IF 130 Programming Fundamentals

06 The Modularized Control Structure pseudocode

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:

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 flowchart

Outline

1. Definition of modular programming
2. Modular Pseudocode
3. Modular desk checking
4. Exercises

Modularization

- As the complexity of the programming problems increases, however, it becomes more and more difficult to consider the solution as a whole.
- When presented with a complex problem, you may need to **divide** the problem into smaller parts.
- Modularization is the process of **dividing a problem** into separate tasks, each with a single purpose.

Benefits of Modular Design

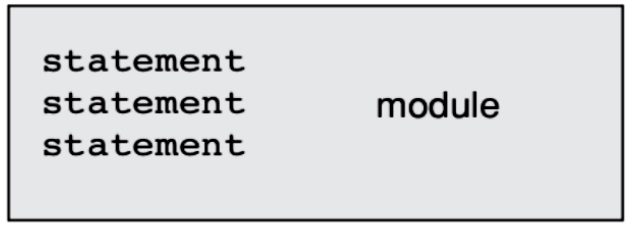
- *Ease of understanding*
  - Each module should perform just one function.
- *Reusable code*
  - Modules used in one program can also be used in other programs.
- *Elimination of redundancy*
  - Using modules can help to avoid the repetition of writing out the same segment of code more than once.
- *Efficiency of maintenance*
  - Each module should be self-contained and have little or no effect on other modules within the program

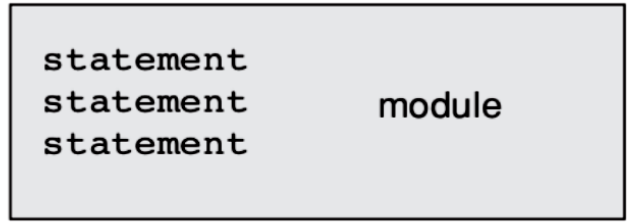
Benefits of Modular Design

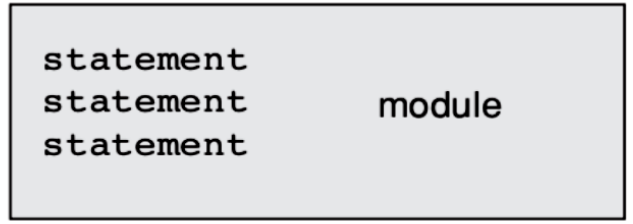
This program is one long, complex sequence of statements.

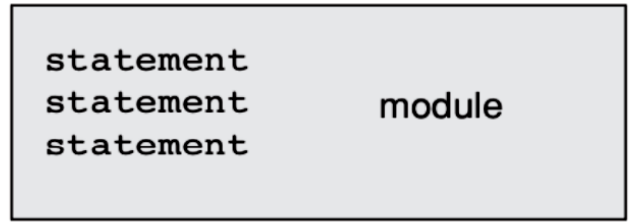
```
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
```

In this program the task has been divided into smaller tasks, each of which is performed by a separate module.









Module Names

- A module's name should be **descriptive** enough so that anyone reading your code can reasonably guess what the module does.
- Because modules **perform actions**, most programmers prefer to use **verbs** in module names. For example, a module that calculates gross pay might be named **calculateGrossPay**.
- Module names **cannot contain spaces**, cannot typically contain punctuation characters, and usually cannot begin with a number.

Example 1: Process three characters

Design an algorithm that will prompt a terminal operator for three characters, accept those characters as input, sort them into **ascending** and output them to the screen. The algorithm is to continue to read characters until ‘XXX’ is entered.

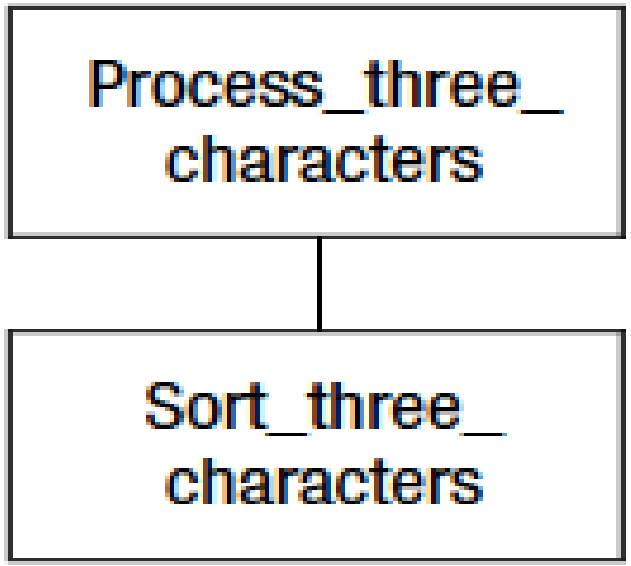
Example 1: Process three characters

Defining diagram

Input	Processing	Output
char_1 char_2 char_3	Prompt for characters Accept three characters Sort three characters Output three characters	char_1 char_2 char_3

