IF 130 Programming Fundamentals

07 Review and Case Studies

Dr. Maria Irmina Prasetiyowati, S.Kom,. M.T.
Alethea Suryadibrata, S.Kom., M.Eng.
Putri Sanggabuana Setiawan, S.Kom, M.T.I.
Januar Wahjudi, S.Kom., M.Sc.
Drs Slamet Aji Pamungkas, M.Eng
Kursehi Falgenti S.Kom., M.Kom.

Sub- Course Learning Outcome:

1. **Sub-CLO 0211**: Students are able to explain the definition of algorithm, flowchart, and pseudocode (C2).
2. **Sub-CLO 0611**: Students are able to apply the problem-solving process using algorithms in the form of flowcharts and pseudocodes (C3).
3. **Sub-CLO 0612**: Students are able to draw flowcharts with selection control structures, repetition control structures, and modularization control structures (C3).
4. **Sub-CLO 0613**: Students are able to compile pseudocode with selection control structures, repetition control structures, and modularization control structures (C3).

Review

1. Definition of modular programming
2. Modular Pseudocode

Outline

1. Flowchart and Pseudocode with Selection Control Structure
2. Flowchart and Pseudocode with Repetition Control Structure
3. Flowchart and Pseudocode with Modularization
4. Case Studies

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

ALGORITHM

What is an algorithm?

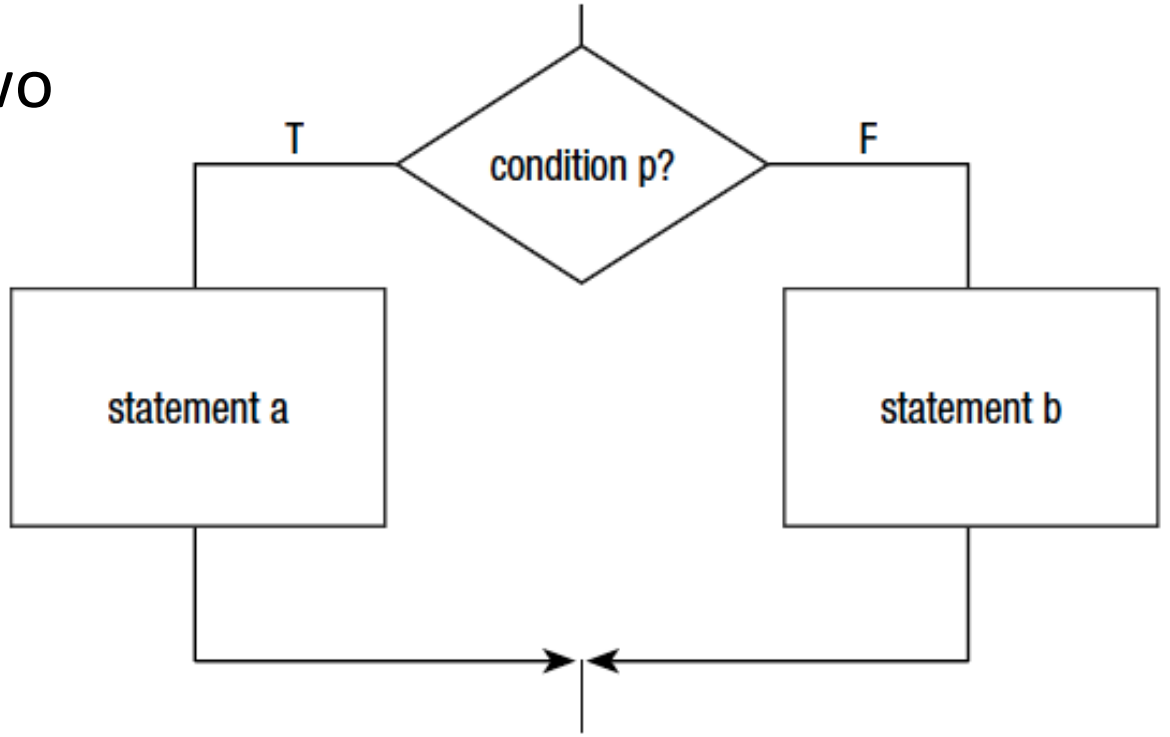
- An algorithm is like a recipe: it lists the steps involved in accomplishing a task.
- It can be defined in programming terms as a set of *detailed*, *unambiguous* and *ordered instructions* developed to describe the **processes** necessary to produce the desired **output** from a given **input**.
- Flowchart and pseudocode are both popular ways of representing algorithms.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

FLOWCHART & PSEUDOCODE WITH SELECTION CONTROL STRUCTURE

Selection Control Structure

The selection control structure can be defined as the presentation of a condition, and the choice between two actions depending on whether the condition is true or false.

