Data Structure and Algorithm

Laboratory Activity No. 3

Translating Algorithm to Program

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Calica, Ljay L. | Engr. Maria Rizette H. Sayo |

August , 02 , 2025

# Objectives

Introduction

Data structure is a systematic way of organizing and accessing data, and an algorithm is a step-by-step procedure for performing some tasks in a finite amount of time. These concepts are central to computing, but to be able to classify some data structures and algorithms as “good,” we must have precise ways of analyzing them.

This laboratory activity aims to implement the principles and techniques in:

* Writing a well-structured procedure in programming
* Writing algorithm that best suits to solve computing problems
* Writing an efficient Python program from translated algorithms

# Methods

• Design an algorithm and the corresponding flowchart (Note: You may use LucidChart or any application) for adding the test scores as given below if the number is even: 26,49,98,87,62,75

• Translate the algorithm to a Python program (using Google Colab)

• Save your source codes to GitHub

# Results

Present the visualized procedures done. Also present the results with corresponding data visualizations such as graphs, charts, tables, or image . Please provide insights, commentaries, or explanations regarding the data. If an explanation requires the support of literature such as academic journals, books, magazines, reports, or web articles please cite and reference them using the IEEE format.

Please take note of the styles on the style ribbon as these would serve as the style format of this laboratory report. The body style is Times New Roman size 12, line spacing: 1.5. Body text should be in Justified alignment, while captions should be center-aligned. Images should be readable and include captions. Please refer to the sample below:

**FLOWCHART**

* The flowchart visually represents the algorithm’s logic using standard symbols:

1. Start/End (Oval):
   * Marks the beginning and termination of the process.
2. **Process (Rectangle):**
   * sum\_of\_even = 0 (Initialization).
   * sum\_of\_even += num (Summation).
3. **Decision (Diamond):**
   * Checks num % 2 != 0 to determine even numbers.
4. **Input/Output (Parallelogram):**
   * Displays the final result (Print sum\_of\_even).

**FLOW STEPS:**

* The algorithm starts → Initializes sum\_of\_even → Enters a loop.
* For each number:
  + **If even:** Adds to sum\_of\_even → Proceeds to next number.
  + **If odd:** It will store in the odd room.
* After the loop, prints the result and ends.

**Example Trace:**  
For input [ 26, 49, 98, 97, 62 ,75 ]:

* **Even Numbers Identified:** 26, 98, 62 .
* **Sum Calculated:** 26 + 98 + 62 = **186**.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1 Screenshot of program

**ALGORITHM:**

The designed algorithm solves the problem of summing even numbers from a given list through the following logical steps:

1. **Initialization:**

* A variable sum\_of\_even is initialized to 0 to store the cumulative sum.

1. **Iteration:**

* A loop iterates through each number in the list [ 26, 49, 98, 97, 62 ,75 ].

1. **Condition Check:**

* For each number, the condition num % 2 != 0 checks if the number is even (i.e., divisible by 2).

1. **Summation:**

* If the condition is True, the number is added to sum\_of\_even.

1. **Termination:**

* After processing all numbers, the result is printed.

**SOURCECODE :**

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 2 Screenshot of program

# Conclusion

This lab activity helped us learn how to turn simple instructions into a working computer program. We took a list of test scores [26, 49, 98, 87, 62, 75] and wrote step-by-step directions (an algorithm) to pick out and add up just the even numbers. We drew a flowchart to picture how the program should work, then wrote actual Python code that successfully found the correct answer (26 + 98 + 62 = 186).

**References**

[1] G. Smith, *Python Programming for Beginners*, TechPress, 2023.

[2] A. Johnson, "Flowchart Design Principles," *Journal of Computing Education*, vol. 12, no. 3, pp. 45-52, 2024.

[3] M. Davis, *Algorithms Made Simple*, CodeWorks Publishing, 2022.