Turiba University

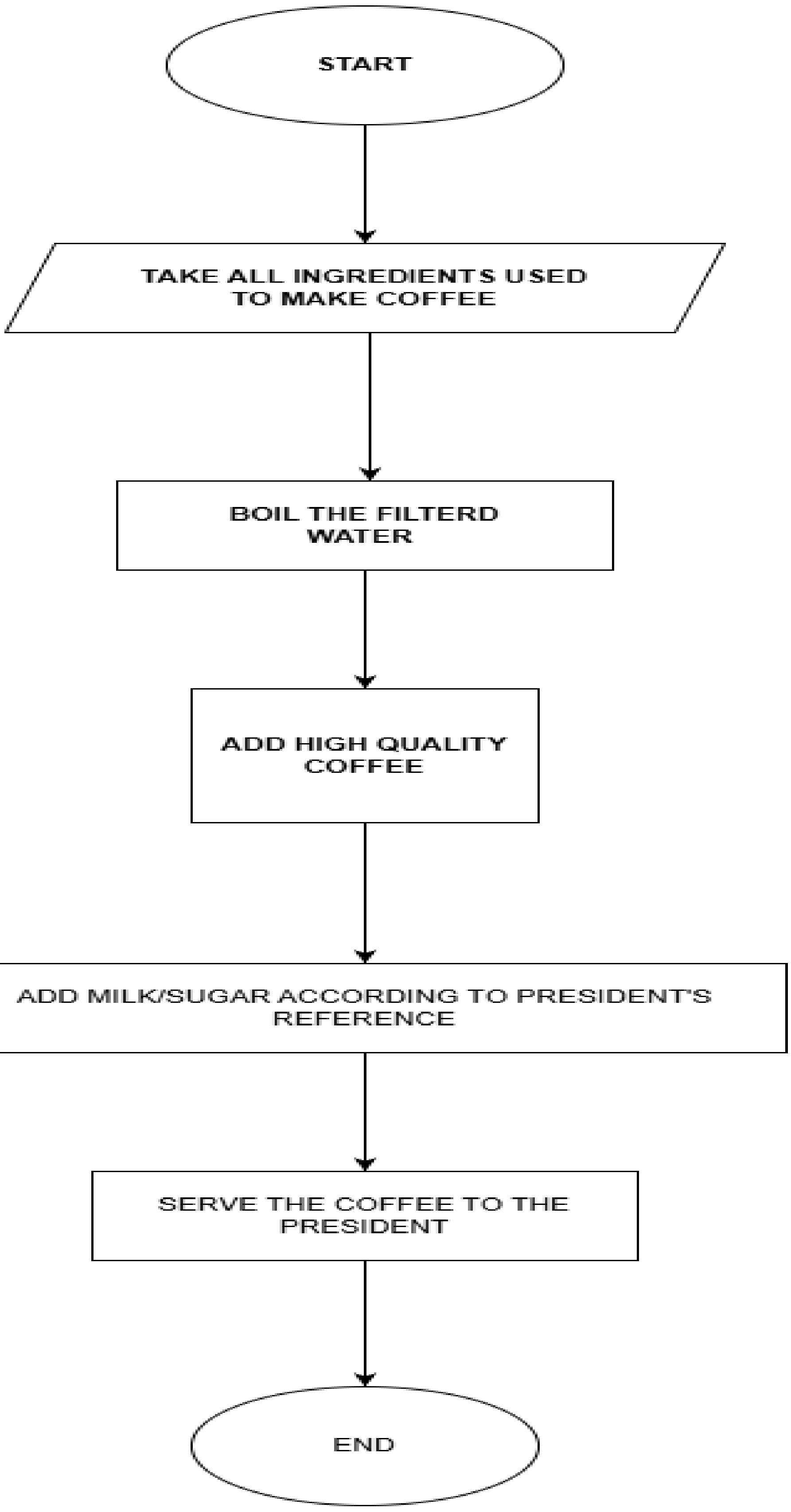
PARTHEEV PULIYANMANAYIL RADHAKRISHNAN

FOUR EYE METHOD

PROFESSIONAL BACHELOR THESIS

Study Programme: Computer Systems

Author:	**PARTHEEV**
***Thesis* Advisor:**	**JAANIS PEKSA**


START
TAKE ALL INGREDIENTS USED TO MAKE COFFEE
BOIL THE FILTERD WATER
ADD HIGH QUALITY COFFEE
ADD MILK/SUGAR ACCORDING TO PRESIDENT'S REFERENCE
SERVE THE COFFEE TO THE PRESIDENT
END

