**ASSIGNMENT 1 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | Unit 1: Programming | | |
| **Submission date** |  | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** |  | **Student ID** |  |
| **Class** |  | **Assessor name** |  |
| **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 | M1 | D1 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **❒ Summative Feedback: ❒ Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Lecturer Signature:** | | |

ACKNOWLEDMENT

After nearly year learning at University of Greenwich(Viet Nam), I’ve been accumulate a bit knowledge and then I can improve my self-learning.

First of all, I so grateful for lectures here and I’ll continue make effort to gain a good job in the future to have a stable life.

**ASSIGNMENT 1**

**FIGURES**

[**Figure 1: Sample for algorithm.** 4](#_Toc43495171)

[**Figure 2: Flowchart Diagram.** 5](#_Toc43495172)

[**Figure 3: Declare variables.** 7](#_Toc43495173)

[**Figure 4: Code to read n.** 7](#_Toc43495174)

[**Figure 5: Result for step 2, computer ask to input number cases of array.** 7](#_Toc43495175)

[**Figure 6: Code for inputing value for each index in array.** 8](#_Toc43495176)

[**Figure 7: Result for step 3, Input value for each index.** 8](#_Toc43495177)

[**Figure 8: Code for printing each value which I had input.** 9](#_Toc43495178)

[**Figure 9: Result for step 4, print value of each index.** 9](#_Toc43495179)

[**Figure 10: Code for sorting array numbers.** 10](#_Toc43495180)

[**Figure 11: Result for step 5, print values after sorting.** 11](#_Toc43495181)

[**Figure 12: Test Case.** 12](#_Toc43495182)

**CONTENTS**

Table of Contents

[Chapter 1: PROBLEM STATEMENT 3](#_Toc43495158)

[1.1 SCENARIO 3](#_Toc43495159)

[1.2 REPORT OBJECTIVES 3](#_Toc43495160)

[1.3 REPORT OUTLINE 3](#_Toc43495161)

[Chapter 2: ALGORITHM 4](#_Toc43495162)

[2.1 What is an Algorithm 4](#_Toc43495163)

[2.2 Algorithm Properties 4](#_Toc43495164)

[Chapter 3: SOLUTION FOR PROCESS 5](#_Toc43495165)

[3.1 FLOWCHART 5](#_Toc43495166)

[3.2 PSEUDO CODE 5](#_Toc43495167)

[3.3 Implementation 7](#_Toc43495168)

[3.4 Test Case 11](#_Toc43495169)

[REFERENCES 13](#_Toc43495170)

# Chapter 1: PROBLEM STATEMENT

## 1.1 SCENARIO

\_ I had applied for a blog to share my programming experiences for viewers. They asked me to demonstrate my problem solving and basic skills. I decided to choose using algorithm to build my program and I’ll solve it by myself. To do this clearly for admin’s blog, I’ll prepare document to outline some basic definitions and process.

## 1.2 REPORT OBJECTIVES

\_ This report aims to show some basic definitions about algorithm of programming and samples with pictures then explaining about process what I’ve made.

\_ This report is used basic c# language programming with Console Application to make a system to sort an array numbers.

\_ Using this system user can input numbers of array then computer will sort follow ascending and display on user screen.

## 1.3 REPORT OUTLINE

\_ This report consists of 2 chapters and few references. Chapters of this thesis are organized as bellow:

+ Chapter 1: Reviewing about the report, answering briefly about problems which I’ll solve and report objectives to have a look detail about solving problems.

+ Chapter 2: Reviewing about Algorithm.

+ Chapter 3: Explain to solution for process.

# Chapter 2: ALGORITHM

## 2.1 What is an Algorithm

\_ Algorithm is a solution to solve problems.

\_ Algorithms are sets of step-by-step instructions for the computer to follow. They are at the heart of all computer programs. (Anon., n.d.) For example, when you want to make noodles, you must follow these steps:

+ Input – Ingredients and quantities

+ The process – recipe or method

+ output – what the finished noodles will be



**Figure 1: Sample for algorithm.**

## 2.2 Algorithm Properties

\_ Accuracy: It’s an essential thing. It can help programmer have a first look overview to build application.

\_ Clarity: Code are sorted cleanly and clearly to make smooth and flexible manipulation.

