# IF 130 Programming Fundamentals

## 05 The modularized control structure flowchart

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

- A loop is a **group of instructions** the computer executes repeatedly while some loop repetition condition remains true.
- 2 kinds of repetition
  1. Sentinel-controlled repetition
  2. Counter-controlled repetition

# Outline

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

# Modularization

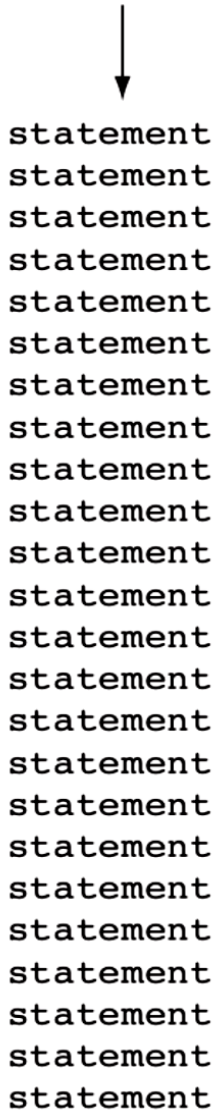
- As the complexity of the programming problems increases, however, it becomes more and more difficult to consider the solution as a whole.
- When presented with a complex problem, you may need to **divide** the problem into smaller parts.
- Modularization is the process of **dividing a problem** into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

# Benefits of Modular Design

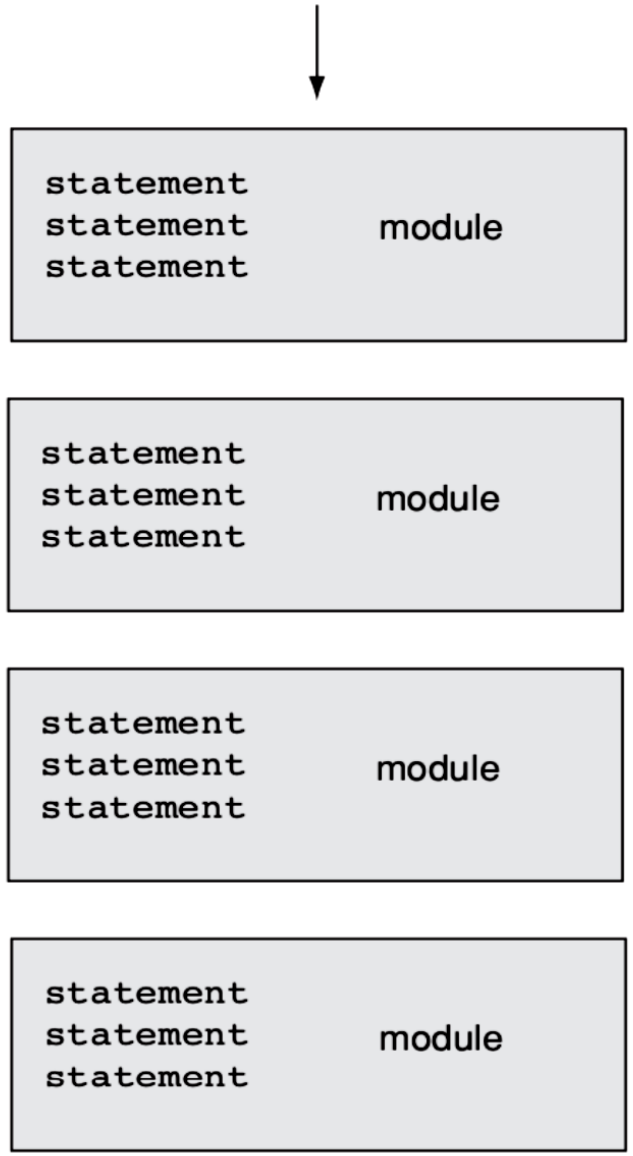
- *Ease of understanding*
  - Each module should perform just one function.
- *Reusable code*
  - Modules used in one program can also be used in other programs.
- *Elimination of redundancy*
  - Using modules can help to avoid the repetition of writing out the same segment of code more than once.
- *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.




