ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 1: Programming		
Submission date	06/08/2024	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Nguyễn Thế Hùng	**Student ID**	Bh01689
Class	SU24 - Programming - SEO7203	**Assessor name**	Tạ Quang Hiếu
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	Hùng

Grading grid

P1	P2	M1	D1

❒ **Summative Feedback:** ❒**Resubmission Feedback:**

Grade:	**Assessor Signature:**	**Date:**

Lecturer Signature:

Table of Contents

A: Introduction

You currently work for ABC software, an independent software development company that designs and builds bespoke software solutions for various companies of different sizes that cover a range of different industries. The software that they design uses a wide range of technologies, from simple stand-alone programs to large web-based applications.

ABC software has requested a simple program that will calculate the monthly waterbill each month.

The CEO of the company has reviewed the client requirements and has determinedthat this is a suitable project for you to take on. The company wants to see how you use and apply the development environments and code standards.

Purpose of the report

This report aims to provide a detailed overview of the fundamentals and important concepts in programming and software development. The report will include the following contents:

Definition of basic algorithms and the process of programming an application:

Introduces basic algorithms such as sorting and searching.

Describe the software development process from requirements analysis to maintenance.
Explain the characteristics of procedural programming, object-oriented programming, and event-driven programming:

Compare and contrast these programming styles.
Discuss the advantages and disadvantages of each method.
Implement basic algorithms in code using an integrated development environment (IDE):

Provides concrete examples of how to implement algorithms in common programming languages.

Instructions for using IDE to develop and test code.
Define the debugging process and explain the importance of coding standards:

Describes the debugging process step by step.
Emphasizes the importance of compliance with coding standards in maintaining source code quality and consistency.

The specific purposes of the report include:
Support for programming learners: Provides basic knowledge and practicalinstructions for those who are just starting out or are in the process of learning programming.

Enhance understanding of different programming methods: Helps readers understand the characteristics, advantages and disadvantages of procedural, object-oriented and event-driven programming styles.

Promote the adoption of good standards and processes: Encourage the practice ofgood coding standards and debugging processes to improve the quality and maintainability of software.

B: Body

I. Algorythm

1. Algorithm Definition

- An algorithm is a step-by-step procedure or formula for solving a problem, accomplishing a task, or achieving a desired outcome. It is a set of well-defined instructions that take an input, perform computations or operations on that input, and produce an output in a finite amount of time.

2. Characteristics of algorithm

- **Well-defined**: An algorithm must have clear and precise instructions. Each step should be unambiguous and executable without any room for interpretation. This clarity ensures that anyone implementing the algorithm will achieve the same result for the same inputs.

- **Finite**: Algorithms must terminate after a finite number of steps. They cannot run indefinitely or loop forever without producing a result. This ensures that algorithms are practical and usable within the constraints of computing resources.

- **Input**: An algorithm receives zero or more inputs, which are the initial data or values it operates on to produce outputs. Inputs can vary depending on the specific problem the algorithm is designed to solve.

- **Output**: An algorithm produces at least one output based on the inputs and the sequence of steps it executes. The output could be a solution to a problem, a transformed dataset, a decision, or any other relevant result.

- **Effective**: An effective algorithm achieves the desired result correctly. It solves the problem it was designed for and produces the correct output for all valid inputs within a reasonable amount of time and using reasonable resources (such as memory and processing power).

- **Generalization**: Algorithms are often designed to be applicable to a class of problems rather than just a single instance. They can be generalized to handle similar situations or data structures, making them reusable and versatile.

- **Ordered**: Algorithms consist of a sequence of steps that are executed in a specific order. The order in which these steps are performed is crucial as it determines the correctness and efficiency of the algorithm.

- **Optimality**: In some contexts, algorithms strive to be optimal, meaning they achieve the desired result using the fewest steps, with minimal resource consumption (such as time or memory).

- **Correctness**: An algorithm should produce the correct output for all valid inputs. It should adhere to the logic and rules defined by its design without errors or inconsistencies.

- **Modifiability**: Algorithms should be designed in such a way that they can be modified or extended easily if the problem requirements change or if improvements are needed.

3. Algorithm representation and examples

- Algorithmic representation is the process of describing the specific steps to solve a problem using appropriate languages, symbols, and structures. This is a way to present an algorithmic solution in a systematic and clear way, helping readers understand how the algorithm works without needing to know the specific implementation details.

- **Common ways of representing algorithms include:**

+ Algorithm flowchart: Uses geometric shapes and arrows to represent the steps and flow of the algorithm. Algorithm flowcharts are often used to clearly illustrate conditions, loops, and choices in an algorithm.

+ Pseudocode: Is a simple pseudo programming language, used to describe the steps of an algorithm without requiring specific syntax like in a specific programming language. Pseudocode often uses natural language and has a structure similar to actual code.

+ Code: Implements an algorithm in a specific programming language, such as Python, Java, C++, etc. The code provides the most detail about how the algorithm is implemented on a particular platform.

+ Textual description: Use natural language to describe each step of the algorithm in detail. This method is often used in technical documents or to explain algorithms to people without programming expertise.

*** For example:** nature language for take a cup of tea

Prepare tea: First, you need to prepare a cup of tea and tea bags or tea leaves depending on your preference.

Boil water: Boil water until it boils

Put tea in the cup: put tea leaves or dry tea in the cup

Pour water into cup: After the water boils, pour water into the cup, let the tea leaves soak in the water for about 3-5 minutes. Steeping time may vary depending on the type of tea.

Enjoy: After steeping, you can enjoy a delicious and warm cup of tea.

II. Step in Developing an application:

1. Clearly state what the application development process is?

- The application development process is a series of steps and activities performed to design, develop, test, and deploy an application from the initial idea until it can be used by the end user. This process includes the following main stages

2. Explain the steps in the process with a sense of what the purpose is?

+ Collect necessary functional and non-functional requirements for the application.