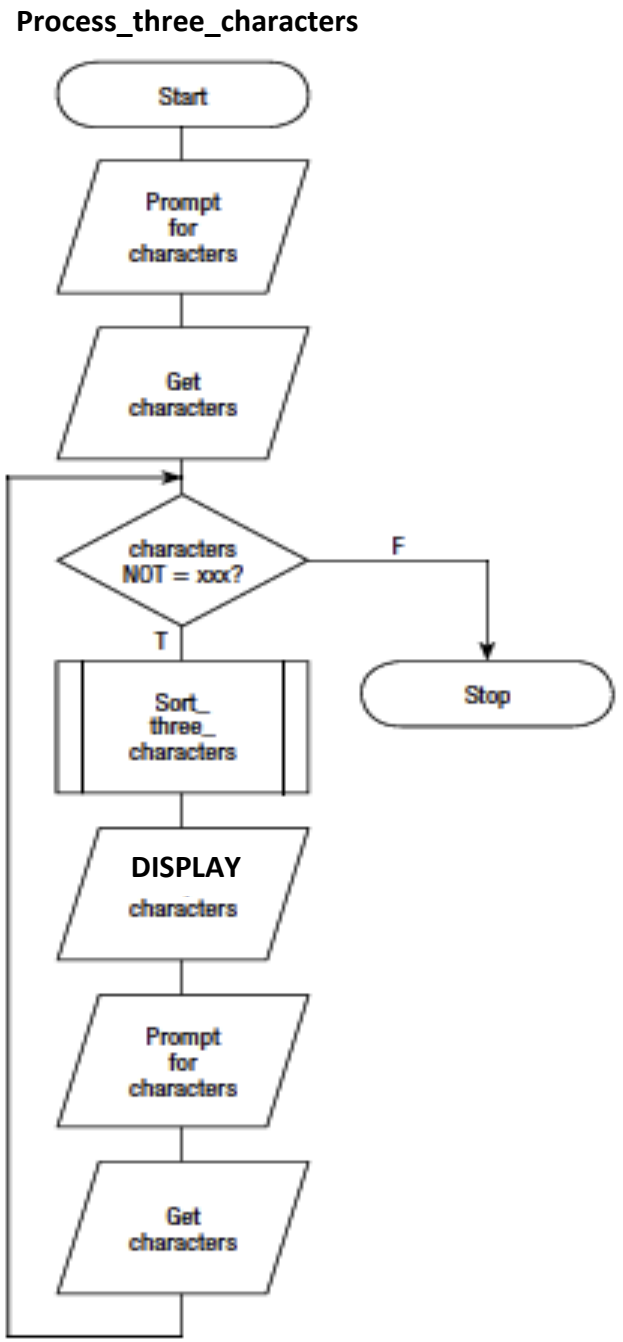
Example 1: Process three characters

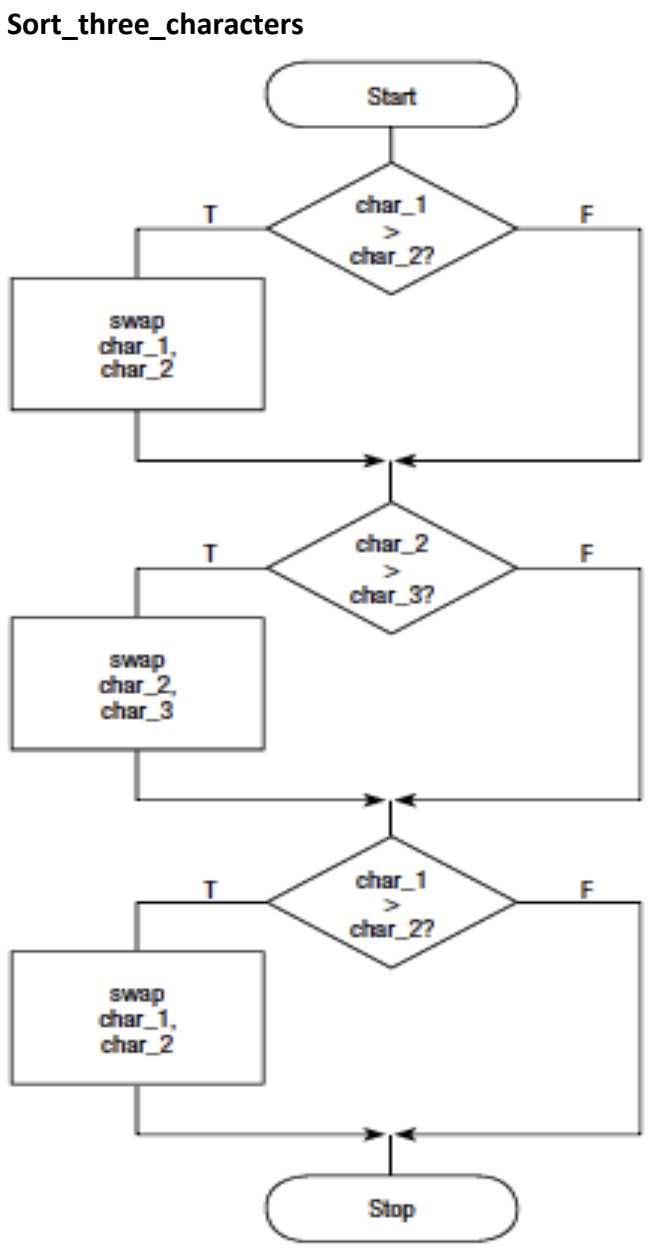
Hierarchy Chart: Solution Algorithm using **a module**