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



# Module Names

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

- 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.

# Flowcharts and Modules

- When designing a modular solution to a problem, using a flowchart, the **predefined process symbol** is used to designate a process or module.

**Predefined process symbol**

The predefined process symbol represents a module in an algorithm – that is, a predefined process that has its own flowchart.

# 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.

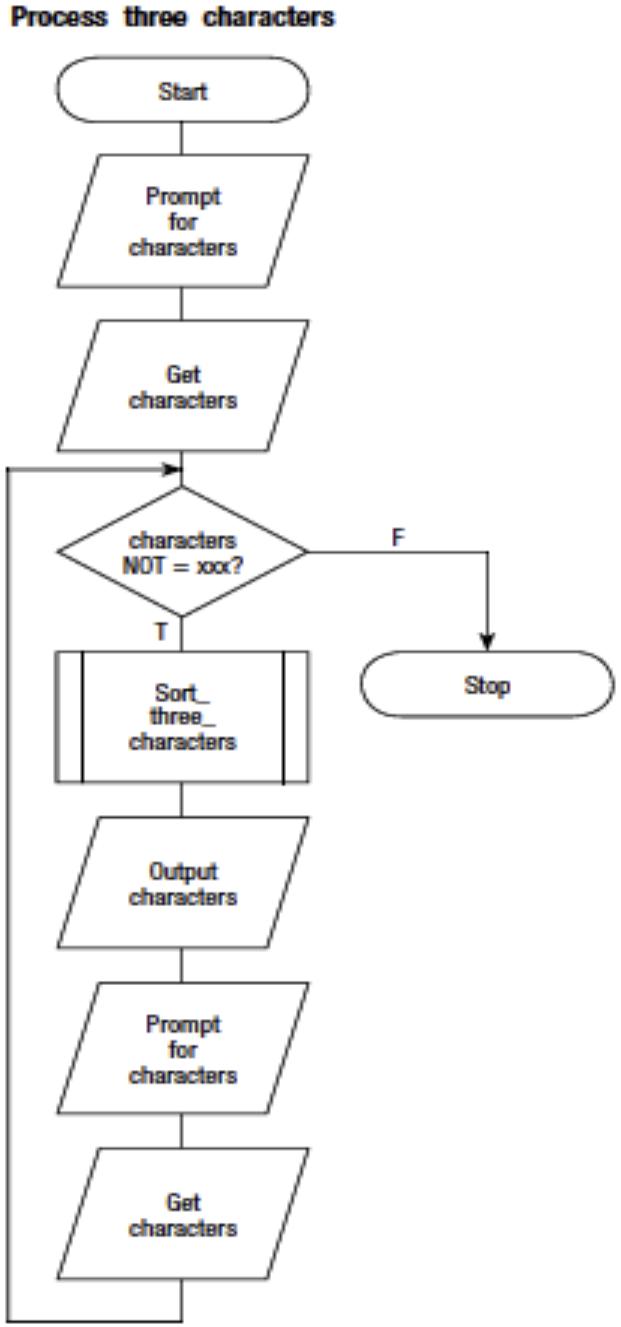
# Example 1: Process three characters

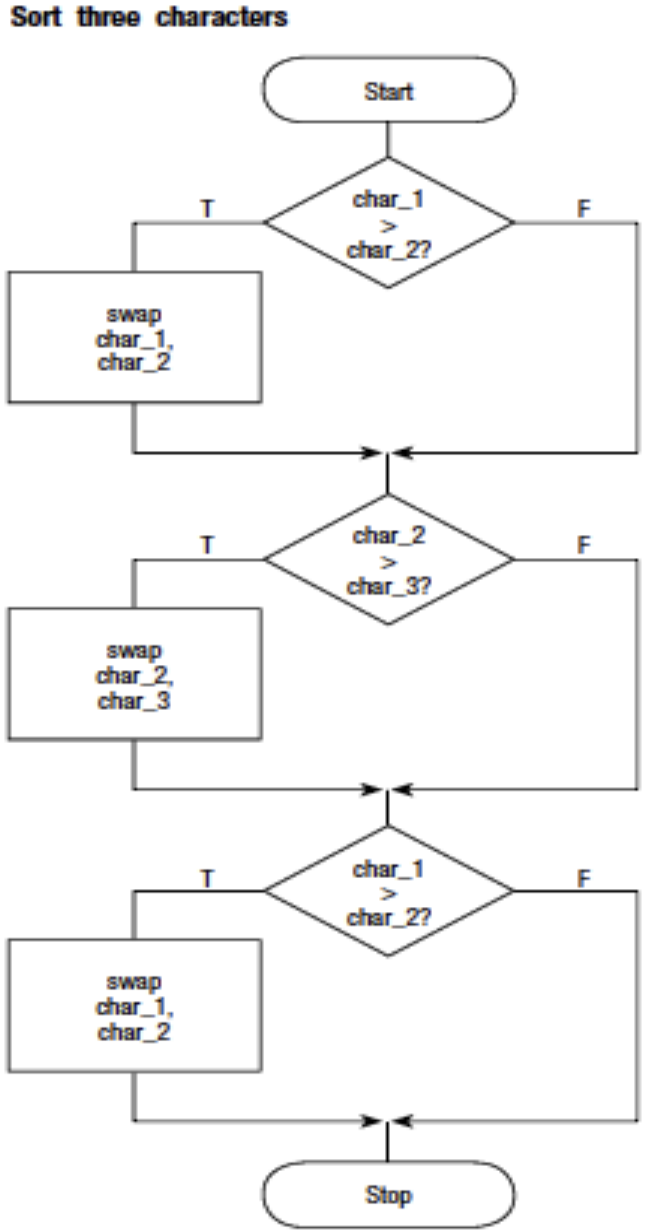
Defining diagram

| Input | Processing | Output |
|---|---|---|
| char_1<br>char_2<br>char_3 | Prompt for characters<br>Accept three characters<br>Sort three characters<br>Output three characters | char_1<br>char_2<br>char_3 |