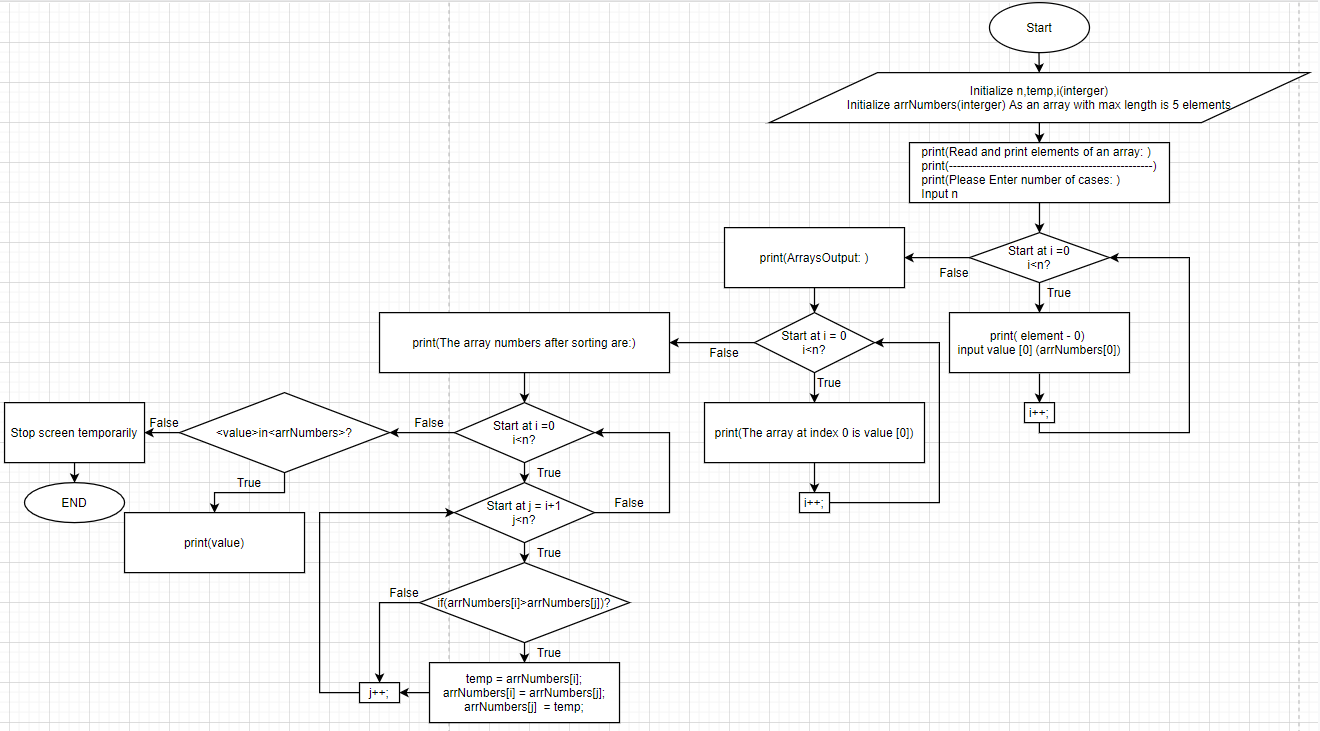
\_ Possibility: it means possibility of each algorithm can be used not only for one problem but also many problems.

\_ Objectivity: This Properties mean that you make a good decision when you have stuck with something happened which you don’t want to appear.

\_ Endly: it’s familiar with the result of your program.

# Chapter 3: SOLUTION FOR PROCESS

## 3.1 FLOWCHART



**Figure 2: Flowchart Diagram.**

## 3.2 PSEUDO CODE

START

PROGRAM SortAnArrayNumbers

INIT n,temp,i,arrNumbers

READ INPUT into n

FOR 0 through n - 1

PRINT element - 0

READ INPUT arrNumbers[0]

ENDFOR

PRINT ArraysOutput

FOR 0 through n - 1

PRINT The array at index 0 is arrNumbers[0]

ENDFOR

PRINT The array numbers after sorting are:

FOR 0 through n - 1

FOR 1 through n -1

IF(arrNumbers[0]>arrNumbers[1])

temp = arrNumbers[i];

arrNumbers[i] = arrNumbers[j];

arrNumbers[j] = temp;

ENDIF

ENDFOR

ENDFOR

FOREACH <value> in <arrNumbers>

PRINT value

ENDFOREACH

END

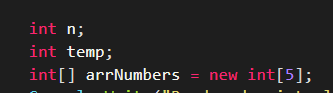
## 3.3 Implementation

\_ step 1, I’ll declare variables.

+ n to input number of cases.

