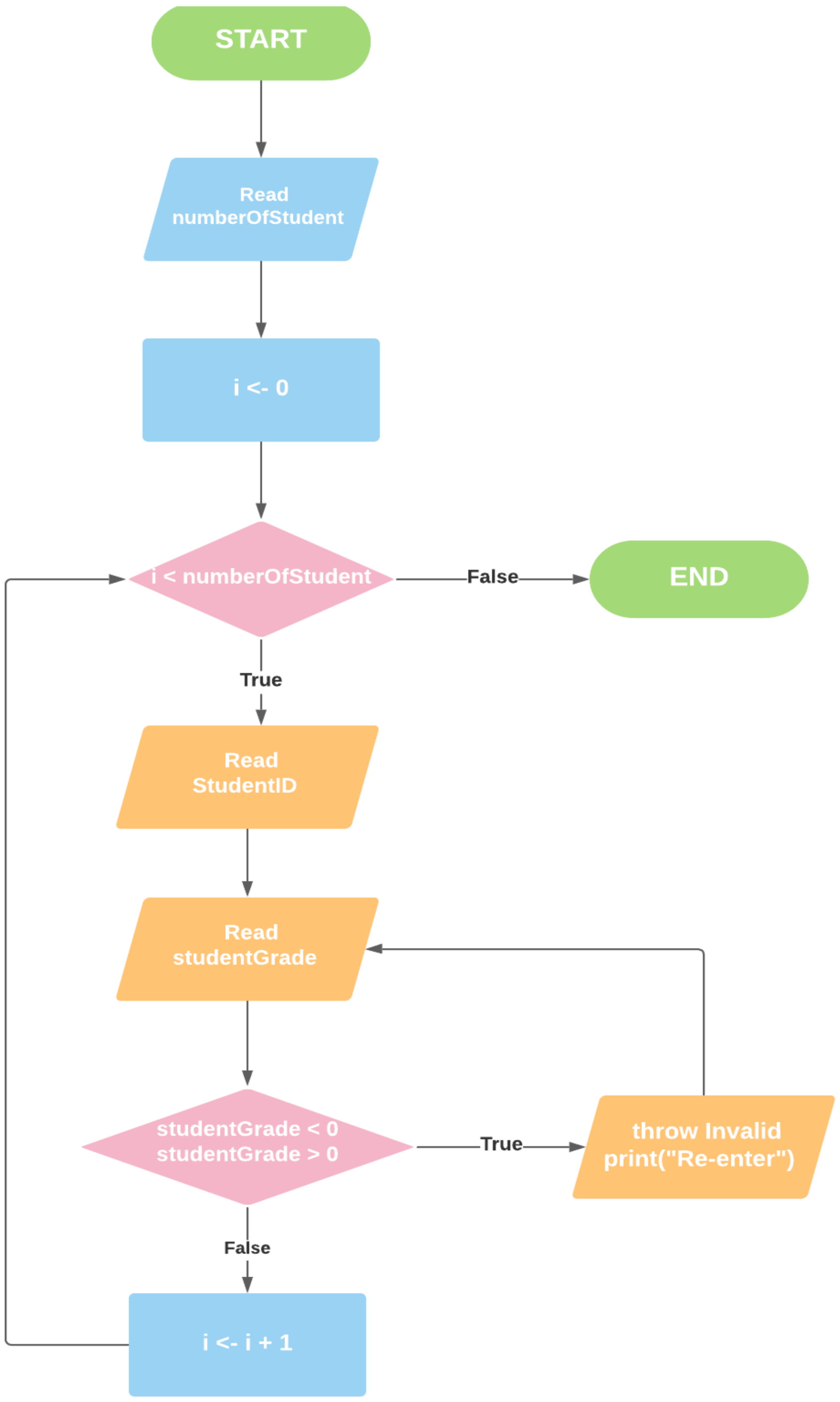


Figure 25: Enter student information student flowchart

2.3. Print student information flowchart:

- Step 1: Start:
- Step 2: Declare and assign i equal to 1: i <- 1.
- Step 3: Check the condition “i <numberOfStudent”.
    - True: Print “studentID[i]&studentGrade[i]
    - False: End the program.
- Step 4: Read studentID.
- Step 5: Read studentGrade.
- Step 6: Increase the iterator i by 1 unit and repeat step 3. The program will loop until the condition is invalid.