# Example 1: Process three characters

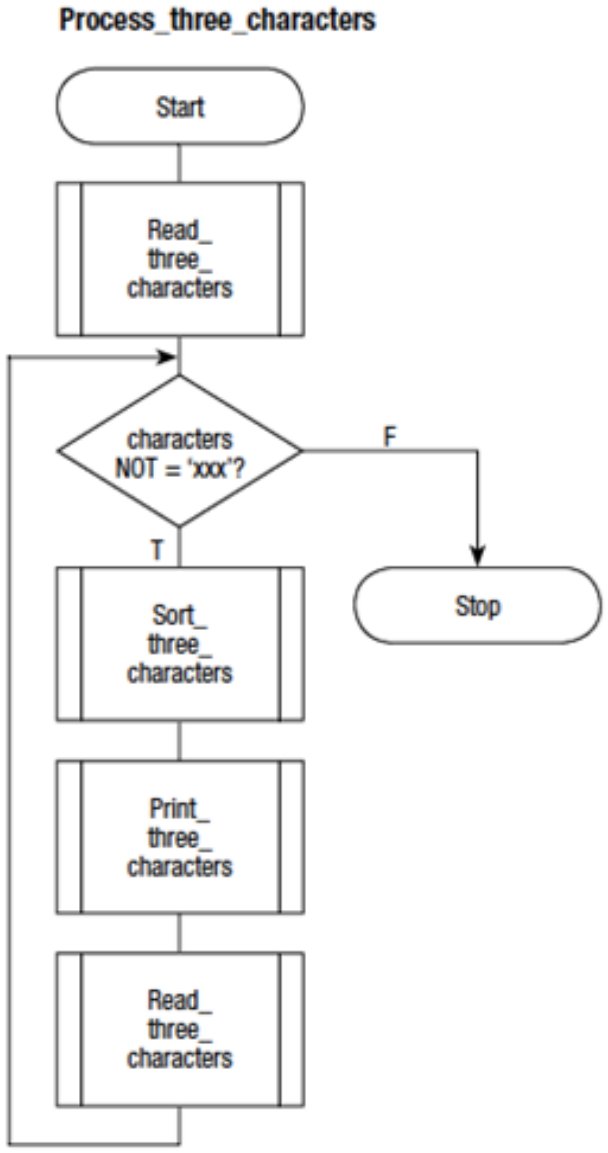
Solution Algorithm using a **module**

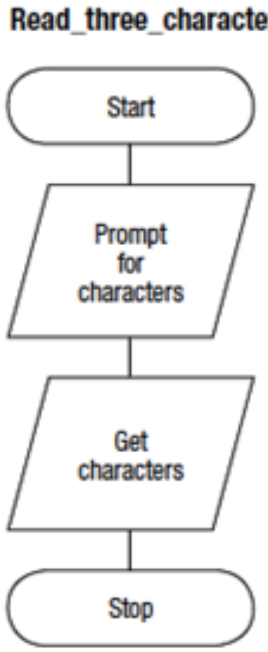


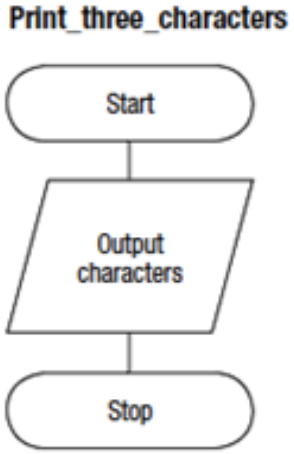



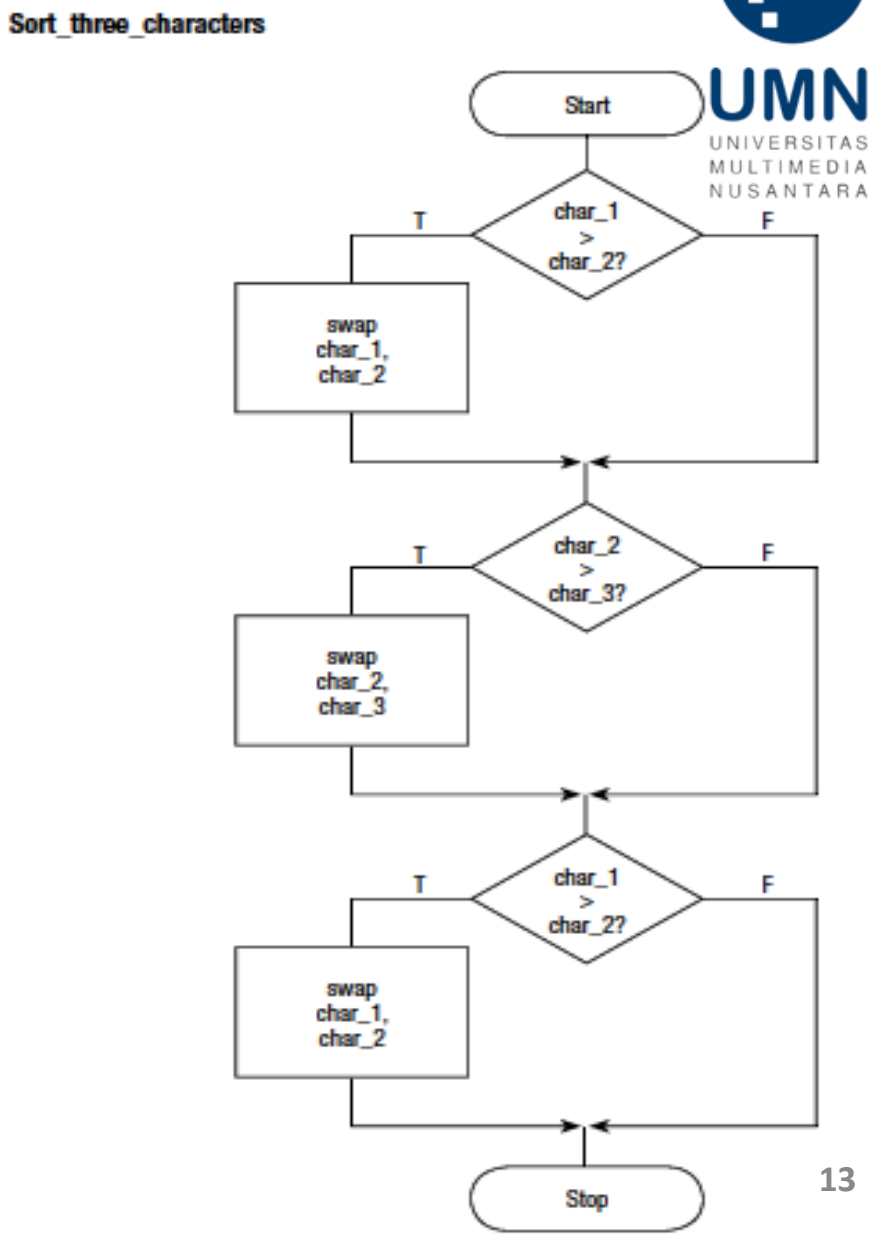
# Example 1: Process three characters

Solution Algorithm using **3 modules**