Example 1: Process three characters

Solution Algorithm using a **module**

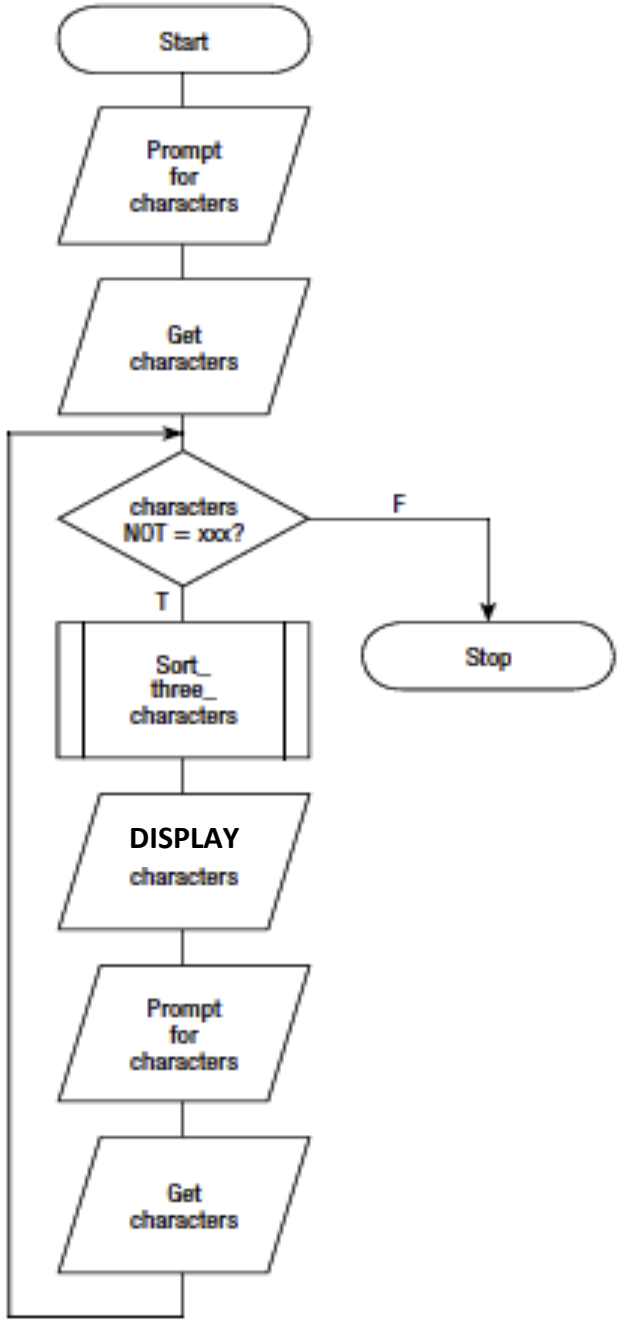




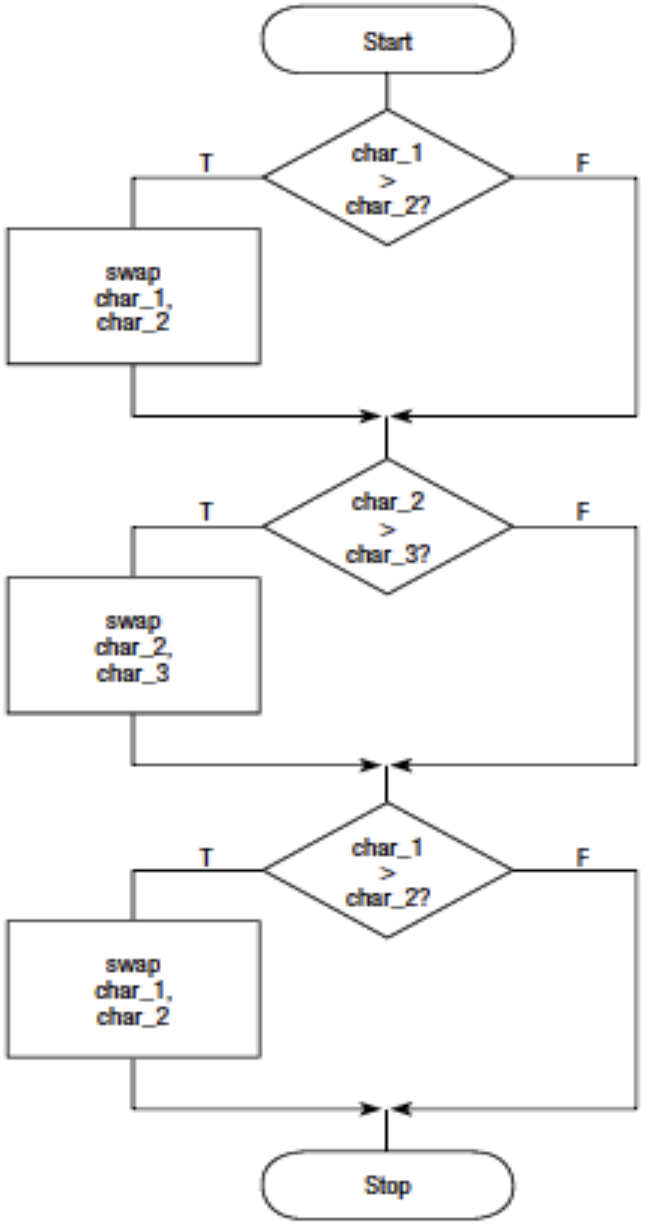
```
Process_three_characters
    Prompt the operator for char_1, char_2, char_3
    Get char_1, char_2, char_3
    WHILE     NOT  (char_1 = 'X' AND char_2 = 'X' AND char_3 = 'X')
        Sort_three_characters
        Output to the screen char_1, char_2, char_3
        Prompt operator for char_1, char_2, char_3
        Get char_1, char_2, char_3
    ENDWHILE
END

Sort_three_characters
    IF char_1 > char_2 THEN
        temp = char_1
        char_1 = char_2
        char_2 = temp
    ENDIF
    IF char_2 > char_3 THEN
        temp = char_2
        char_2 = char_3
        char_3 = temp
    ENDIF
    IF char_1 > char_2 THEN
        temp = char_1
        char_1 = char_2
        char_2 = temp
    ENDIF
END
```