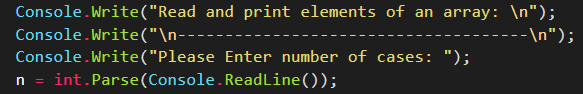
+ temp to make a temporary memory to store value during swapping values.

+ arrNumbers is a place stored values in an array and I declare it with max length is 5 index.



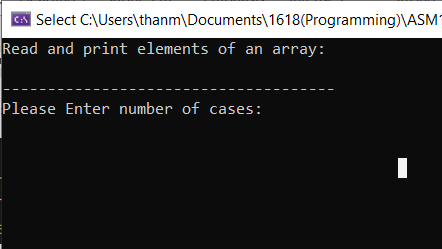
**Figure 3: Declare variables.**

\_ step 2, I’ll print some message and read input from key board



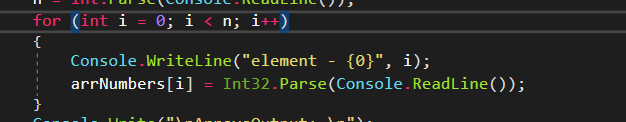
**Figure 4: Code to read n.**

+ Result for step 2: computer will print the message and ask you to enter number cases of array.



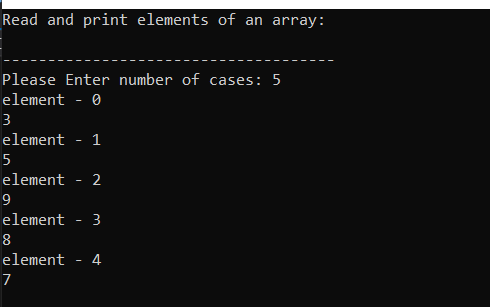
**Figure 5: Result for step 2, computer ask to input number cases of array.**

\_ step 3: After enter number cases of array, I write code to run for loop from i=0 to i = n – 1 with purpose to Enter value for each index from 0 to (n – 1).



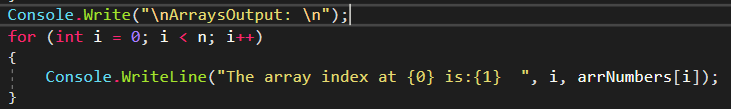
**Figure 6: Code for inputing value for each index in array.**

+ Result for step 3: Computer will allow you input each value into index. They will be stored into temporary memory.



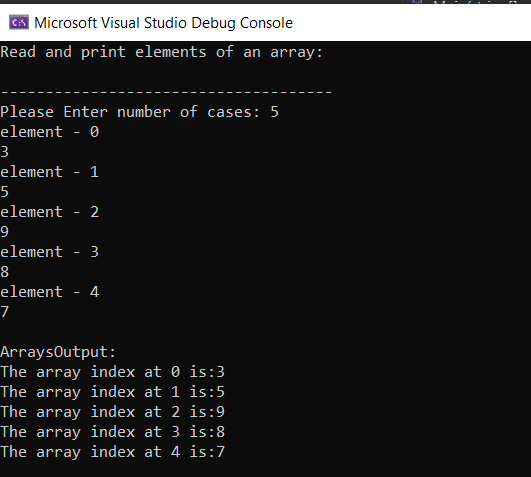
**Figure 7: Result for step 3, Input value for each index.**

\_ Step 4: This one, I’ll write another for loop to run from i = 0 to I = n – 1 to print values from index 0 to index n -1 which I had input in step 3.



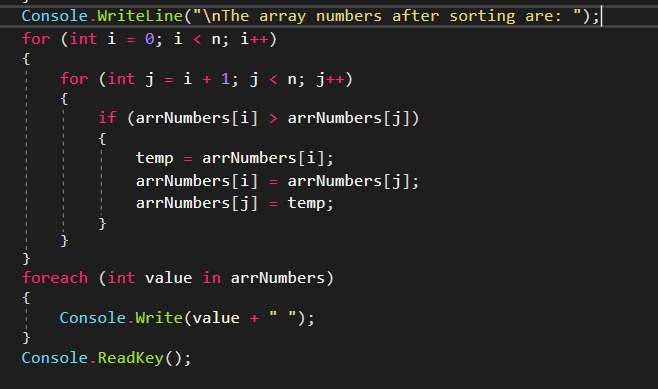
**Figure 8: Code for printing each value which I had input.**

+ Result for step 4: Computer will print value of each index which you had input in step 3.