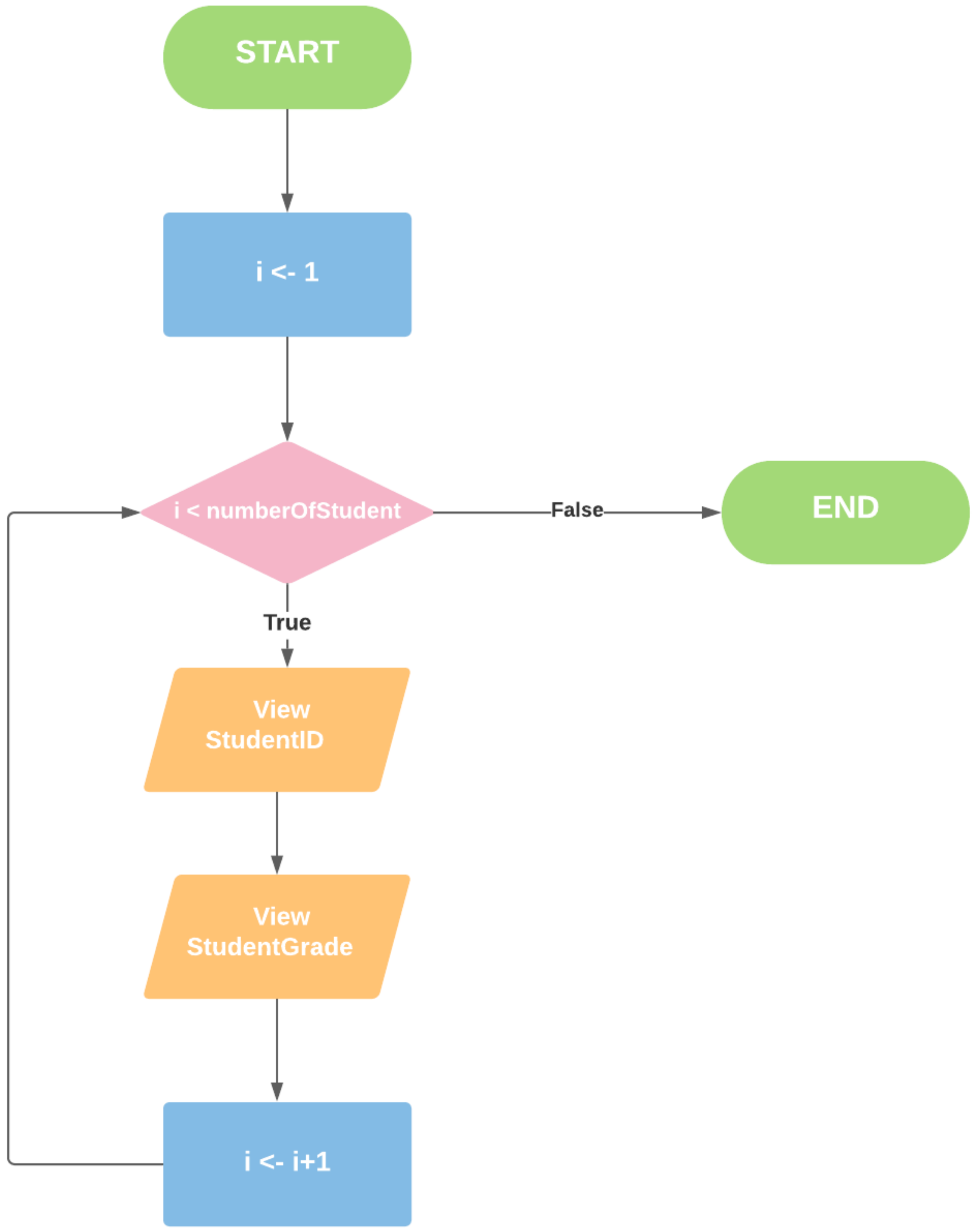


Figure 26: Printing information of student flowchart

3.6. Flowchart of the highest grade:

Explain the algorithm in the flowchart:

- Step 1: Step 1: Start.
  Input: **numberOfStudent & studentGrade & studentID[i]**
- Step 2: assign the variable highestGrade for the value gradeOfStudent[0] representing the first student value to express the first highest grade that the do-while loop has and execute the iterator starting from the next student index value: **highestGrade <- gradeOfStudent[0], i <- 1**.
- Step 3: Check the condition statement, decide whether the running variable i is greater than the number of students or not: **i > numberOfstudent.**
    - TRUE: then end the program: print **highestGrade & studentID[i]**.
    - FALSE: Go to step 4.
- Step 4: Check if the variable studentGrade[i] is smaller than the latest lowestGrade variable: **studentGrade[i] > highestGrade**.
    - TRUE: assign it to be the next value of the lowestGrade: **highestGrade <- gradeOfStudent[i].**
    - FALSE: Increase variable i by 1 and go back to step 3 as a do while loop.

Output: **highestGrade&highestStudentID[i]**

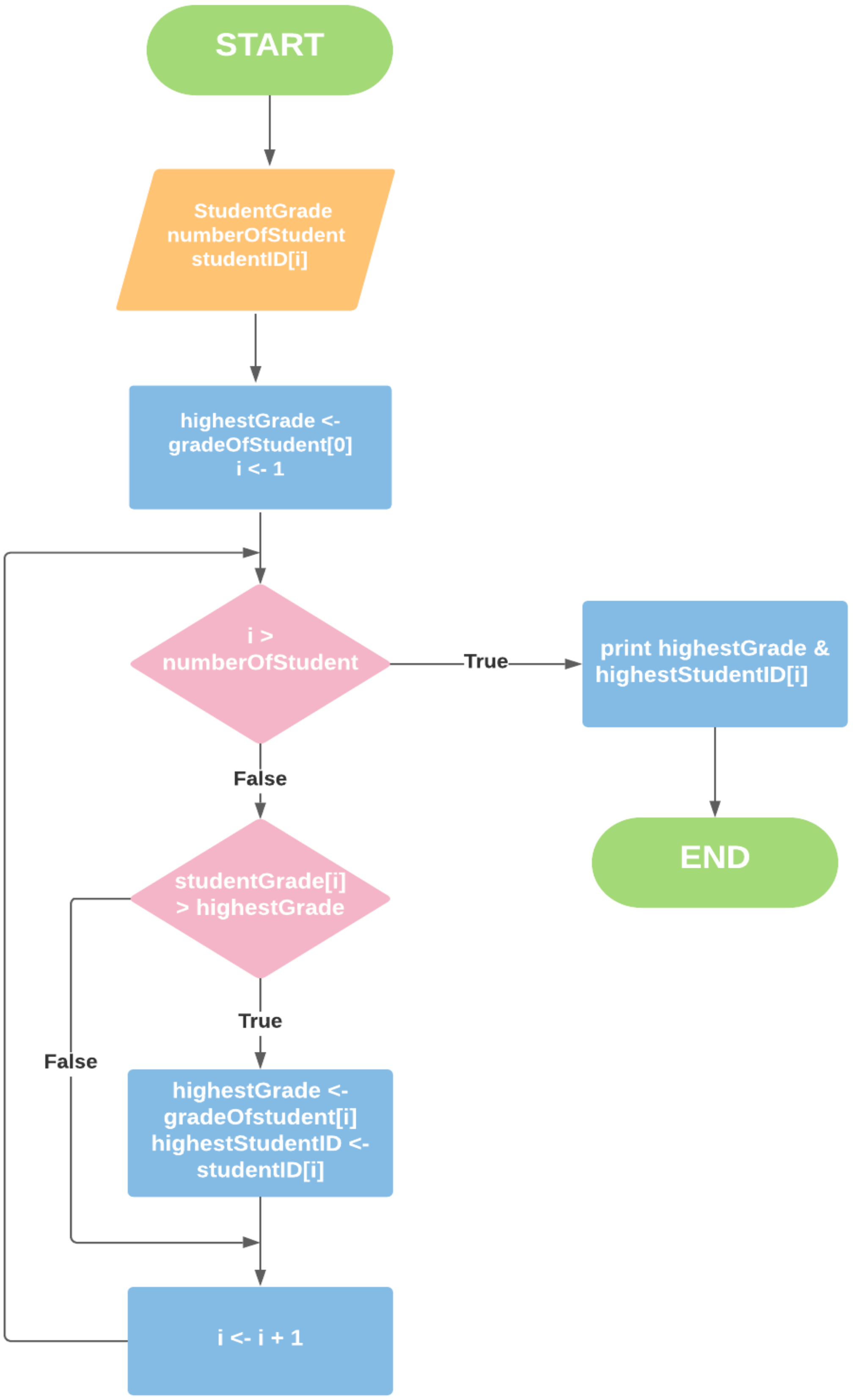


Figure 27: Student with highest grade flowchart

3.5. Flowchart of the lowest grade:

- Step 1: Step 1: Start.
  Input: **numberOfStudent & studentGrade & studentID[i]**
- Step 2: assign the variable lowestGrade for the value gradeOfStudent[0] representing the first student value to express the first highest grade that the do-while loop has and execute the iterator starting from the next student index value: **lowestGrade <- gradeOfStudent[0], i <- 1**.
- Step 3: Check the condition statement, decide whether the running variable i is greater than the number of students or not: **i > numberOfstudent.**
  - TRUE: then end the program: print **lowestGrade & studentID[i]**.
  - FALSE: Go to step 4.
- Step 4: Check if the variable studentGrade[i] is smaller than the latest lowestGrade variable: **studentGrade[i] > lowestGrade**.
  - TRUE: assign it to be the next value of the lowestGrade: **lowestGrade <- gradeOfStudent[i].**
  - FALSE: Increase variable i by 1 and go back to step 3 as a do while loop.