Process_three_characters

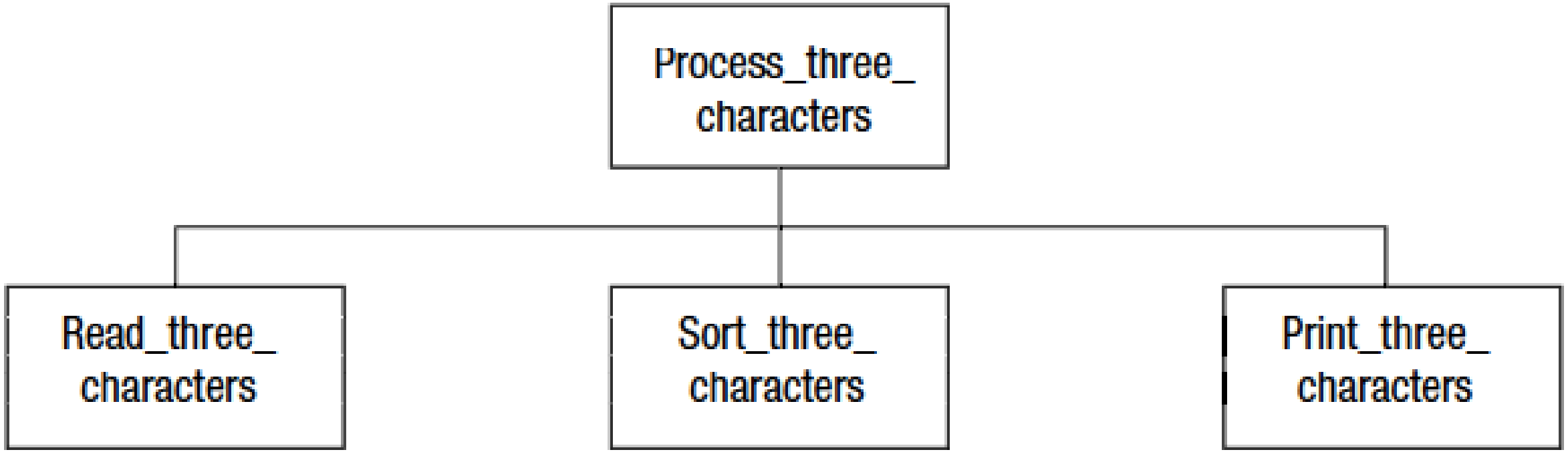


Sort_three_characters



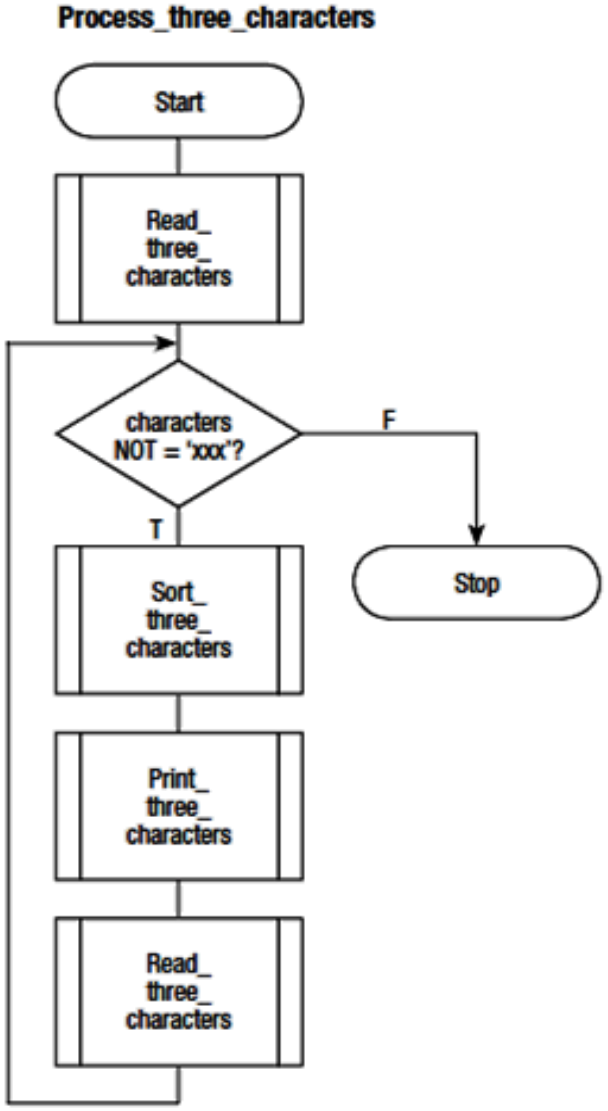
Example 1: Process three characters

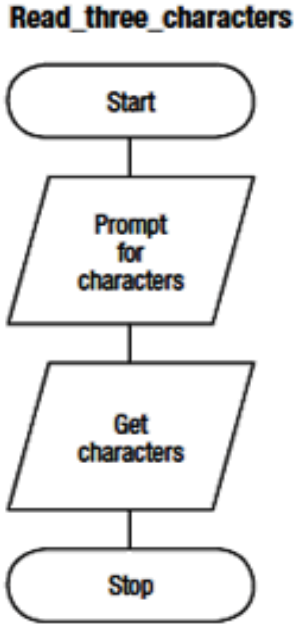
Hierarchy Chart: Solution Algorithm using **3 modules**