+ Analyze and produce a detailed requirements document as a basis for the next steps.

- **Design (Design)**:

+Design includes two main parts: user interface (UI) design and user experience (UX) design, along with the system's architectural design.

+ Define data structures, define functional components and determine how components interact.

+ Popular tools in this stage include Adobe XD, Sketch for UI/UX design, and architectural design tools like UML (Unified Modeling Language).

- **Develop (Development)**:

+ This is the stage of implementing application programming based on the determined design.

+ Development can be done using many different programming languages and frameworks depending on the project requirements.

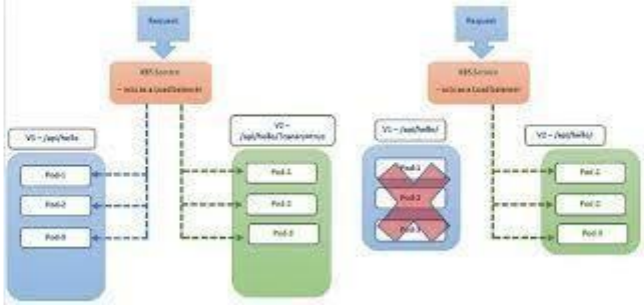
+ Manage source code and use support tools such as Git for version control and source code management.

- **Test (Testing)**:

+Includes functional testing, integration testing, system testing and user acceptance testing (UAT).

+ The purpose is to ensure that the application works as expected, is error-free and meets the set requirements.

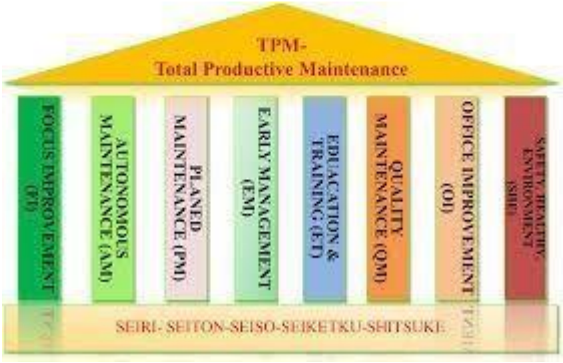
- **Deployment (Deployment)**:



+This is the stage of deploying the application to the production environment.

+ Includes activities such as system configuration, application installation on the server, and preparation for release to end users.

- **Maintenance and improvements (Maintenance and Improvement)**:


TPM-
Total Productive Maintenance
FOCUS IMPROVEMENT (FI)
AUTONOMOUS MAINTENANCE (AM)
PLANED MAINTENANCE (PM)
EARLY MANAGEMENT (EM)
EDUACATION & TRAINING (ET)
QUALITY MAINTENANCE (QM)
OFFICE IMPROVEMENT (OI)
SAFETY HEALTHY ENVIRONMENT (SHE)
SEIRI- SEITON-SEISO-SEIKETKU-SHITSUKE

+ Once developed, the application needs to be maintained to ensure stability and performance.

+ At the same time, developers can also continue to improve the application by adding new features, optimizing performance and improving user experience.

III.Outline the process for the given scenario:

1. Process Outline for Developing a Water Bill Calculator Application

1.1 Problem Definition and Requirements Gathering

- **Purpose**: Develop a console application that calculates water bills based on customer input for different customer types.

- **Steps**:

+ Define the scope: Calculate water bills based on consumption and customer type.

+ Gather requirements: Input customer name, last month's and this month's water meter readings, customer type, and number of people (if household).

1.2 Algorithm Definition and Characteristics

- **Algorithm Definition**: The algorithm calculates the water bill based on consumption and customer type, using predefined pricing and environmental fees.

- **Characteristics**:

+ **Final Note**

- While this process outline focuses on a console application, future enhancements could include features like sorting, searching customer records, generating invoices, and managing payment dues, expanding the application's functionality and usability.

1.3 Algorithm Representation and Examples

- **Representation**:

+ Nature Language Example: This algorithm can be described in natural language as follows:

- Gather customer details and water meter readings.
- Determine customer type and calculate consumption.
- Apply specific pricing rules based on customer type and consumption.
- Calculate total bill including environmental fees.
- Display customer information and calculated bill.

- **Final Note**

+ While this process outline focuses on a console application, future enhancements could include features like sorting, searching customer records, generating invoices, and managing payment dues, expanding the application's functionality and usability.

Example Flowchart for Water Bill Calculator

2. Steps in Developing the Application

- **Clearly State What the Application Development Process Is**: The process involves designing, coding, testing, and deploying a console application for calculating water bills.

+ Explain the Steps with Purpose:

- **Input Gathering**: Obtain user inputs such as customer name, water meter readings, and customer type.
- **Consumption Calculation**: Compute water consumption based on meter readings.
- **Bill Calculation**: Determine pricing and environmental fees based on customer type and consumption.
- **Output Generation**: Display customer information and calculated water bill.