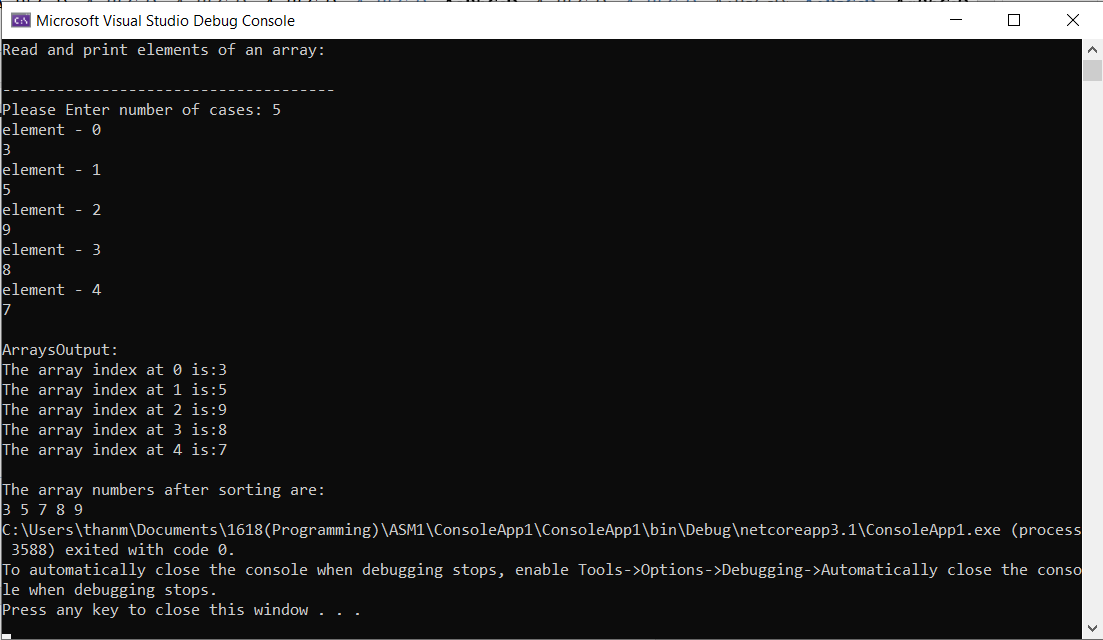
**Figure 9: Result for step 4, print value of each index.**

\_ Step 5: I’ll write two for loops. The first one for checking the first position of an array and iterates the length(n) of the array. The second loop starts with second position is i+1 position and iterates the length(n) of the array. In the second loop I write an if statement to check conditions between two values of the first position and the second position if conditions true do the task inside if statement. Foreach loop with purpose to print values after sorting in the array.



**Figure 10: Code for sorting array numbers.**

+ Result for step 5: after print values, computer will execute to sort numbers in the array follow to ascending way.



**Figure 11: Result for step 5, print values after sorting.**

## 3.4 Test Case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TEST CASE ID | TEST CASE OBJECTIVE | PREREQUISITE | TEST STEP | INPUT DATA | EXPECTED OUTPUT | ACTUAL OUTPUT | STATUS:  PASS/FAIL |
| TEST CASE 1 | INPUT n as an integer | 0 < n <= 5 | 1. Enter n  2. Enter values for each index  3. Enter | 1.4  2. 3  5  4  6 | 3  4  5  6 | 3  4  5  6 | PASS |
| TEST CASE 2 | INPUT n as an integer | 0 < n <= 5 | 1. Enter n  2. Enter values for each index  3. Enter | 1. 6  2. 3  5  9  8  7  2 | 2  3  5  7  8  9 | ERROR | PASS |
| TEST CASE 3 | INPUT n as a char | 0 < n <= 5 | 1. Enter n  2. Enter values for each index  3. Enter | 1.5  2. B  C  D  A  E | A  B C D E | A  B C D E | FAIL |
| TEST CASE 4 | INPUT n as a char | 0 < n <= 5 | 1. Enter n  2. Enter values for each index  3. Enter | 1.4  2. B  C  D  A | A  B C D | ERROR | PASS |
| TEST CASE 5 | INPUT n as an integer | 0 < n <= 5 | 1. Enter n  2. Enter values for each index  3. Enter | 1. 6  2. 3  5  9  8  7  2 | 2  3  5  7  8  9 | 2  3  5  7  8  9 | FAIL |

**Figure 12: Test Case.**

# REFERENCES

https://www.bbc.co.uk/bitesize/guides/z22wwmn/revision/1