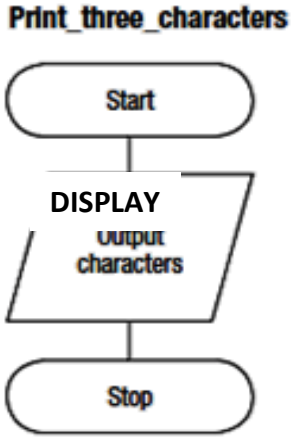


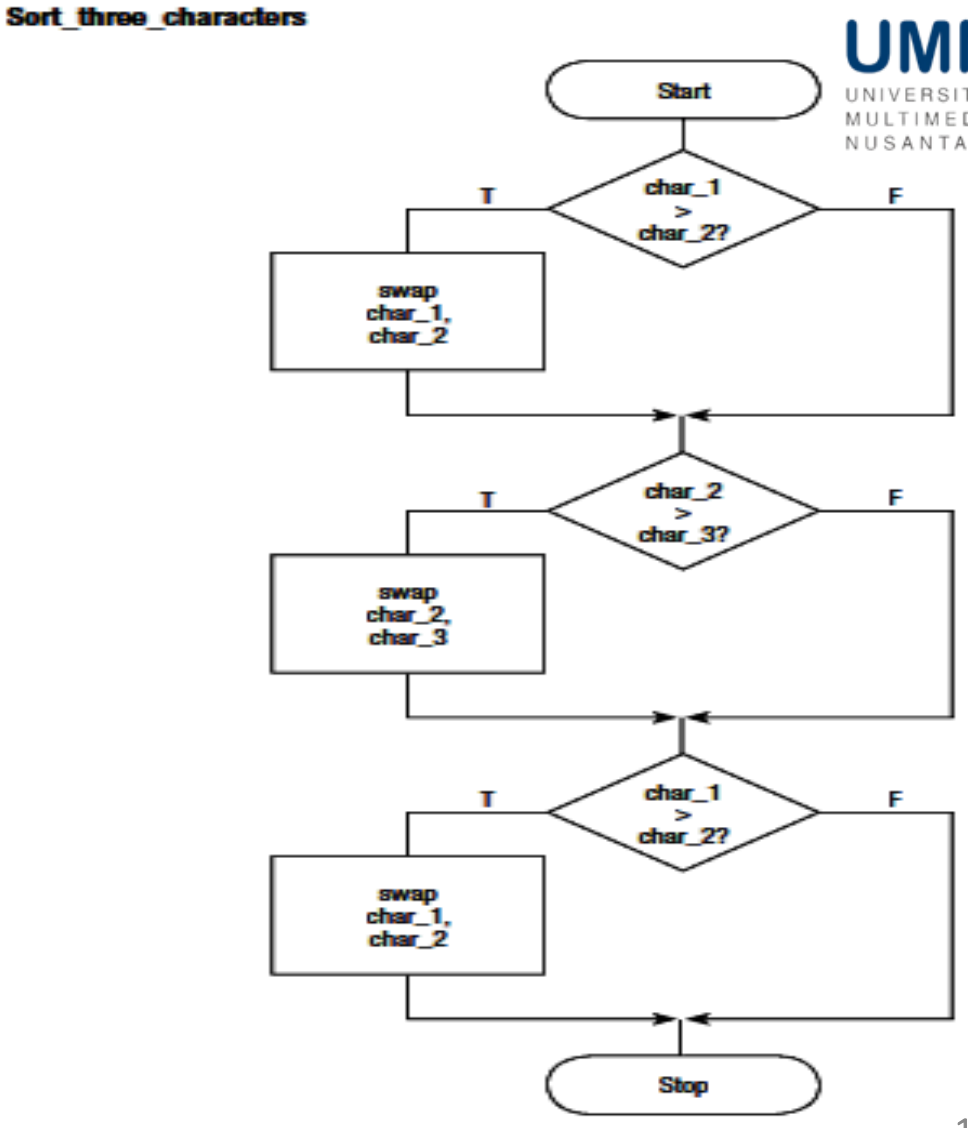
Example 1: Process three characters

Solution Algorithm using **3 modules**





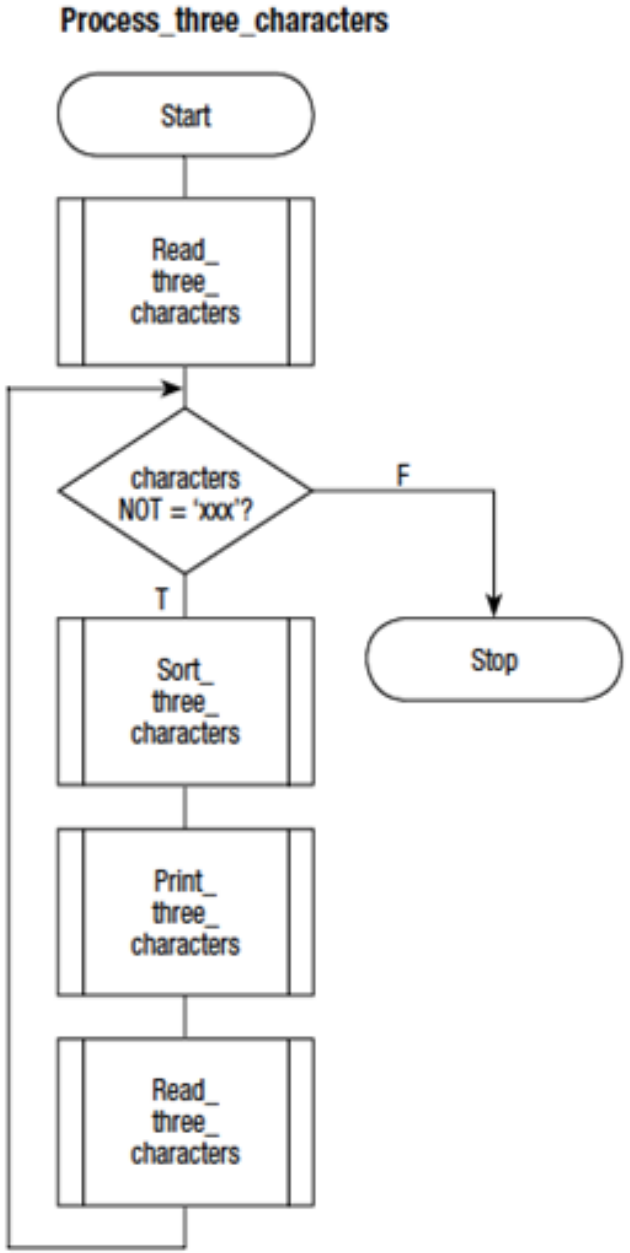


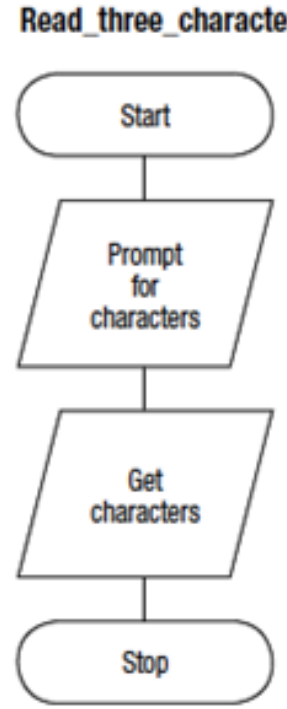


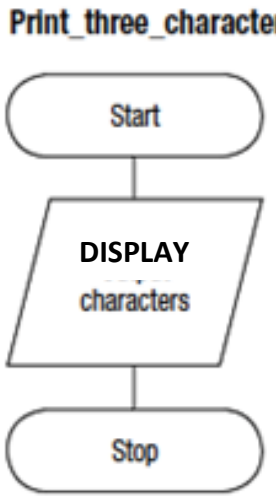
```
Process_three_characters
    Read_three_characters
    WHILE    NOT  (char_1 = 'X' AND char_2 = 'X' AND char_3 = 'X')
        Sort_three_characters
        Print_three_characters
        Read_three_characters
    ENDWHILE
END

Read_three_characters
    Prompt the operator for char_1, char_2, char_3
    Get char_1, char_2, char_3
END

Print_three_characters
    DISPLAY  to the screen char_1, char_2, char_3
END
```

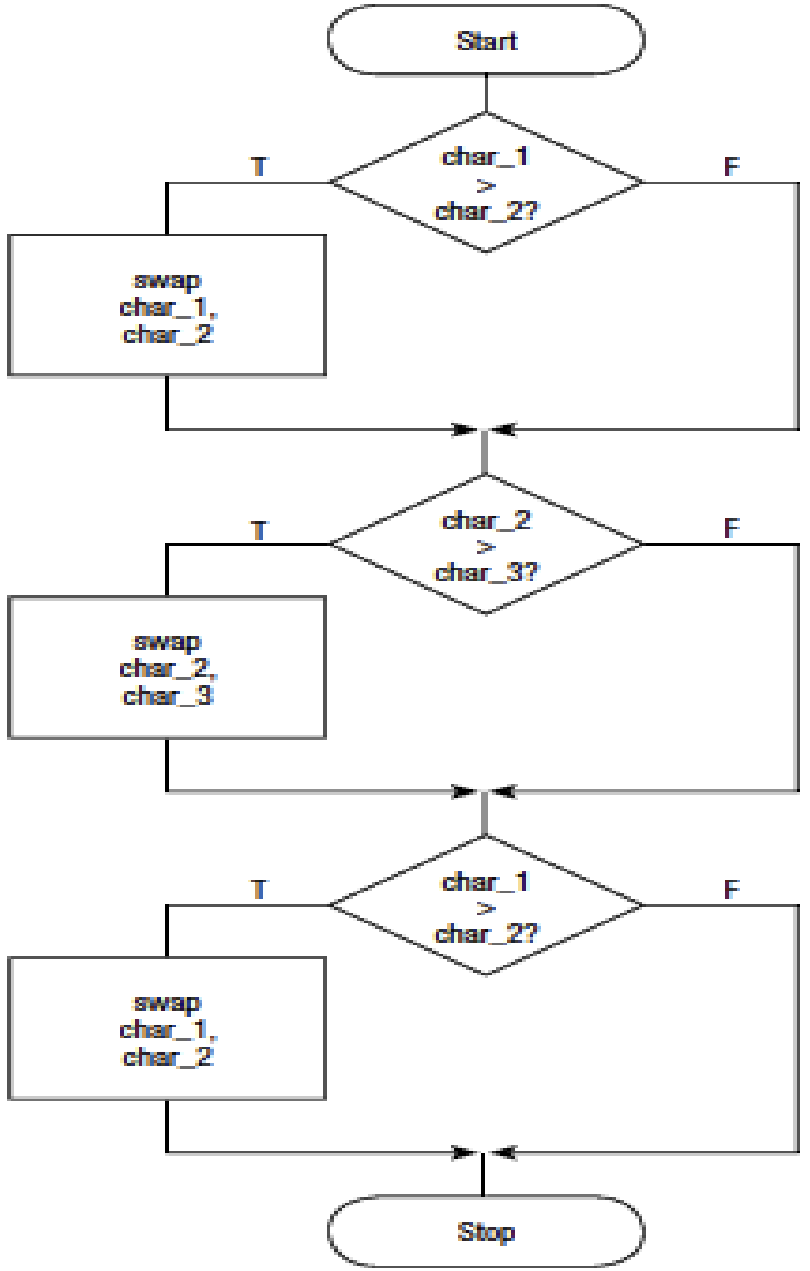






```
Sort_three_characters
    IF char_1 > char_2 THEN
        temp = char_1
        char_1 = char_2
        char_2 = temp
    ENDIF
    IF char_2 > char_3 THEN
        temp = char_2
        char_2 = char_3
        char_3 = temp
    ENDIF
    IF char_1 > char_2 THEN
        temp = char_1
        char_1 = char_2
        char_2 = temp
    ENDIF
END
```

Sort_three_characters



UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

COMMUNICATION BETWEEN MODULES

Communication Between Modules

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- When designing solution algorithms, it is necessary to consider not only the division of the problem into modules, but also the **flow of information** between the modules.
- The fewer and simpler the communications between modules, the easier it is to understand and maintain one module without reference to other modules.
- This flow of information, called '**intermodule communication**', can be accomplished by the **scope of the variable** (local or global data) or the **passing of parameters**.