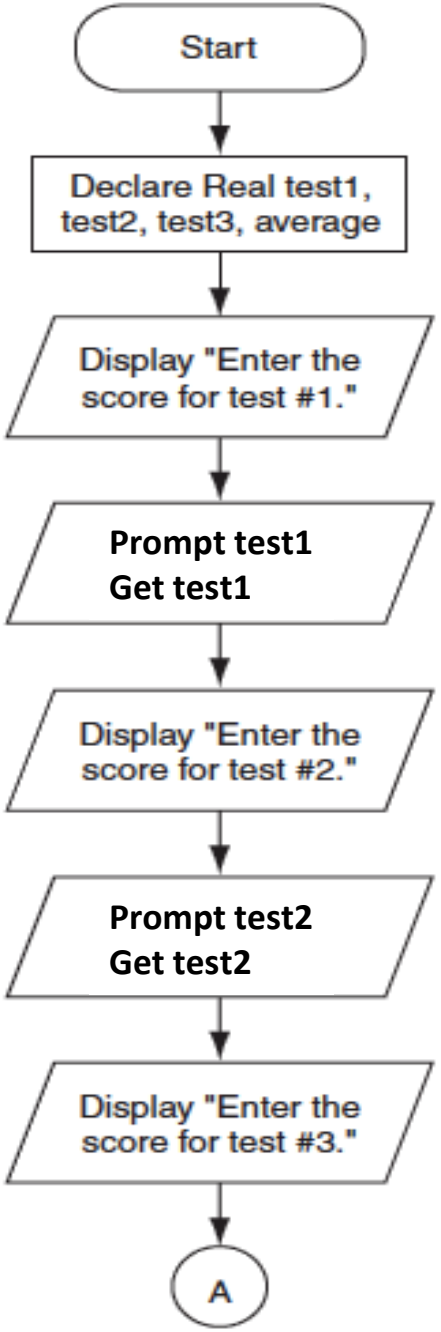


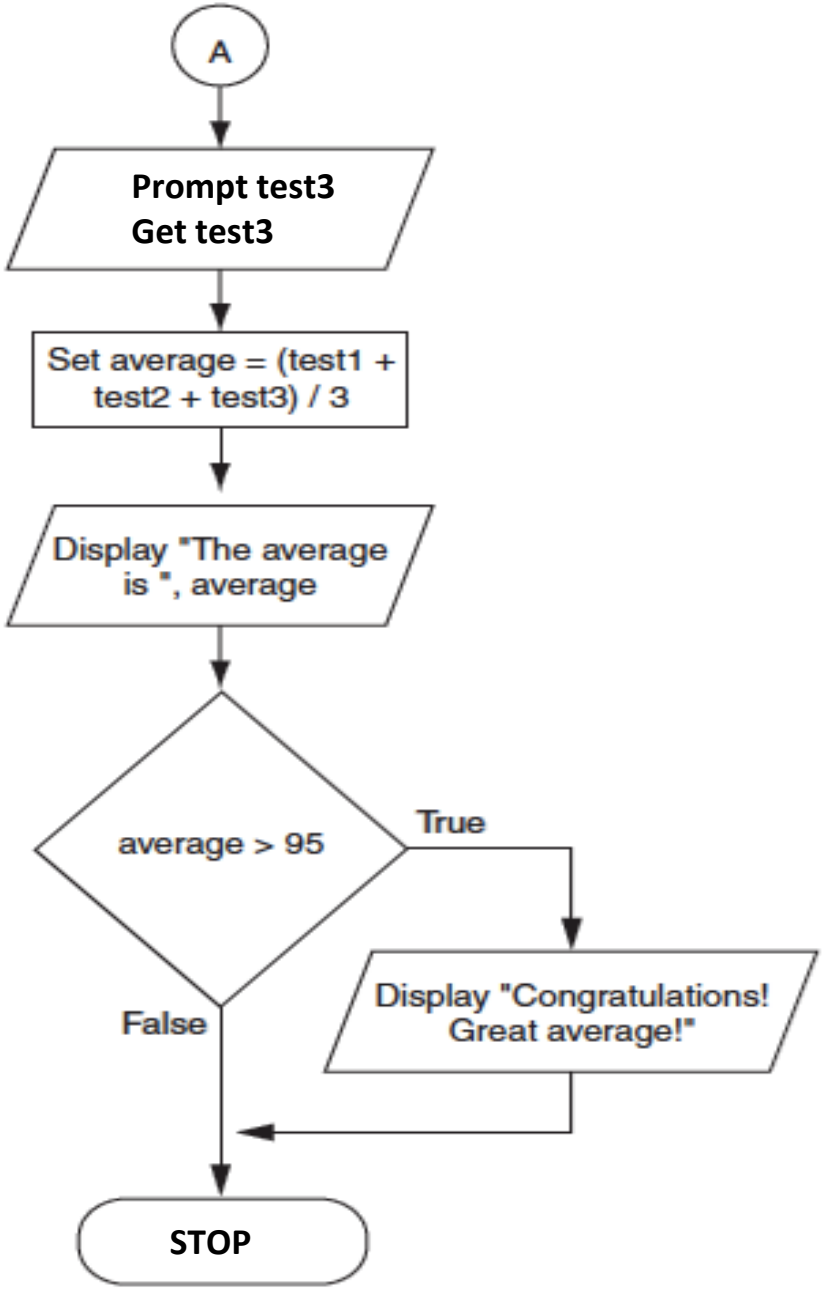
Case Study

Kathryn teaches a science class and her students are required to take three tests. She wants to write a program that her students can use to calculate their average test score. She also wants the program to congratulate the student enthusiastically if the average is greater than 95. Here is the algorithm:

1. Get the first test score
2. Get the second test score
3. Get the third test score
4. Calculate the average
5. Display the average
6. If the average is greater than 95, congratulate the user

Solution Algorithm in Flowchart





Solution Algorithm in **Pseudocode**

Program Output (with Input Shown in Bold)

```
Enter the score for test #1.
82 [Enter]
Enter the score for test #2.
76 [Enter]
Enter the score for test #3.
91 [Enter]
The average is 83
```

Program Output (with Input Shown in Bold)

```
Enter the score for test #1.
93 [Enter]
Enter the score for test #2.
99 [Enter]
Enter the score for test #3.
96 [Enter]
The average is 96
Congratulations! Great average!
```

```
1  // Declare variables
2  Declare Real test1, test2, test3, average
3
4  // Get test 1
5  Display "Enter the score for test #1."
6  Prompt test1
7  Get test1
8  // Get test 2
9  Display "Enter the score for test #2."
10 Prompt test2
11 Get test2
12 // Get test 3
13 Display "Enter the score for test #3."
14 Prompt test3
15 Get test3
16 // Calculate the average score.
17 Set average = (test1 + test2 + test3) / 3
18
19 // Display the average.
20 Display "The average is ", average
21
22 // If the average is greater than 95
23 // congratulate the user.
24 If average > 95 Then
25    Display "Congratulations! Great average!"
26 End If
27 END
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

FLOWCHART & PSEUDOCODE WITH REPETITION CONTROL STRUCTURE

While Loop

- The while loop is known as a *prestest* loop, which means it tests its condition *before* performing iteration.
- Because the test is done at the beginning of the loop, you usually have to perform some steps prior to the loop to make sure that the loop executes at least once.

Case Study

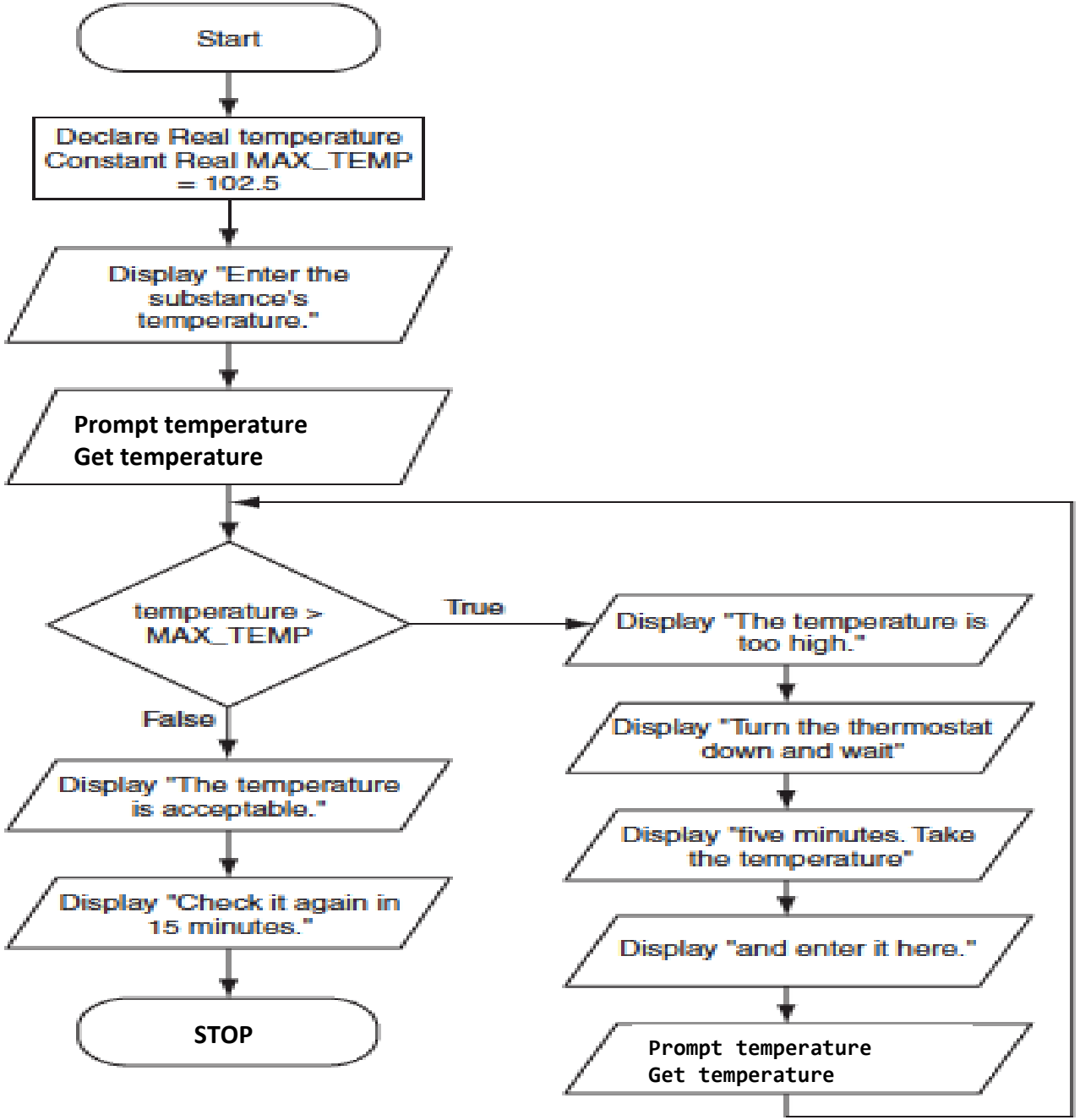
A project currently underway at Chemical Labs, Inc. requires that a substance be continually heated in a vat. A technician must check the substance's temperature every 15 minutes. If the substance's temperature does not exceed 102.5, then the technician does nothing. However, if the temperature is greater than 102.5, the technician must turn down the vat's thermostat, wait five minutes, and check the temperature again. The technician repeats these steps until the temperature does not exceed 102.5. The director of engineering has asked you to design a program that guides the technician through this process.

Here is the algorithm:

1. Get the substance's temperature.
2. Repeat the following steps as long as the temperature is greater than 102.5:
   a. Tell the technician to turn down the thermostat, wait five minutes, and check the temperature again.
   b. Get the substance's temperature.
3. After the loop finishes, tell the technician that the temperature is acceptable and to check it again in 15 minutes.

After reviewing this algorithm, you realize that steps 2(a) and 2(b) should not be performed if the test condition (temperature is greater than 102.5) is false to begin with. The `While` loop will work well in this situation, because it will not execute even once if its condition is false. Program 5-2 shows the pseudocode for the program, and Figure 5-4 shows a flowchart.

Solution Algorithm in **Flowchart**



Solution Algorithm in Pseudocode