+ **Illustrative Example Images**

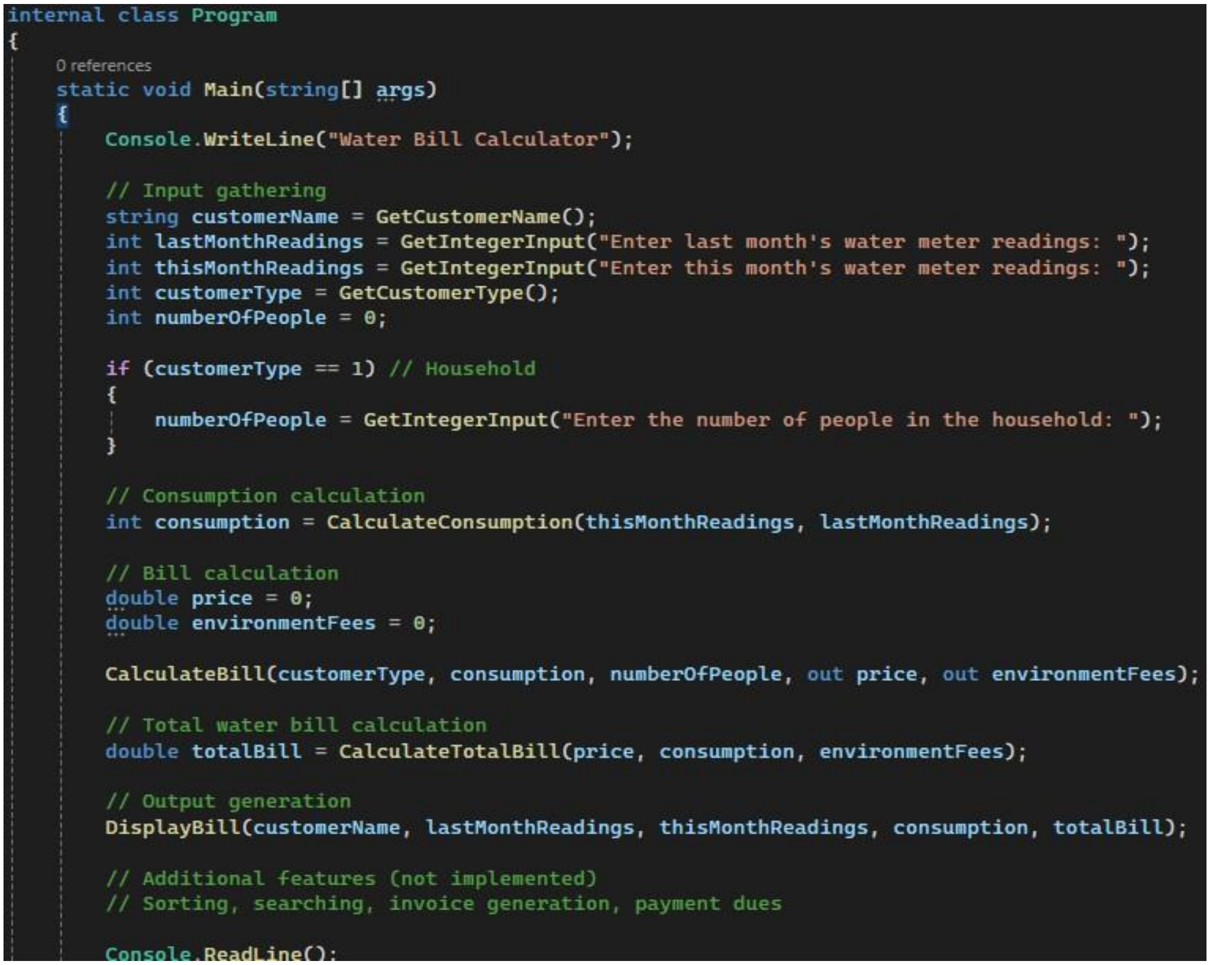
- **UI Wireframe**: Example not applicable for a console application.
- **Test Plan**: Outline not shown as it's typically for functional validation.
- **Deployment Pipeline**: Example not relevant for a standalone console application.

+ **Conclusion**

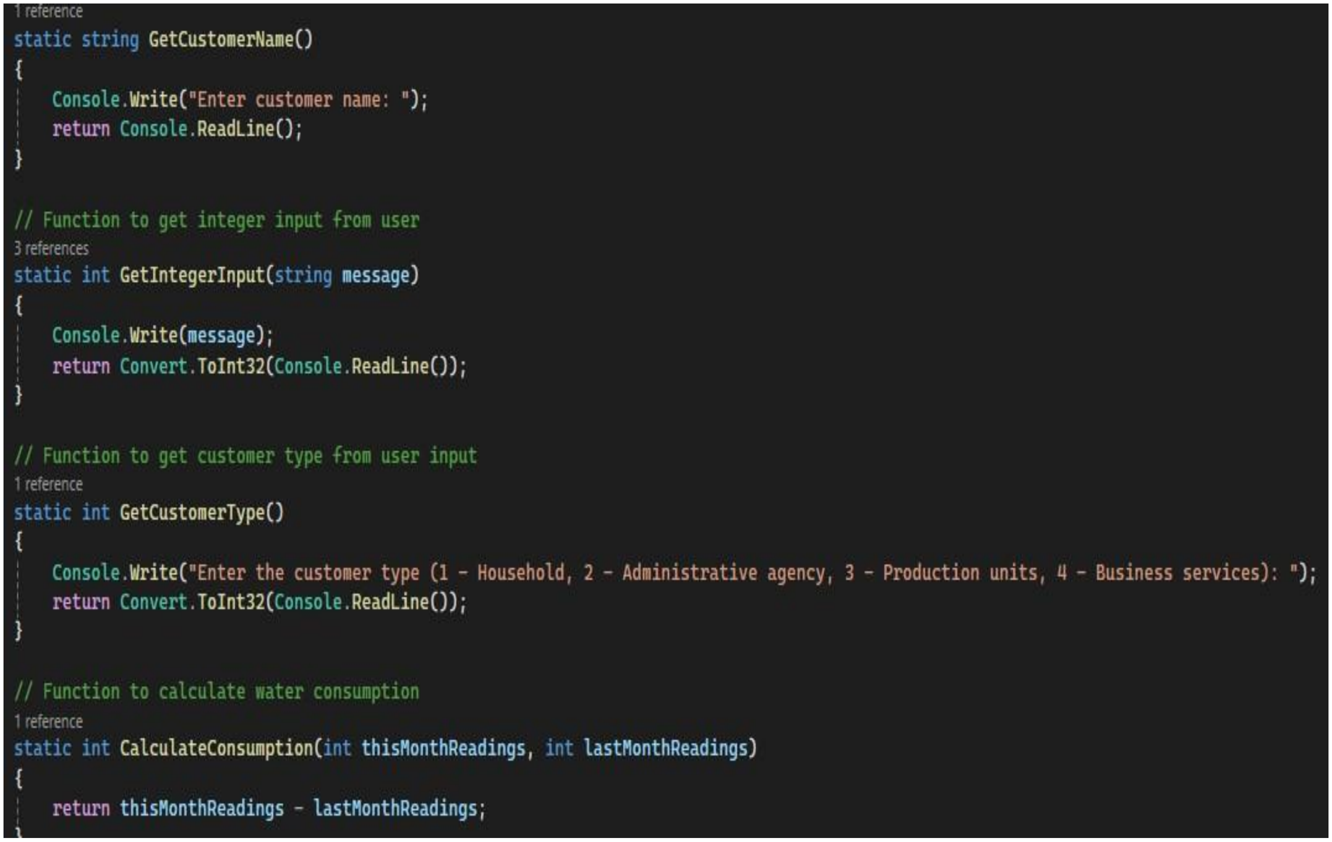
- his structured approach ensures that the water bill calculator application is developed systematically, starting from gathering requirements to implementing and testing the application logic. Each step ensures clarity and efficiency in delivTering the intended functionality.