Communication Between Modules

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- **Scope of a variable**
  - Is the portion of a program in which that variable has been defined and to which it can be referenced.
- **Global data**
  - Is data that can be used by all the modules in a program.
  - The scope of a global variable is the whole program
- **Local data**
  - Variables that are defined within a submodule
  - The scope of a local variable is limited to the execution of the single submodule in which it is defined

Passing Parameters

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- A particularly **efficient** method of intermodule communication is the ***passing of parameters*** or arguments between modules.
- **Parameters** are simply **data items** transferred from a **calling module** to its **subordinate module** at the time of calling.
- To pass parameters between modules, 2 things must happen:
  - The **calling module** must name the **parameters** that it wants to pass to the submodule, at the time of calling.
  - The **submodule** must be able to **receive** those parameters and return them to the calling module, if required.

Passing Parameter (cont.)

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- In pseudocode and most programming languages, when a calling module wants to pass parameters to a submodule, it simply **list the parameters**, enclosed in **parentheses**, beside the name of the submodule.

```
Print_page_headings (pageCount, lineCount)
```

- The submodule must be able to receive those parameters, so it, too, lists the parameters that it expects to receive, enclosed in parentheses, beside the submodule name when it is defined.

```
Print_page_headings (pageNumber, lineNumber)
```

Value and Reference Parameters

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- Parameters may have one of three functions:
  1. To pass information from a calling module to a subordinate module.
  2. To pass information from a subordinate module to its calling module.
  3. To fulfil a two-way communication role.

Value and Reference Parameters

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- Value parameters
  - Pass a **copy** of the value of a parameter from one module to another.
  - This form of parameter passing is called '**passing by value**'.
- Reference parameters
  - Pass the **memory address** of a parameter from one module to another.
  - This form of parameter passing is called '**passing by reference**'.
- The **requirements** of the program will **determine** whether a parameter is passed by value or by reference.

Example 1: Calculate percentage value

Design an algorithm that will receive a fraction in the form of a numerator and a denominator, convert that fraction to a percentage and display the result. Your program is to use a module to calculate the percentage.

Example 1: Calculate percentage value

Defining diagram

Input	Processing	Output
numerator denominator	Get numerator, denominator Convert fraction to percentage Display percentage	percentage

Example 1: Calculate percentage value

Solution algorithm

```
Calculate_percentage_value
    Prompt for numerator, denominator
    Get numerator, denominator
    Convert_fraction_value (numerator, denominator, percentage)
    IF percentage NOT = 0 THEN
        Output to screen, percentage, '%'
    ELSE
        Output to screen 'invalid fraction'
    ENDIF
END

Convert_fraction_value (numerator, denominator, calculatedPercentage)
    IF denominator NOT = 0
        calculatedPercentage = numerator / denominator * 100
    ELSE
        calculatedPercentage = 0
    ENDIF
END
```

Example 2: Increment two counters

Design an algorithm that will increment two counters from 1 to 10 and then output those counters to the screen. Your program is to use a module to increment the counters.

Example 2: Increment two counters

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Defining diagram

Input	Processing	Output
counter1 counter2	Increment counters Output counters	counter1 counter2