```
1  // Variable to hold the temperature
2  Declare Real temperature
3
4  // Constant for the maximum temperature
5  Constant Real MAX_TEMP = 102.5
6
7  // Get the substance's temperature.
8  Display "Enter the substance's temperature."
9  Prompt temperature
10 Get temperature
11 // If necessary, adjust the thermostat.
12 While temperature > MAX_TEMP
13    Display "The temperature is too high."
14    Display "Turn the thermostat down and wait"
15    Display "five minutes. Take the temperature"
16    Display "again and enter it here."
17    Prompt temperature
18    Get temperature
19 ENDWHILE
20 // Remind the user to check the temperature
21 // again in 15 minutes.
22 Display "The temperature is acceptable."
23 Display "Check it again in 15 minutes."
24 END
```

Program Output (with Input Shown in Bold)

```
Enter the substance's temperature.
104.7 [Enter]
The temperature is too high.
Turn the thermostat down and wait
five minutes. Take the temperature
again and enter it here.
103.2 [Enter]
The temperature is too high.
Turn the thermostat down and wait
five minutes. Take the temperature
again and enter it here.
102.1 [Enter]
The temperature is acceptable.
Check it again in 15 minutes.
```

Program Output (with Input Shown in Bold)

```
Enter the substance's temperature.
102.1 [Enter]
The temperature is acceptable.
Check it again in 15 minutes.
```

Do-While Loop

- The Do-While loop is a *posttest* loop. This means it performs an iteration before testing its condition.
- As a result, the Do-While loop always performs at least one iteration, even if its condition is false to begin with.

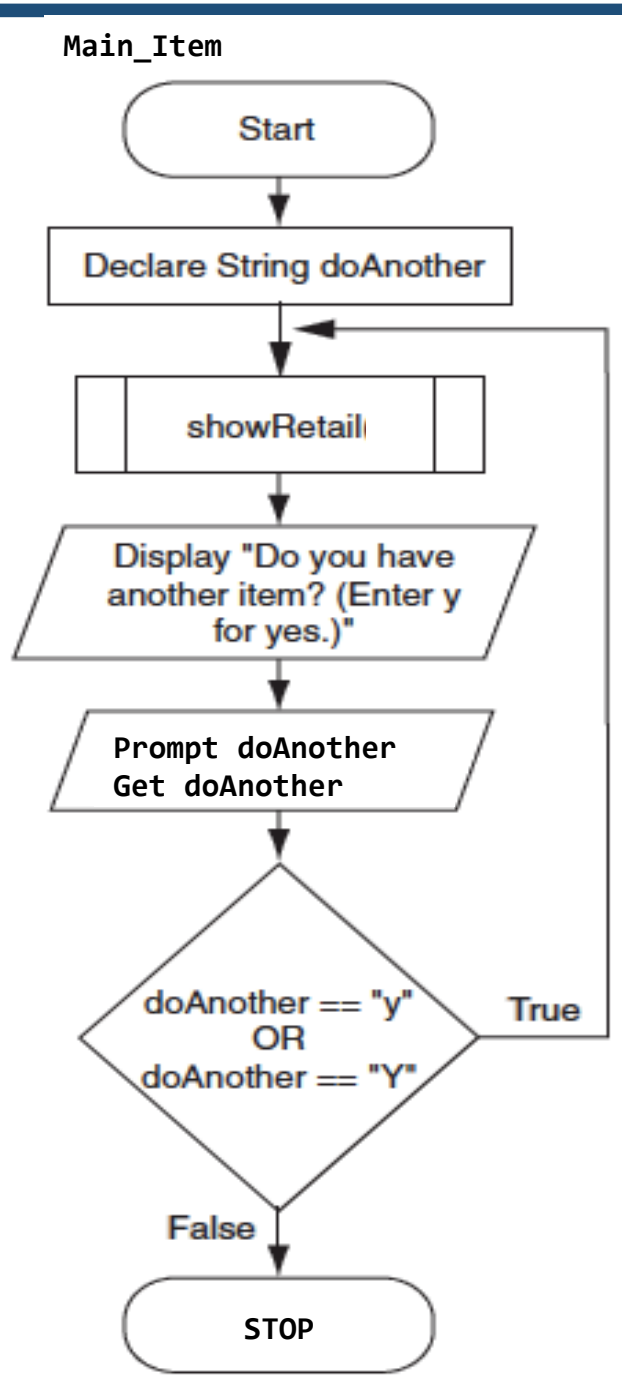
Case Study

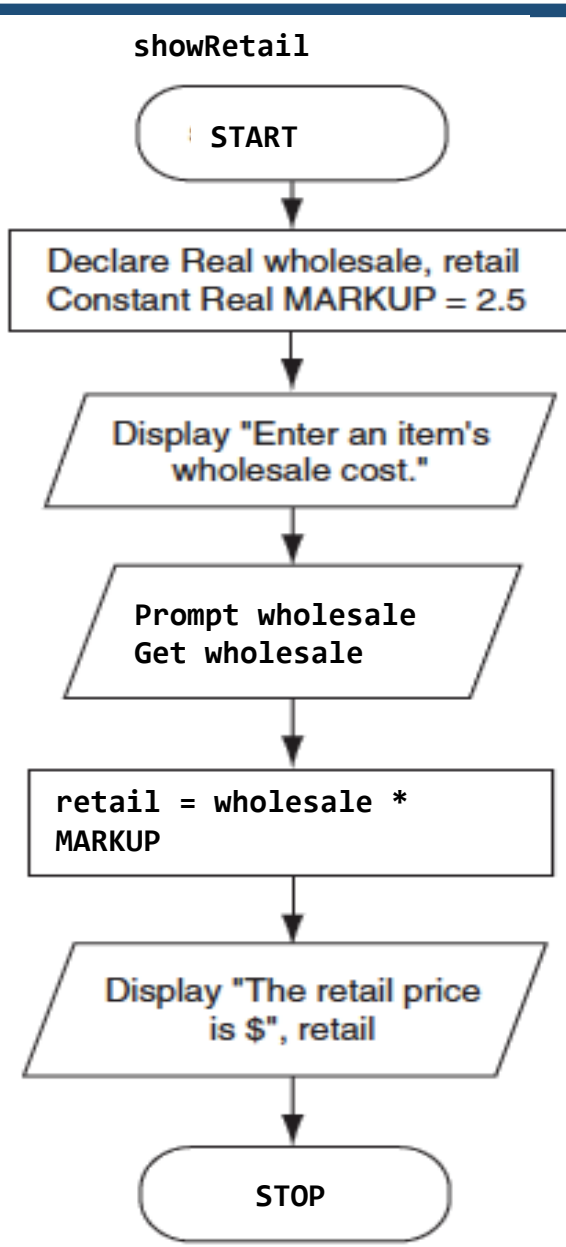
- Samantha owns an import business and she calculates the retail prices of her products with the following formula:

  Retail price = wholesale cost x 2.5