3. Analyze each function that needs to be developed, and draw a flowchart for eachfunction

- Input customer Details :

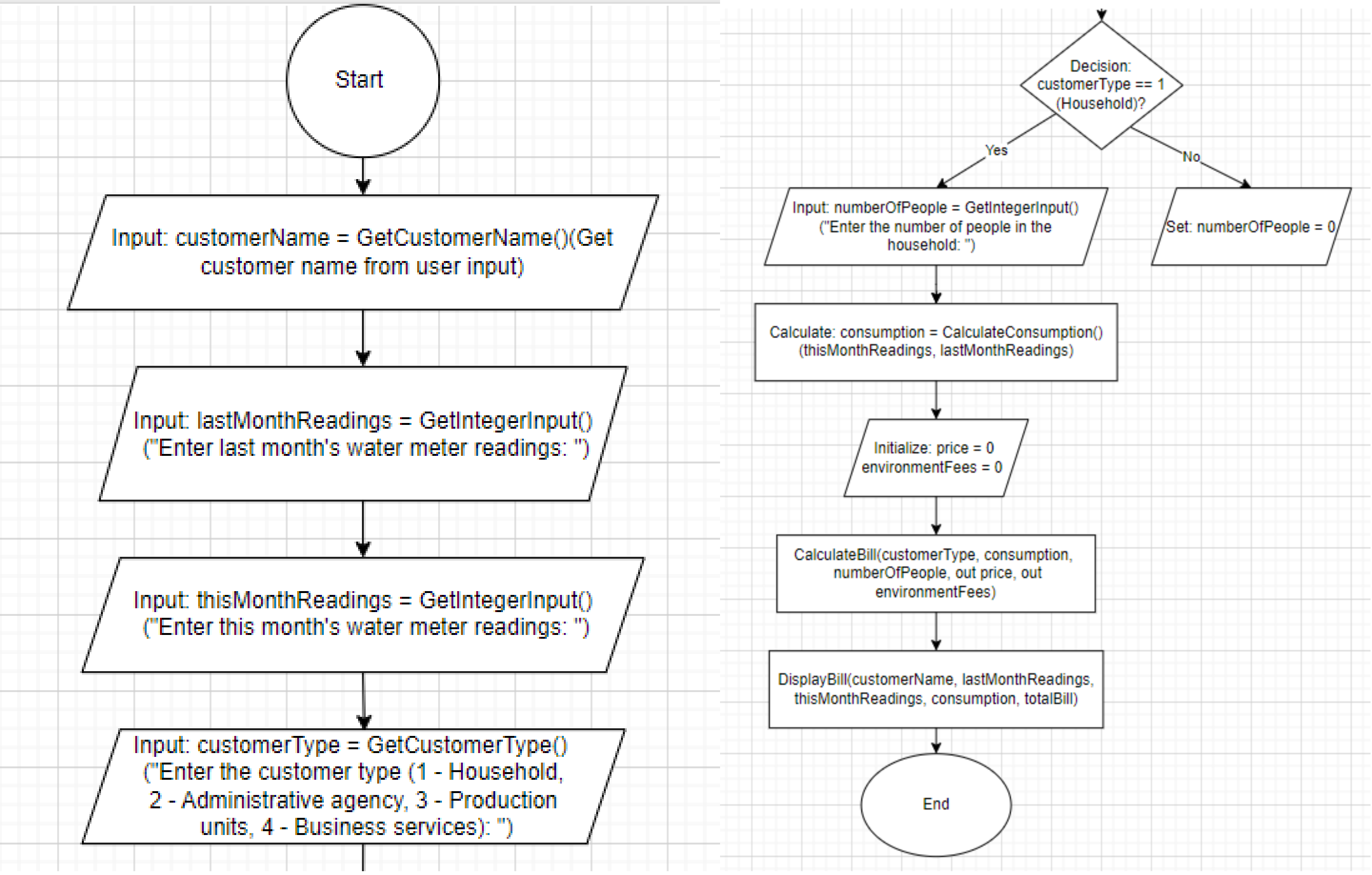


```
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Water Bill Calculator");

        // Input gathering
        string customerName = GetCustomerName();
        int lastMonthReadings = GetIntegerInput("Enter last month's water meter readings: ");
        int thisMonthReadings = GetIntegerInput("Enter this month's water meter readings: ");
        int customerType = GetCustomerType();
        int numberOfPeople = 0;

        if (customerType == 1) // Household
        {
            numberOfPeople = GetIntegerInput("Enter the number of people in the household: ");
        }

        // Consumption calculation
        int consumption = CalculateConsumption(thisMonthReadings, lastMonthReadings);

        // Bill calculation
        double price = 0;
        double environmentFees = 0;

        CalculateBill(customerType, consumption, numberOfPeople, out price, out environmentFees);

        // Total water bill calculation
        double totalBill = CalculateTotalBill(price, consumption, environmentFees);

        // Output generation
        DisplayBill(customerName, lastMonthReadings, thisMonthReadings, consumption, totalBill);

        // Additional features (not implemented)
        // Sorting, searching, invoice generation, payment dues

        Console.ReadLine();
```