Example 2: Increment two counters

Solution algorithm

```
Increment_two_counters
    Set counter1, counter2 to zero
    FOR I = 1 to 10
        Increment_counter (counter1)
        Increment_counter (counter2)
        DISPLAY  to the screen counter1, counter2
    ENDFOR
END

Increment_counter (counter)
    counter = counter + 1
END
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

HIERARCHY CHART
&
PARAMETERS

Hierarchy charts and parameters

- Parameters that pass between modules can be incorporated into a hierarchy chart or structure chart using the following symbols:
- **Data parameters** contain the **actual variables or data items** that will be passed as parameters between modules.
- **Status parameters** act as program **flags** and should contain just one of two values: **true or false**.

For data parameters

For status parameters

Example : Calculate employee's pay

A program is required by a company to read an employee's number, pay rate and the number of hours worked in a week. The program is then to validate the pay rate field and the hours work field and, if valid, compute the employee's weekly pay and then print it and the input data.

Validation: According to the company's rules, the maximum hours an employee can work per week is 60 hours, and the maximum hourly rate is $25.00 per hour. If the hours worked field or the hourly rate field is out of range, the input data and an appropriate message are to be printed and the employee's weekly pay is not to be calculated.

Weekly pay calculation: Weekly pay is calculated as hours worked times pay rate. If more than 35 hours are worked, payment for the overtime hours worked is calculated at time-and-a-half.

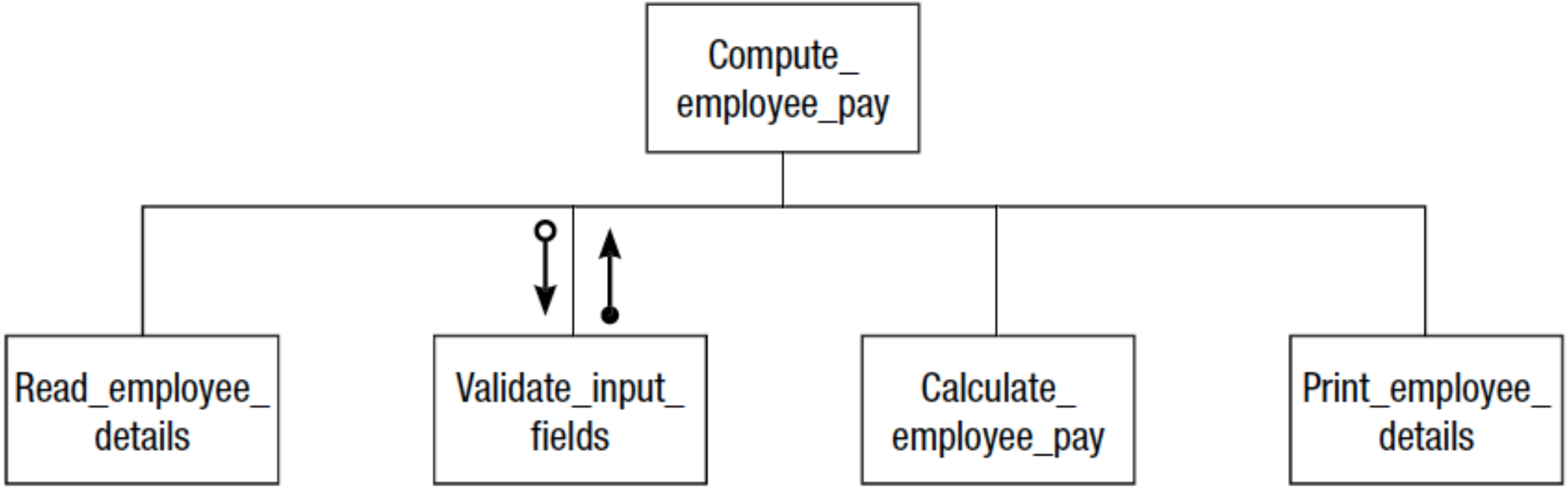
Example : Calculate employee's pay

Defining diagram

Input	Processing	Output
emp_no pay_rate hrs_worked	Read employee details Validate input fields Calculate employee pay Print employee details	emp_no pay_rate hrs_worked emp_weekly_pay error_message

Example : Calculate employee's pay

Hierarchy chart



Example : Calculate employee's pay

Solution algorithm

```
Compute_employee_pay
    Read_employee_details
    WHILE       more records
        Validate_input_fields (pay_rate, hrs_worked, validInput)
        IF validInput THEN
            Calculate_employee_pay
            Print_employee_details
        ELSE
            Print emp_no, pay_rate, hrs_worked, error_message
        ENDIF
        Read_employee_details
    ENDWHILE
END
```

```
Read_employee_details
    Read emp_no, pay_rate, hrs_worked
END

Validate_input_fields (payRate, hrsWorked, validInput)
    set validInput to true
    Set error_message to blank
    IF payRate > $25 THEN
        error_message = 'Pay rate exceeds $25.00'
        validInput = false
    ENDIF
    IF hrsWorked > 60 THEN
        error_message = 'Hours worked exceeds 60'
        validInput = false
    ENDIF
END

Calculate_employee_pay
    IF hrs_worked <= 35 THEN
        emp_weekly_pay = pay_rate * hrs_worked
    ELSE
        overtime_hrs = hrs_worked – 35
        overtime_pay = overtime_hrs * pay_rate * 1.5
        emp_weekly_pay = (pay_rate * 35) + overtime_pay
    ENDIF
END

Print_employee_details
    Print emp_no, pay_rate, hrs_worked, emp_weekly_pay
END
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Practice 1

Design an algorithm **in modular pseudocode** that will receive two integer items from a terminal operator, and display to the screen their sum, difference, product, and quotient. Note that the quotient calculation (first integer divided by second integer) is only to be performed if the second integer does not equal zero.

Practice 2

Design an algorithm **in modular pseudocode** that will prompt an operator for a student's serial number and the student's exam score out of 100. Your program is then to match the exam score to a letter grade and print the grade to the screen. Calculate the letter grade as follows:

Exam score	Assigned grade
90 and above	A
80–89	B
70–79	C
60–69	D
below 60	F

Practice 3

Design an algorithm **in modular pseudocode** that will prompt a terminal operator for the price of an article and a pricing code. Your program is then to calculate a discount rate according to the pricing code and print to the screen the original price of the article, the discount amount, and the new discounted price. Calculate the pricing code and accompanying discount amount as follows:

If the pricing code is Z, the words 'No discount' are to be printed on the screen. If the pricing code is not H, F, T, Q, or Z, the words 'Invalid pricing code' are to be printed.

Pricing code	Discount rate
H	50%
F	40%
T	33%
Q	25%
Z	0%

Practice 4

Design an algorithm that will read a file of sales volume records and print a report showing the sales commission owing to each salesperson. Each input record contains salesperson number, name and that person's volume of sales for the month. The commission rate varies according to sales volume, as follows:

On sales volume ($) of	Commission rate (%)
$0.00–$200.00	5
$200.01–$1000.00	8
$1000.01–$2000.00	10
$2000.01 and above	12

The calculated commission is an accumulated amount according to the sales volume figure. For example, the commission owing for a sales volume of $1200.00 would be calculated as follows:

Commission = (200 * 5%) + ((1000 – 200) * 8%) + ((1200 – 1000) * 10%))

Your program is to print the salesperson's number, name, volume of sales and calculated commission, with the appropriate column headings.

NEXT WEEK'S OUTLINE

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

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.