- She asked you to design a program to do this calculation for each item that she receives in a shipment. You learn that each shipment contains various numbers of items, so you decide to use a loop that calculates the price for one item, and then asks her whether she has another item. The loop will iterate as long as she indicates that she has another item.

Solution Algorithm in **Flowchart**





Solution Algorithm in **Pseudocode**

Program Output (with Input Shown in Bold)

```
Enter an item's wholesale cost.
10.00 [Enter]
The retail price is $25
Do you have another item? (Enter y for yes.)
y [Enter]
Enter an item's wholesale cost.
15.00 [Enter]
The retail price is $37.50
Do you have another item? (Enter y for yes.)
y [Enter]
Enter an item's wholesale cost.
12.50 [Enter]
The retail price is $31.25
Do you have another item? (Enter y for yes.)
n [Enter]
```

```
1  Main_Item
2     // Local variable
3     Declare String doAnother
4
5     Do
6        // Calculate and display a retail price.
7        Call showRetail()
8
9        // Do this again?
10       Display "Do you have another item? (Enter y for yes.)"
11       Prompt doAnother     Get doAnother
12    While doAnother == "y" OR doAnother == "Y"
13 End
14
15 // The showRetail module gets an item's wholesale cost
16 // from the user and displays its retail price.
17  showRetail
18    // Local variables
19    Declare Real wholesale, retail
20
21    // Constant for the markup percentage
22    Constant Real MARKUP = 2.50
23
24    // Get the wholesale cost.
25    Display "Enter an item's wholesale cost."
26    Prompt wholesale
27    Get wholesale
28    // Calculate the retail price.
29    Set retail = wholesale * MARKUP
30
31    // Display the retail price.
32    Display "The retail price is $", retail
33 End
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