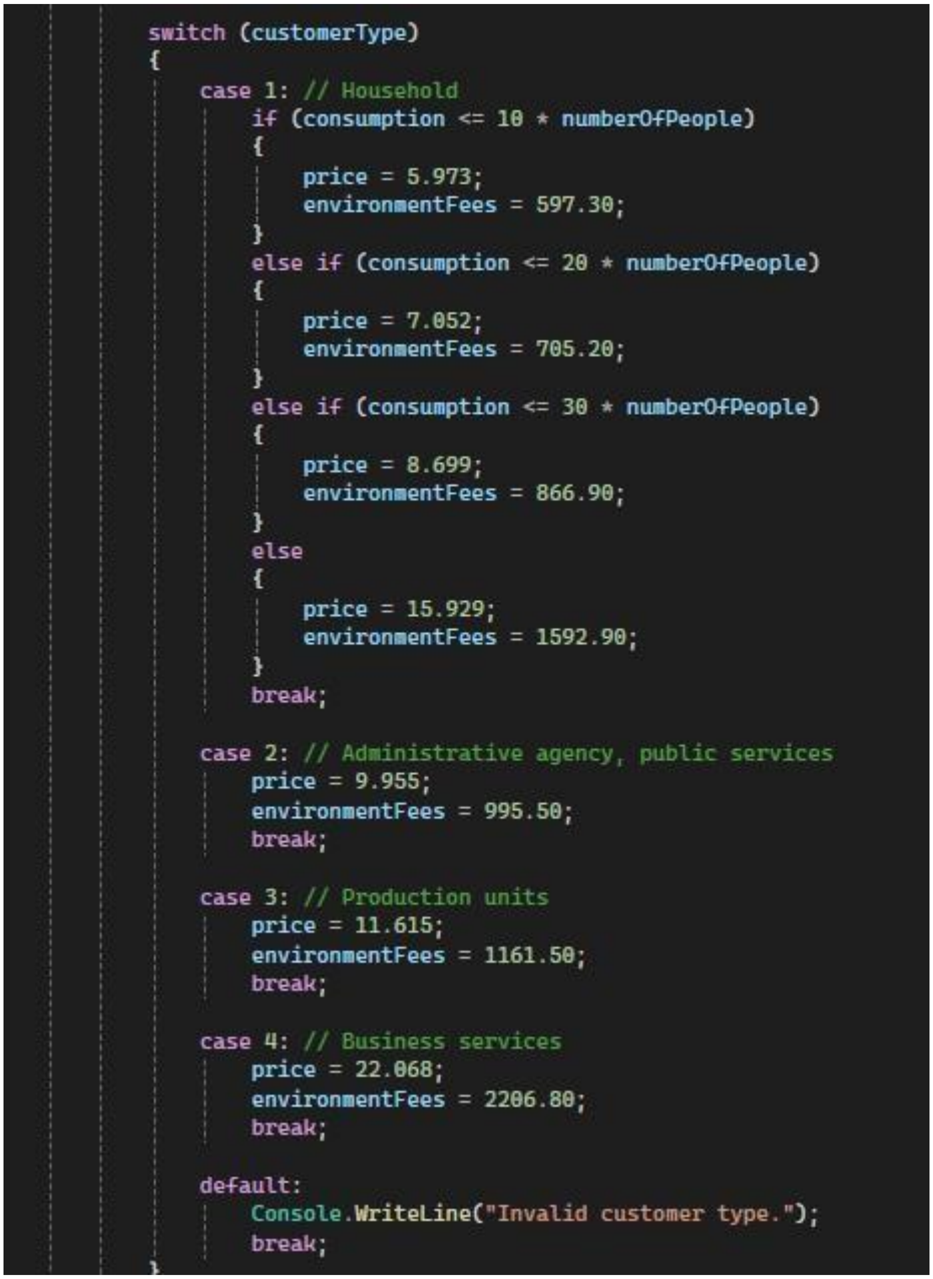


```
1 reference
static string GetCustomerName()
{
    Console.Write("Enter customer name: ");
    return Console.ReadLine();
}

// Function to get integer input from user
3 references
static int GetIntegerInput(string message)
{
    Console.Write(message);
    return Convert.ToInt32(Console.ReadLine());
}

// Function to get customer type from user input
1 reference
static int GetCustomerType()
{
    Console.Write("Enter the customer type (1 - Household, 2 - Administrative agency, 3 - Production units, 4 - Business services): ");
    return Convert.ToInt32(Console.ReadLine());
}

// Function to calculate water consumption
1 reference
static int CalculateConsumption(int thisMonthReadings, int lastMonthReadings)
{
    return thisMonthReadings - lastMonthReadings;
```

Flowchart :