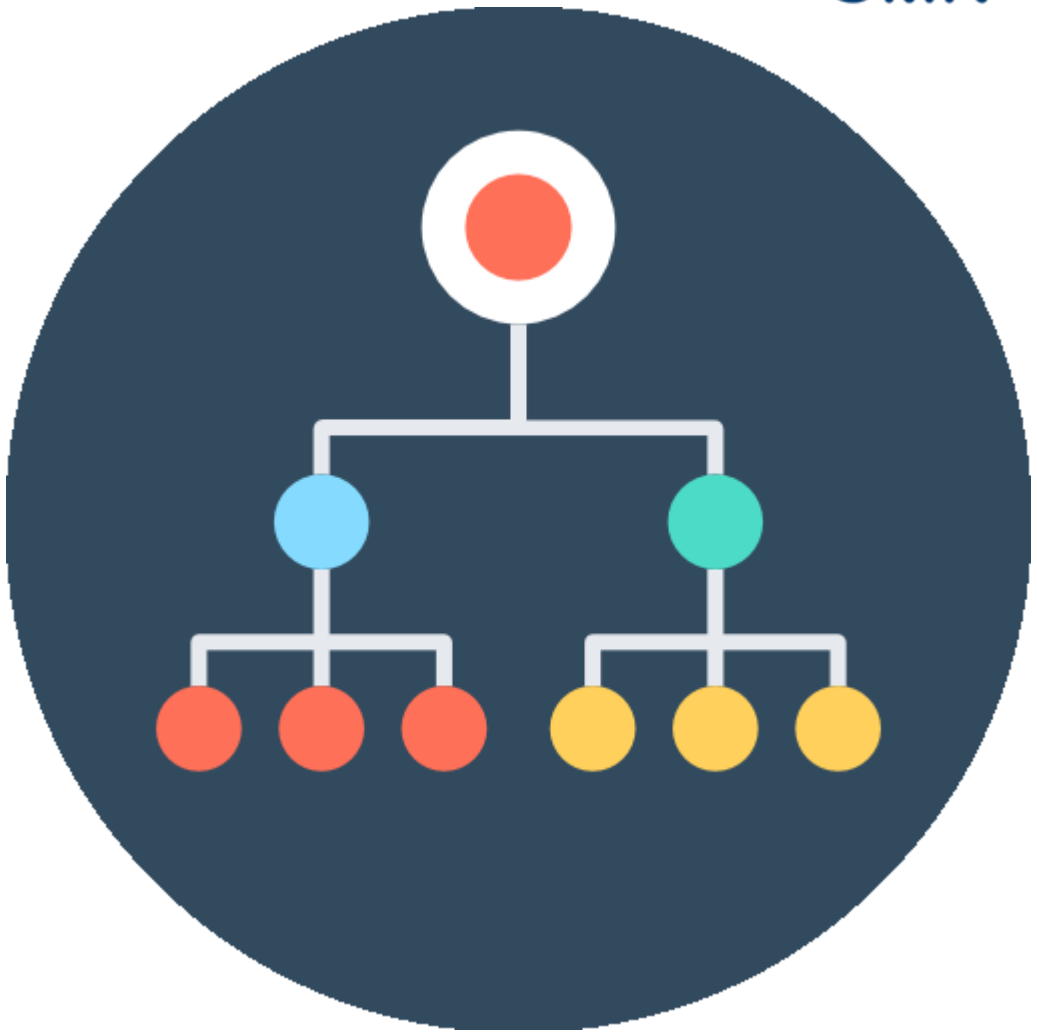


# HIERARCHY CHART
# OR
# STRUCTURE CHART

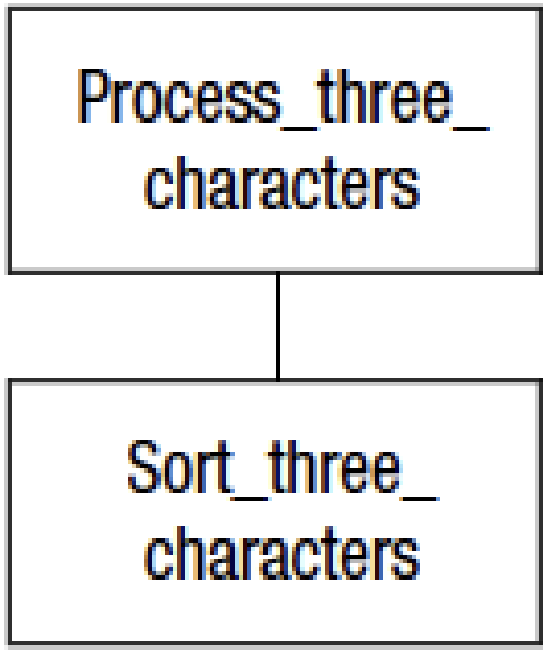


# Hierarchy Chart or Structure Chart

- Once the tasks have been grouped into functions or modules, these modules can be represented **graphically** in a diagram.
- This diagram is known as a hierarchy chart, as it shows not only name of all the modules but also their **hierarchical relationship** to each other.
- The hierarchy chart uses a tree-like diagram of **boxes**
- Each **box** represents a **module** in the program and the **lines** connecting the boxes represent the **relationship of the modules** to others in the program hierarchy.