Output: **lowestGrade&lowestStudentID[i]**

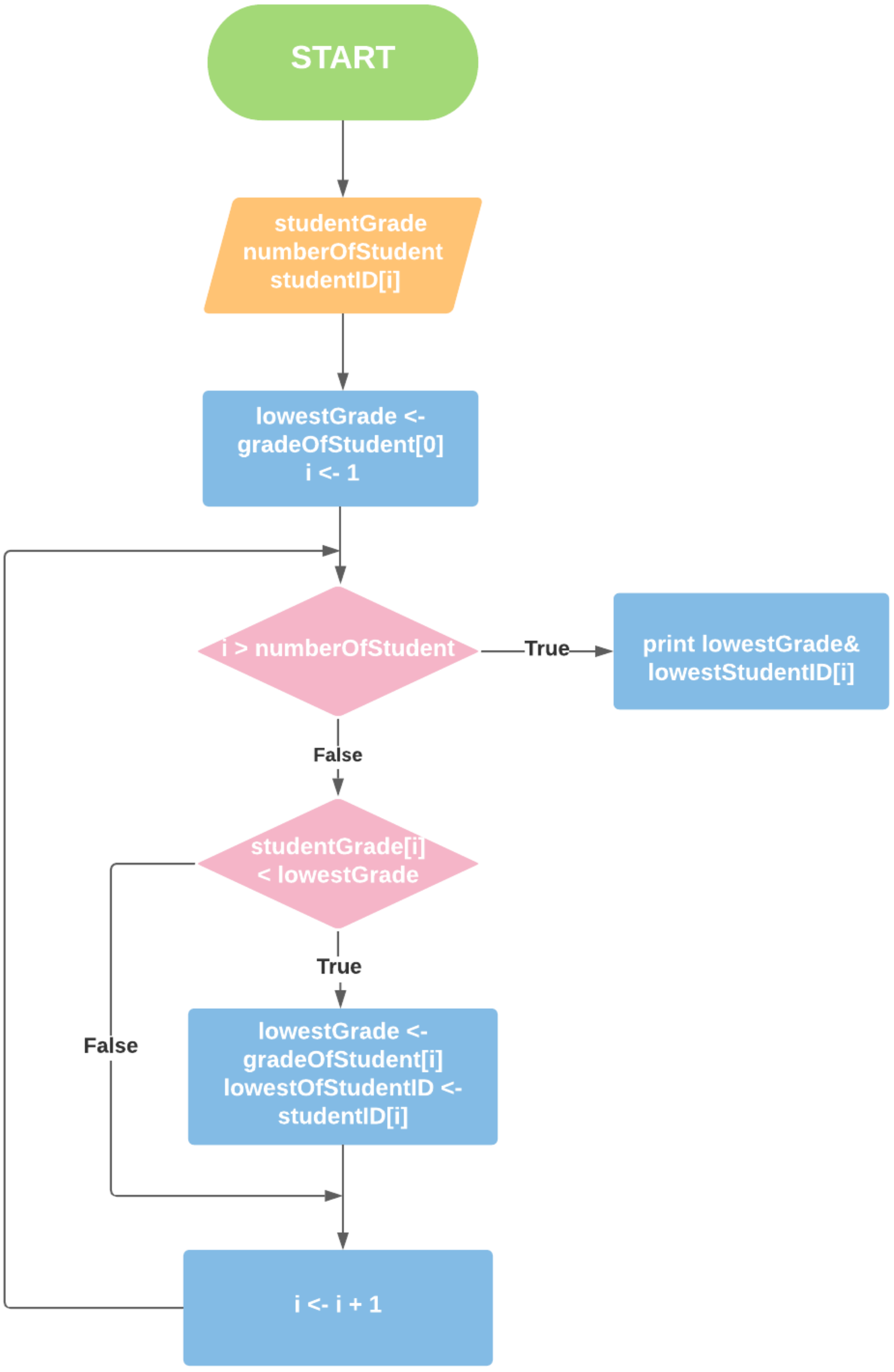


Figure 28: Student with lowest grade flowchart

4.WORK BREAKDOWN STRUCTURE:

4.1. Definition:

WBS is an abbreviation for Work Breakdown Structure. The term work breakdown structure (WBS) refers to the separation and fragmentation of work objects. The goal is to make deployment and control easier and more convenient. Small details must be kept within the scope of the larger project. At the same time, the subtasks might be completely autonomous or completely reliant on each other. Only when all the sub-jobs have been completed is the major task deemed finished.

4.2. Advantages of the Work Breakdown Structure:

- **Create common awareness in work:** Create a work organization that is both basic and effective.
- **Build a schedule for the project:** We can easily identify the time and resources for each task and plan the allocation for those tasks by breaking down the work into smaller chunks.
- **Reduce project risk:** Building a solid WBS will aid in the early detection of difficulties on the project. When a problem, whether good or bad, is also reflected in the WBS, project managers can rapidly assess its influence on the whole process thanks to a well-defined task hierarchy.