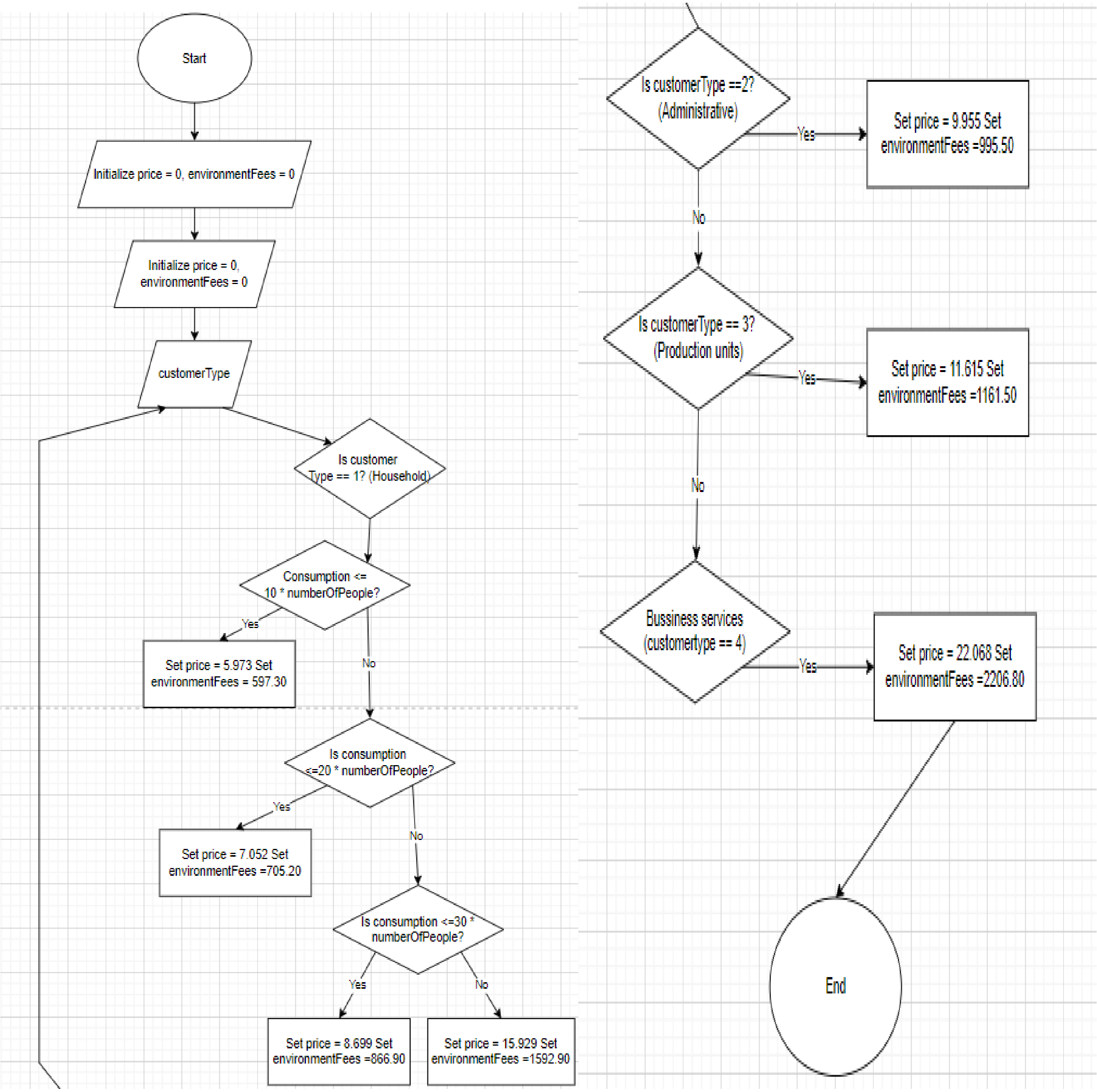

Start
Input: customerName = GetCustomerName()(Get customer name from user input)
Input: lastMonthReadings = GetIntegerInput() ("Enter last month's water meter readings: ")
Input: thisMonthReadings = GetIntegerInput() ("Enter this month's water meter readings: ")
Input: customerType = GetCustomerType() ("Enter the customer type (1 - Household, 2 - Administrative agency, 3 - Production units, 4 - Business services): ")
Decision: customerType == 1 (Household)?
Yes
No
Input: numberOfPeople = GetIntegerInput() ("Enter the number of people in the household: ")
Set: numberOfPeople = 0
Calculate: consumption = CalculateConsumption() (thisMonthReadings, lastMonthReadings)
Initialize: price = 0 environmentFees = 0
CalculateBill(customerType, consumption, numberOfPeople, out price, out environmentFees)
DisplayBill(customerName, lastMonthReadings, thisMonthReadings, consumption, totalBill)
End

- Initialize Billing Variables :