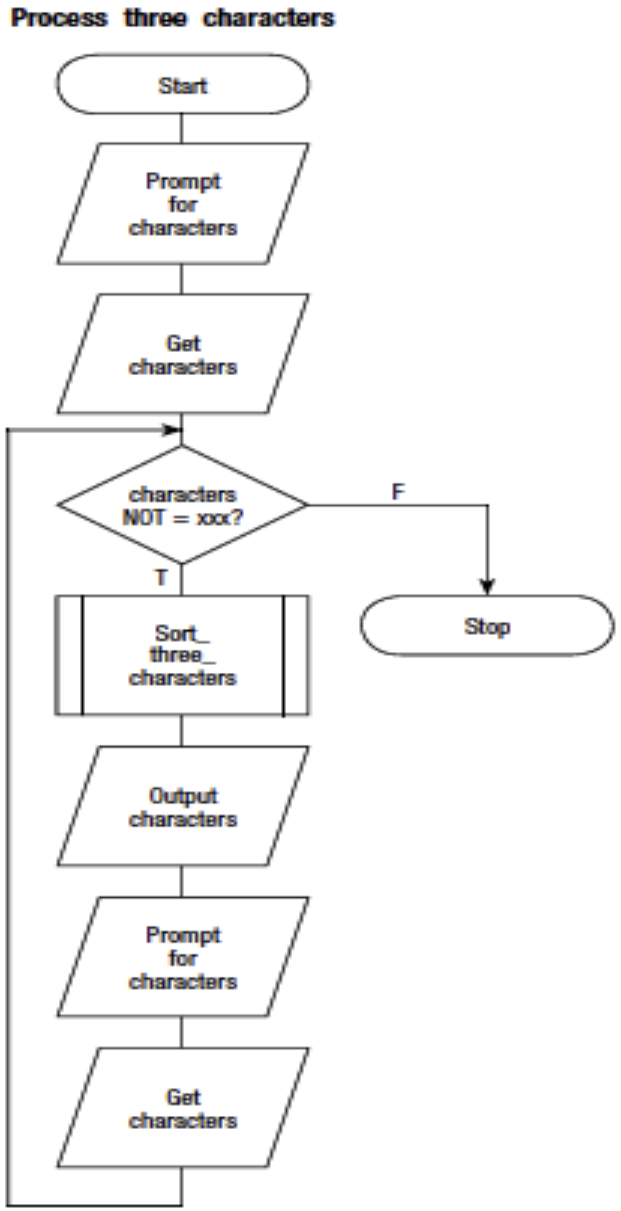
# Example of Hierarchy Chart

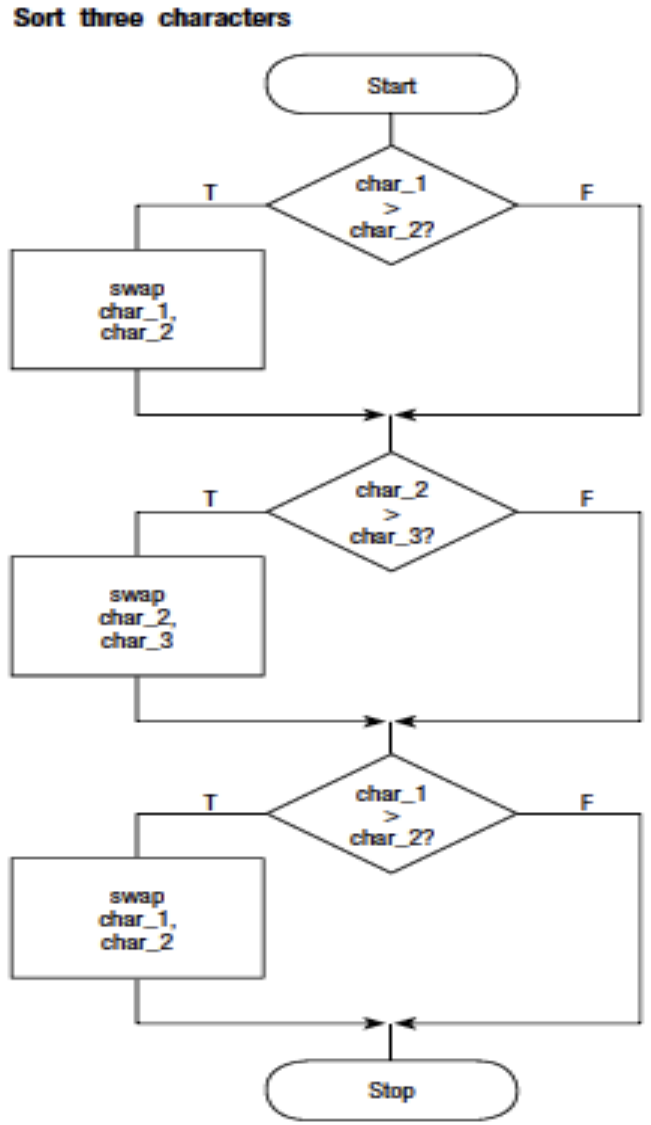
- The hierarchy chart for Example 1: Process three characters is relatively simple.
- It shows a **calling** module (Process_three_characters) and a **called** module (Sort_three_characters)