4.3. Work Breakdown Structure in the task:

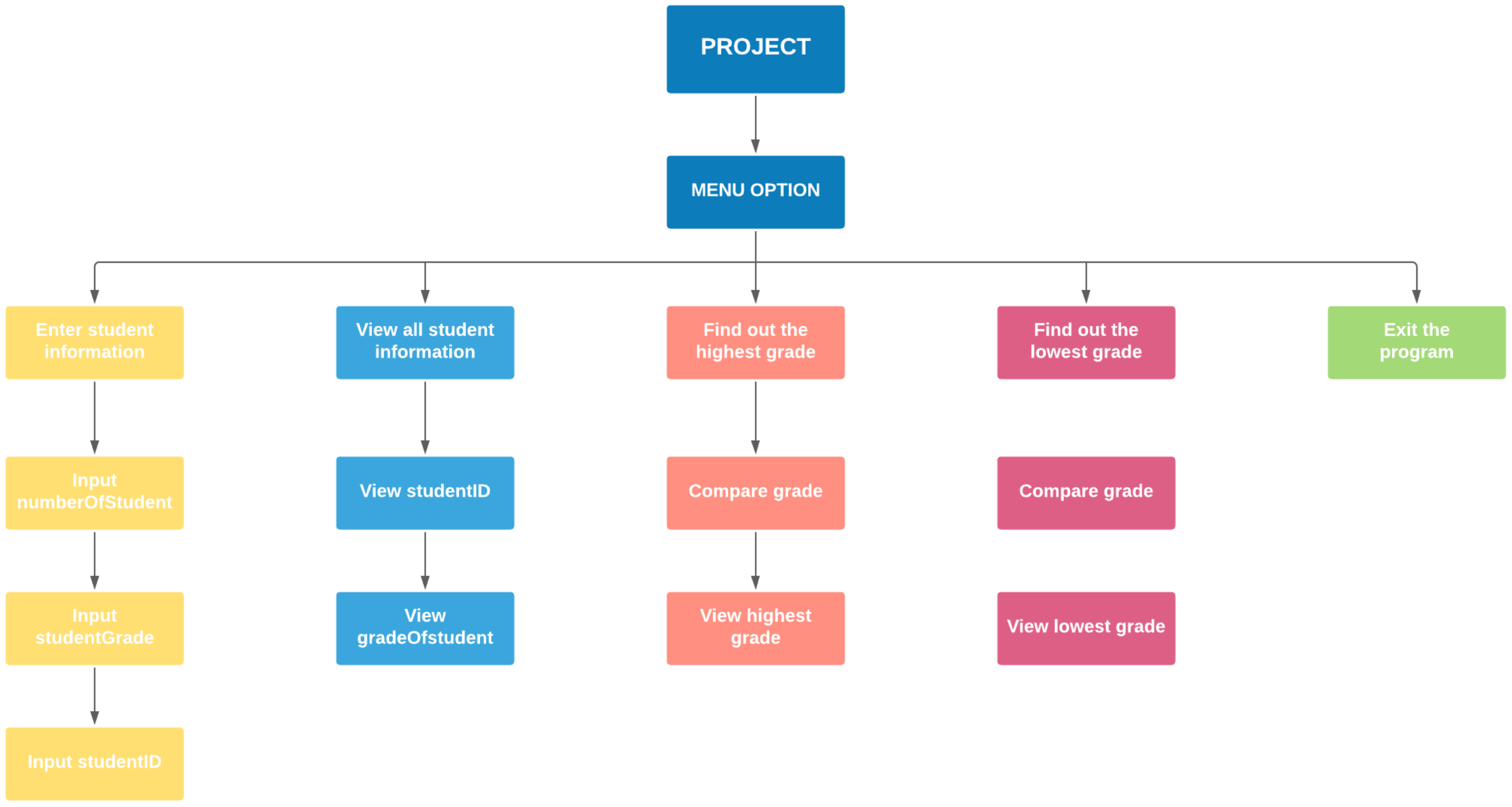


Figure 31: WBS of program

III. CONCLUSION (M2, P1):

1.Review flowchart:

My grade management program flowchart is quite comprehensive and very specific for each function, easy to understand, easy to apply because the implementation steps are very short but still logical and fully apply the requirements. My grade management application fully meets user requirements such as input points and IDs, showing highest and lowest scores. Each color in the flowchart represents a distinct icon shape, for example, a pink diamond (condition) icon, a blue square (process) icon, etc. However, there are still some disadvantages that make the program.

Become totally perfect. The first thing to mention is, when utilizing this flowchart, which is not the perfect answer for all kinds of matters out there, making changes may be a pain. This is because you will very certainly have to redraw the entire flowchart if there are changes to the process or if a process must be modified, resulting in lost time.