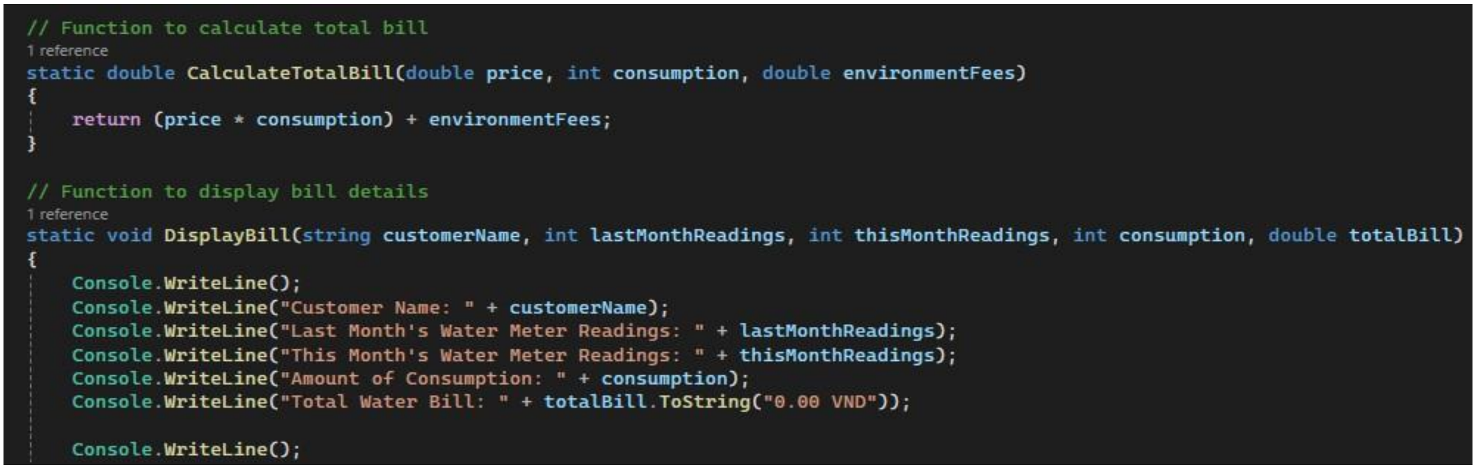


```
switch (customerType)
{
    case 1: // Household
        if (consumption <= 10 * numberOfPeople)
        {
            price = 5.973;
            environmentFees = 597.30;
        }
        else if (consumption <= 20 * numberOfPeople)
        {
            price = 7.052;
            environmentFees = 705.20;
        }
        else if (consumption <= 30 * numberOfPeople)
        {
            price = 8.699;
            environmentFees = 866.90;
        }
        else
        {
            price = 15.929;
            environmentFees = 1592.90;
        }
        break;

    case 2: // Administrative agency, public services
        price = 9.955;
        environmentFees = 995.50;
        break;

    case 3: // Production units
        price = 11.615;
        environmentFees = 1161.50;
        break;

    case 4: // Business services
        price = 22.068;
        environmentFees = 2206.80;
        break;

    default:
        Console.WriteLine("Invalid customer type.");
        break;
}
```

Flowchart :