CLASSMATE : **BISMAYA**

FLOWCHART : Coffee to the president

Report on the Pros and Cons of the Coffee-Making Flowchart for Coding

1. Introduction

Flowcharts are commonly used in software development to visually represent processes, logic flows, and system behavior. The provided flowchart outlines the steps involved in preparing coffee for a "president," starting from gathering ingredients to serving the final beverage. This report evaluates the flowchart in terms of how effectively it supports coding, algorithm design, and system implementation.

2. Strengths (Pros)

2.1 Clear Sequential Structure

The flowchart follows a top-to-bottom linear progression, which makes it easy for developers to translate the process into a procedural program. Each step is clearly connected, helping coders visualize the order of operations.

2.2 Direct Mapping to Functions or Methods

Each block represents a concrete action that can be turned into individual functions, such as:

- gatherIngredients()
- boilWater()
- addCoffee()
- adjustForPreference()
- serveCoffee()

This modularity is helpful in coding.

2.3 Good Use of Standard Flowchart Symbols

The chart uses:

- Oval for Start/End
- Rectangle for processes
- Parallelogram/Stadium shape for serving step

While not perfect, it generally follows flowchart conventions, making it readable and familiar to developers.

2.4 Simple and Easy to Understand

The process is easy to follow even for beginners. It avoids unnecessary complexity, which is good for basic algorithm translation.

---

3. Weaknesses (Cons)

3.1 No Decision (Conditional) Blocks

From a coding perspective, the flow lacks decision points. For example:

- What if a required ingredient is missing?
- What if the president wants no milk or sugar?
- What if the water fails to boil?

In programming, such decisions require if/else or switch statements, but the flowchart does not show any of these branches.

3.2 User Input Not Clearly Defined

The step “Add milk/sugar according to president's preference” implies input or variables, but the chart does not show:

- How preferences are obtained
- What happens if preferences are unknown
- How default values are handled

A more complete flowchart would include an input step or a user-interaction block.

3.3 Lack of Error Handling

Realistic coding requires error handling, such as:

- Missing ingredients
- Water not heating
- Incorrect measurements

None of these are represented, making the flowchart less useful for robust programming.

3.4 Overly High-Level

The steps are general and lack the detail needed for accurate translation into code. For instance, “add high quality coffee” does not specify:

- Quantity
- Measurement units
- Conditions for quality

This makes it difficult to create precise algorithms.

3.5 No Loops or Repetition Structures

If the process needed repeating (e.g., preparing multiple cups), nothing in the chart indicates looping. Coding often requires loops such as for, while, or repeat, which are not represented.

---

4. Recommendations for Improvement

4.1 Add Conditional Blocks

Include decision symbols (diamonds) for:

- Ingredient availability
- Water temperature
- Preferences on milk and sugar

4.2 Add Input/Output Steps

Explicitly show:

- Getting preference input
- Confirming ingredient list

4.3 Include Error-Handling Paths

Add branches for:

- Missing ingredients
- Boiling failure
- Incorrect input

4.4 Add More Technical Detail

Include quantitative steps (e.g., “Add 10g coffee”), which can directly translate into constants or variables in code.

4.5 Consider Modularity

Group steps into logical modules so coders can map them into classes or methods.

---

5. Conclusion

The current flowchart is simple, readable, and suitable for explaining basic sequential logic. However, from a coding perspective, it lacks decision points, error handling, variable inputs, and technical detail necessary for an accurate algorithm or automated system. With additional decision blocks, inputs, and branching paths, the chart can become a highly effective tool for program design and implementation.

.