2.Evaluation:

2.1. Evaluation brief:

- Basically, the first version of the chapter still meets the basic needs of users and will have the latest updates from bug fixes or feature updates.
- Add some features according to user preferences such as cross-device usability or interface options to increase user interaction.
- use SEO (Search Engine Optimization) ranking enhancement process for search engine optimization.
- Add student details: Get data from the user and add a student to the student list.
- Update student number: to update student records and not request new details for all fields.
- program has more Object-Oriented Programming to optimize the program
- After users use the score entry feature, they will further customize the feature of sending score notifications to students
- Use database to store data of student

2.2. Evaluation of procedural program:

Advantage**:**

- It is supported by many general-purpose programming languages.
- They have the advantage that we can immediately execute the coding of a program in a remarkably short amount of time without having to create any additional objects.
- Its relative simplicity, as well as the ease with which compilers and interpreters may be implemented
- Programs are designed in a procedural oriented, so they are clearly functional and well structured
- We use less memory.

Disadvantage**:**

- It has poor data security, so all data is made public.
- It quite abstract compared to real world problems.
- They don't have the ability to reuse old code, so rewriting and syncing is quiet time consuming

- It solves real-world issues, although these are usually complicated programs.
- Procedural programming is often difficult to check for errors because the programs are broken up, making it difficult to access the program.

2.3. Evaluation difficulties:

- Since they break down the functions in the program, debugging is difficult.
- The inability to reuse code throughout a program is a characteristic of procedural programming. Rewriting the same sort of code several times across a program might increase the project's development cost and time.
- Data is accessible to the entire application while using procedural programming, making it less safe. With real-world difficulties, it's tough to relate to the PP paradigm. In a big application, it might be difficult to determine where global data belongs.

References

[1]Available at: https://codeforwin.org/2017/05/low-level-languages-advantages-disadvantages.html
[Accessed 16 October 2021].

[2]Available at: https://codeforwin.org/2017/05/high-level-languages-advantages-disadvantages.html
[Accessed 16 October 2021].

[3]Available at: https://www.dammio.com/2019/11/30/bang-xep-hang-cac-ngon-ngu-lap-trinh-theo-do-pho-bien/popular-programming-languages
[Accessed 18 October 2021].

[4]Available at: \https://javaconceptoftheday.com/history-of-programming-languages/
[Accessed 16 OCtober 2021].

[5]Available at: https://intech.vietnamworks.com/article/giai-thich-14-ngon-ngu-lap-trinh-pho-bien
[Accessed 17 OCtober 2021].

[6]Available at: https://www.javatpoint.com/functions-in-c
[Accessed 20 October 2021].

[7]Available at: https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/
[Accessed 16 OCtober 2021].

[9]Available at: https://insights.stackoverflow.com/survey/2021
[Accessed 16 October 2021].

[11]Available at: https://codeforwin.org/2017/08/switch-case-statement-in-c.html
[Accessed 19 October 2017].

[12]Available at: https://www.codesansar.com/c-programming/if-else-if-statement-or-ladder-with-examples.htm
[Accessed 19 October 2021].

[13]Available at:
https://isaaccomputerscience.org/concepts/prog_pas_paradigm?examBoard=all&stage=all&topic=procedural_programming
[Accessed 17 October 2021].

[14]Available at: https://www.istockphoto.com/vi/h%C3%ACnh-minh-h%E1%BB%8Da/math-teacher
[Accessed 18 October 2021].

[15]Available at: https://www.javatpoint.com/functions-in-c
[Accessed 20 October 20221].

[16]Available at: https://www.knowprogram.com/c-programming/data-types-in-c/
[Accessed 18 October 2021].

[17]Available at: https://www.programiz.com/c-programming/c-for-loop
[Accessed 20 October 2021].

[18]Available at: https://www.programiz.com/c-programming/c-do-while-loops
[Accessed 20 October 2021].

[20]Available at: https://www.programiz.com/c-programming/c-do-while-loops
[Accessed 20 October 2021].

[21]Available at: https://www.knowprogram.com/c-programming/data-types-in-c/
[Accessed 20 October 2021].

[22]Available at: http://www.trytoprogram.com/c-programming/c-programming-if-statement/
[Accessed 19 October 2021].

[23]Available at: https://www.researchgate.net/figure/Flowchart-of-the-involved-processes-when-using-C64-to-call-a-compiled_fig2_314092258
[Accessed 20 October 2021].