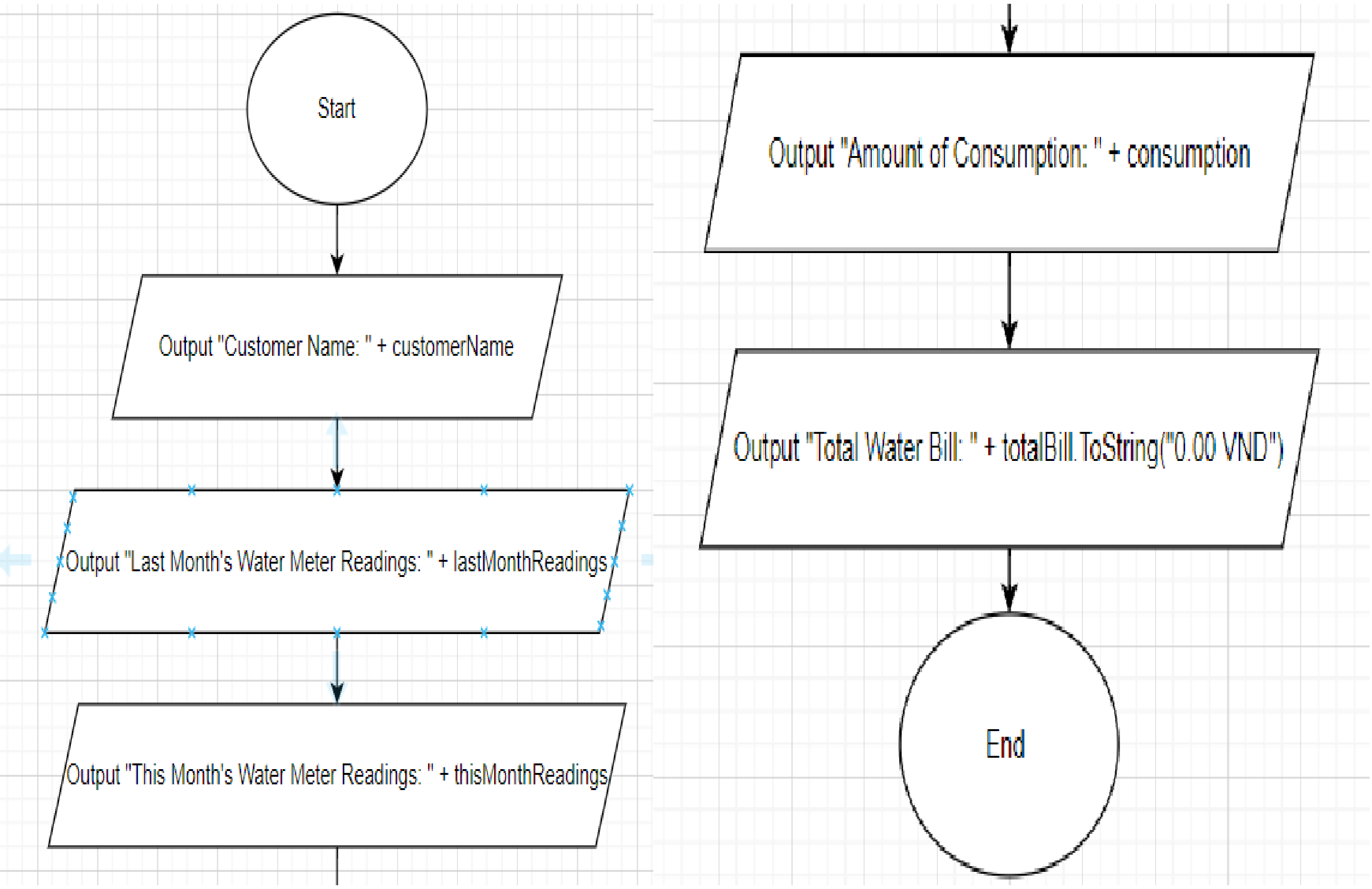

Start
Initialize price = 0, environmentFees = 0
Initialize price = 0, environmentFees = 0
customerType
Is customer Type == 1? (Household)
Consumption <= 10 * numberOfPeople?
Yes
Set price = 5.973 Set environmentFees = 597.30
No
Is consumption <=20 * numberOfPeople?
Yes
Set price = 7.052 Set environmentFees =705.20
No
Is consumption <=30 * numberOfPeople?
Yes
No
Set price = 8.699 Set environmentFees =866.90
Set price = 15.929 Set environmentFees =1592.90
Is customerType ==2? (Administrative)
Yes
Set price = 9.955 Set environmentFees =995.50
No
Is customerType == 3? (Production units)
Yes
Set price = 11.615 Set environmentFees =1161.50
No
Bussiness services (customertype == 4)
Yes
Set price = 22.068 Set environmentFees =2206.80
End

- Output the Bill Details :

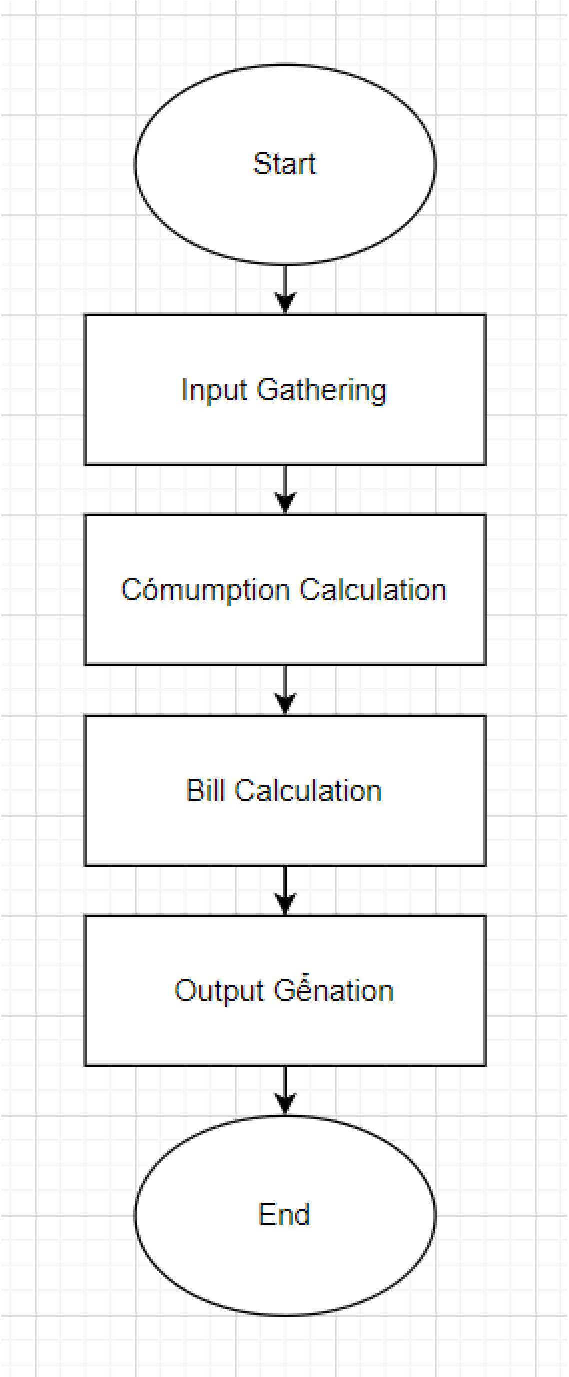


```
// Function to calculate total bill
1 reference
static double CalculateTotalBill(double price, int consumption, double environmentFees)
{
    return (price * consumption) + environmentFees;
}

// Function to display bill details
1 reference
static void DisplayBill(string customerName, int lastMonthReadings, int thisMonthReadings, int consumption, double totalBill)
{
    Console.WriteLine();
    Console.WriteLine("Customer Name: " + customerName);
    Console.WriteLine("Last Month's Water Meter Readings: " + lastMonthReadings);
    Console.WriteLine("This Month's Water Meter Readings: " + thisMonthReadings);
    Console.WriteLine("Amount of Consumption: " + consumption);
    Console.WriteLine("Total Water Bill: " + totalBill.ToString("0.00 VND"));

    Console.WriteLine();
```

Flowchart :


Start
Output "Customer Name: " + customerName
Output "Last Month's Water Meter Readings: " + lastMonthReadings
Output "This Month's Water Meter Readings: " + thisMonthReadings
Output "Amount of Consumption: " + consumption
Output "Total Water Bill: " + totalBill.ToString("0.00 VND")
End

4. Link flowchart => flowchart solves the problem :



C. References

- Link code :

1. https://github.com/Hung-2602/Code_asm_part1/blob/main/Code_asm_part1.cs

2. https://github.com/Hung-2602/Slide_ASM_Part1/upload

3. Nikolopoulou, K. (2023). *What Is an Algorithm? | Definition & Examples*. [online] Scribbr. Available at: https://www.scribbr.com/ai-tools/what-is-an-algorithm/.

4. UNF Its. (n.d.). *The Application Development Process*. [online] Available at: https://www.unf.edu/its/knowledgebase/banner/procedure-manual/2-Application-Development-Process.html.

5. SlideShare. (2024). *Water billing management system project report.pdf*. [online] Available at: https://fr.slideshare.net/slideshow/water-billing-management-system-project-report-pdf/269461898 [Accessed 4 Aug. 2024].