FLOWCHART & PSEUDOCODE WITH MODULARIZATION

Modules

- A module is a group of statements that exist within a program for the purpose of performing a specific task
- Benefits of using Modules:
  - Simpler code
  - Code reuse
  - Better Testing
  - Faster Development
  - Easier Facilitation of Teamwork

Case Study

- Professional Appliance Service, Inc. offers maintenance and repair services for household appliances. The owner wants to give each of the company's service technicians a small handheld computer that displays step-by step instructions for many of the repairs that they perform.
- To see how this might work, the owner has asked you to develop a program that displays the following instructions for disassembling an ACME laundry dryer.

  Step 1: Unplug the dryer and move it away from the wall
  Step 2: Remove the six screws from the back of the dryer
  Step 3: Remove the dryer's back panel
  Step 4: Pull the top of the dryer straight up

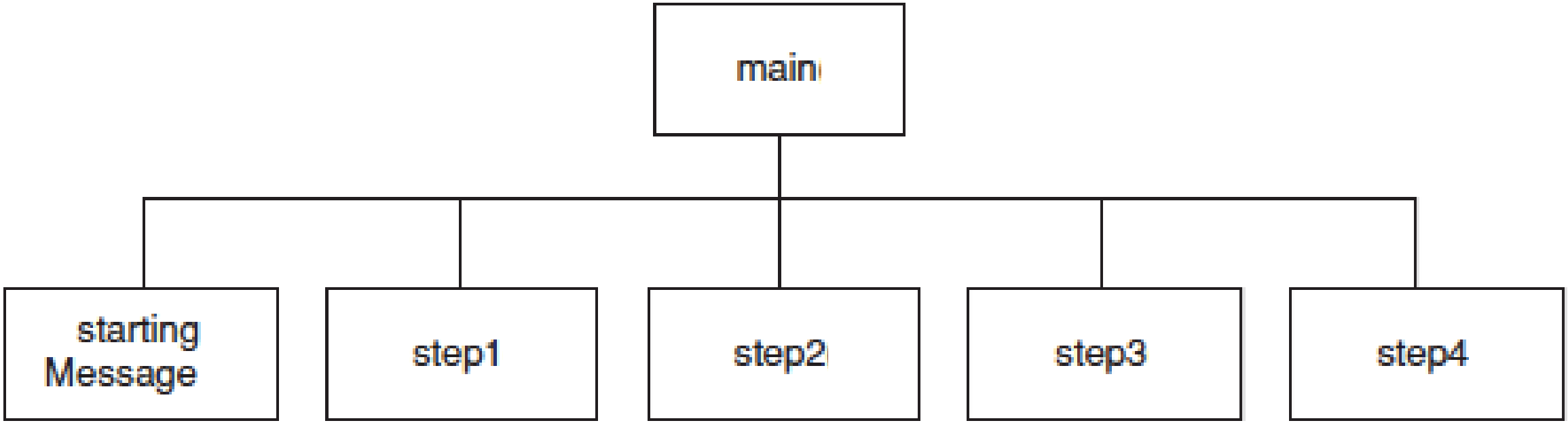
Case Study (cont.)

During your interview with the owner, you determine that the program should display the steps one at a time. You decide that after each step is displayed, the user will be asked to press a key to see the next step. Here is the algorithm for the program:

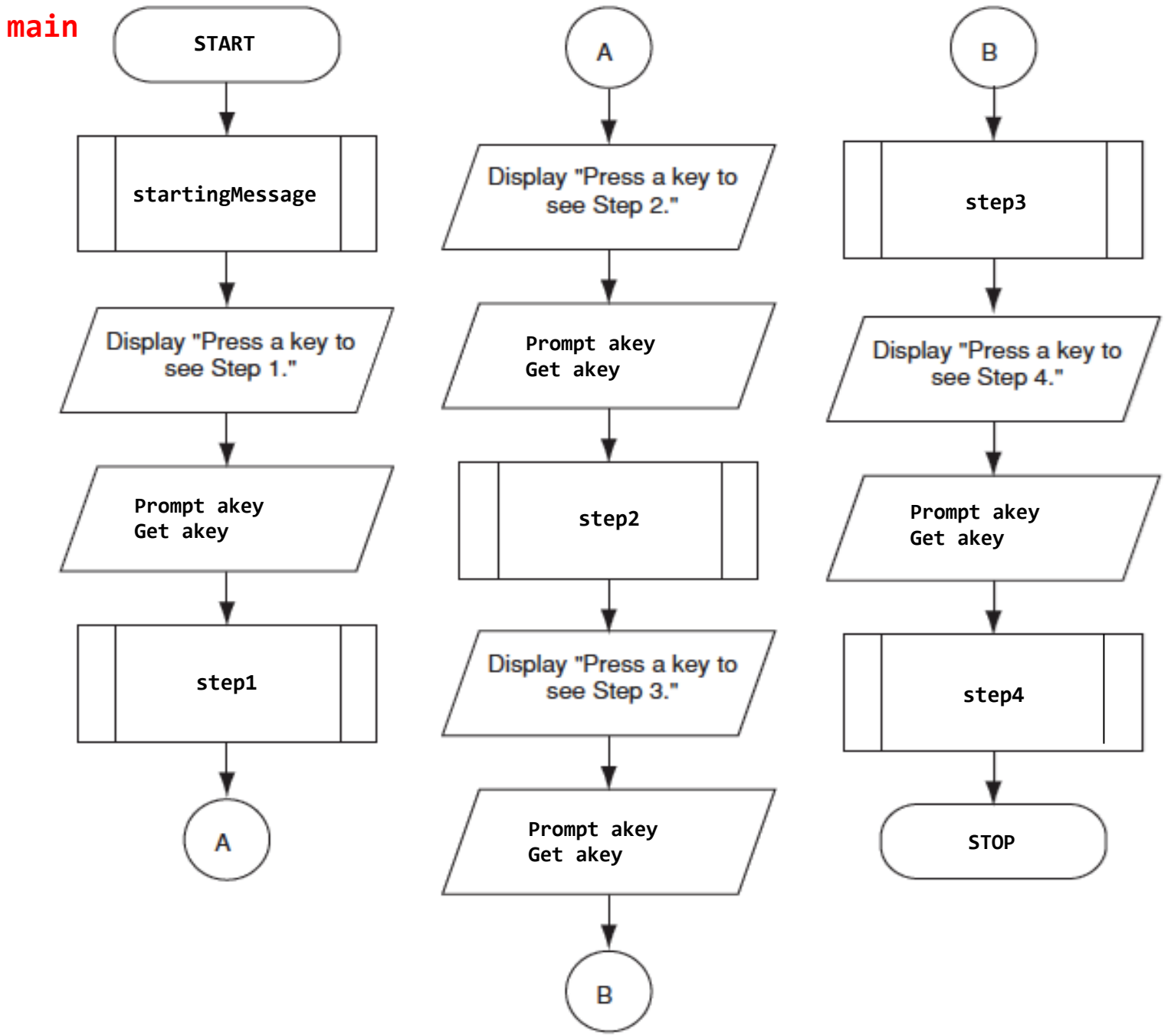
1. Display a starting message, explaining what the program does.
2. Ask the user to press a key to see Step 1.
3. Display the instructions for Step 1.
4. Ask the user to press a key to see the next step.
5. Display the instructions for Step 2.
6. Ask the user to press a key to see the next step.
7 . Display the instructions for Step 3.
8. Ask the user to press a key to see the next step.
9. Display the instructions for Step 4.

Case Study (Cont.)

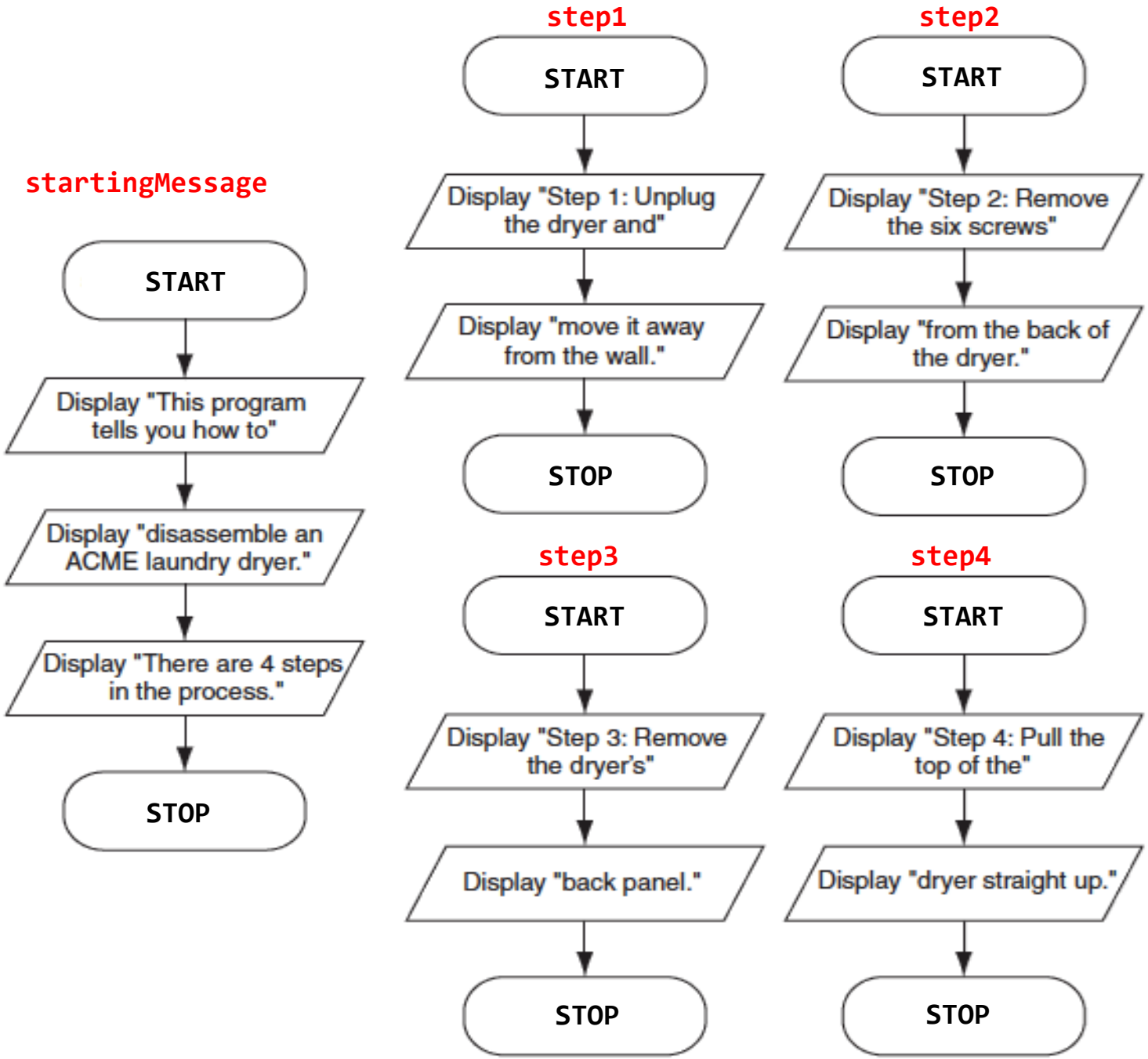
This algorithm lists the top level of tasks that the program needs to perform, and becomes the basis of the program's main module. The following figure shows the program's structure in a **hierarchy chart.**



Solution Algorithm in **Flowchart**



Solution Algorithm in **Flowchart** (Cont.)



Solution Algorithm in **Pseudocode**

```
1  main
2       // Display the starting message.
3       Call startingMessage
4       Display "Press a key to see Step 1. “
5       Prompt akey
6       Get akey
7        // Display step 1.
8       Call step1
9       Display "Press a key to see Step 2.”
10      Prompt akey
11      Get akey
12       // Display step 2.
13      Call step2
14      Display "Press a key to see Step 3. “
15      Prompt akey
16      Get akey
17      // Display step 3.
18      Call step3
19      Display "Press a key to see Step 4 ."
20      Prompt akey
21      Get akey
22       // Display step 4.
23      call step4
24 End
25
26 // The startingMessage module displays
27 // the program’s starting message
28 startingMessage
29       Display "This program tell you how to"
30       Display “disassemble an ACME laundry dryer.”
31       Display "There are 4 steps in the process . “
32 End
33
34 // The step1 module displays the instructions for Step 1.
35
36 step1
37       Display “Step 1: unplug the dryer and”
38       Display “move it away from the wall."
39 End
40
```

Solution Algorithm in **Pseudocode** (Cont.)

```
41 // The step2 module displays the instructions for step 2 .
42
43 step2
44       Display “step 2: Remove the six screws”
45       Display “from the back of the dryer."
46 End
47
48 // The step3 module displays the instructions for Step 3 .
49
50 step3
51       Display “step 3: Remove the dryer's"
52       Display "back panel. "
53 End
54
55 // The step4 module displays the instructions for Step 4 .
56
57 step4
58       Display "Step 4: Pull the top of the"
59       Display "dryer straight up.”
60 End
```

Program Output

```
This program tells you how to
disassemble an ACME laundry dryer.
There are 4 steps in the process.
Press a key to see Step 1.
[Enter]
Step 1: Unplug the dryer and
move it away from the wall.
Press a key to see Step 2.
[Enter]
Step 2: Remove the six screws
from the back of the dryer.
Press a key to see Step 3.
[Enter]
Step 3: Remove the dryer's
back panel.
Press a key to see Step 4.
[Enter]
Step 4: Pull the top of the
dryer straight up.
```

REFERENCES

1. Gaddis, Tony, 2019, Starting out with programming logic & design, Fifth edition, Pearson Education, Inc.
2. Robertson, Lesley Anne, 2007, Simple Program Design A Step-by-Step Approach, Fith Edition, Thomson Learning, Inc.
3. Informatics study program slides, 2023, Fundamentals of Programming, Universitas Multimedia Nusantara.

Visi

Menjadi Program Studi Strata Satu Informatika **unggulan** yang menghasilkan lulusan **berwawasan internasional** yang **kompeten** di bidang Ilmu Komputer (*Computer Science*), **berjiwa wirausaha** dan **berbudi pekerti luhur**.

INFORMATIKA
UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Misi

1. Menyelenggarakan pembelajaran dengan teknologi dan kurikulum terbaik serta didukung tenaga pengajar profesional.
2. Melaksanakan kegiatan penelitian di bidang Informatika untuk memajukan ilmu dan teknologi Informatika.
3. Melaksanakan kegiatan pengabdian kepada masyarakat berbasis ilmu dan teknologi Informatika dalam rangka mengamalkan ilmu dan teknologi Informatika.