# Example 1: Process three characters

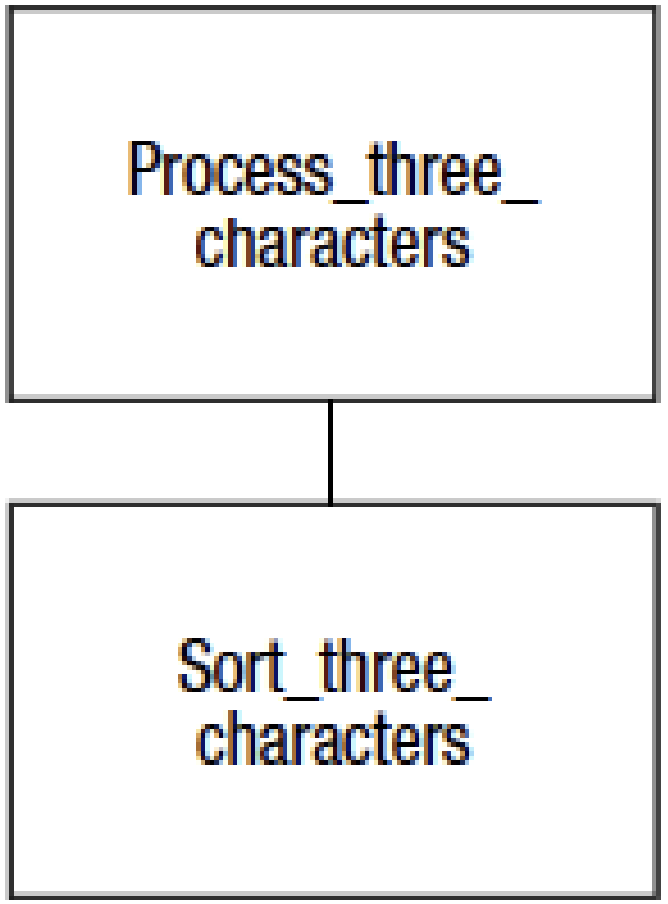
Solution Algorithm using a **module**



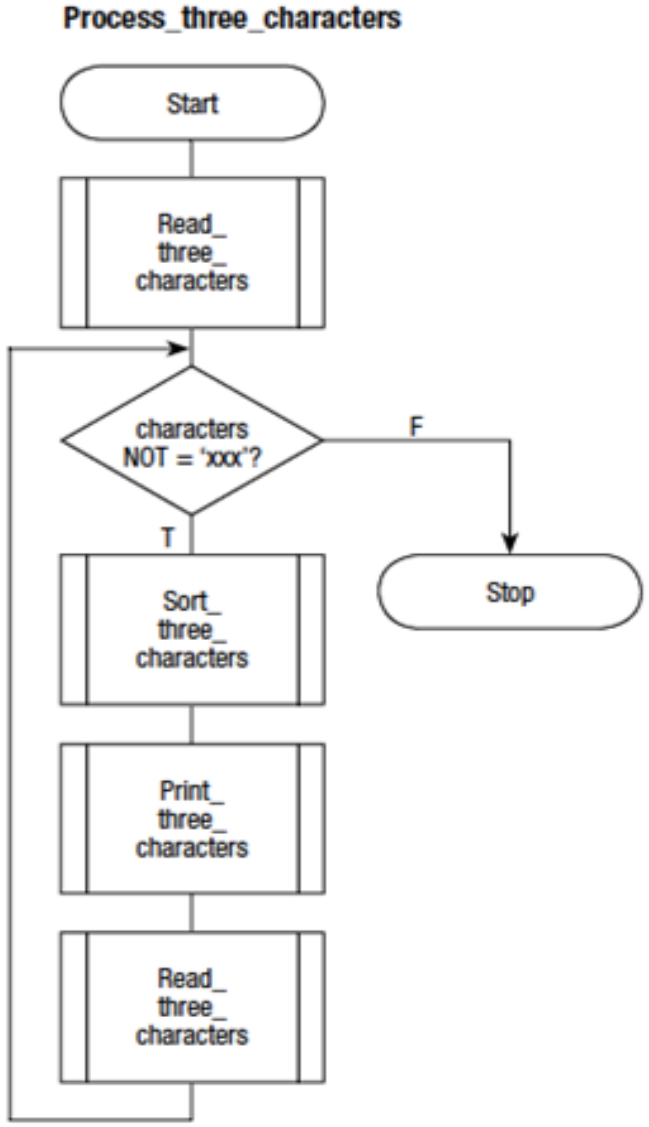
# Example 1: Process three characters

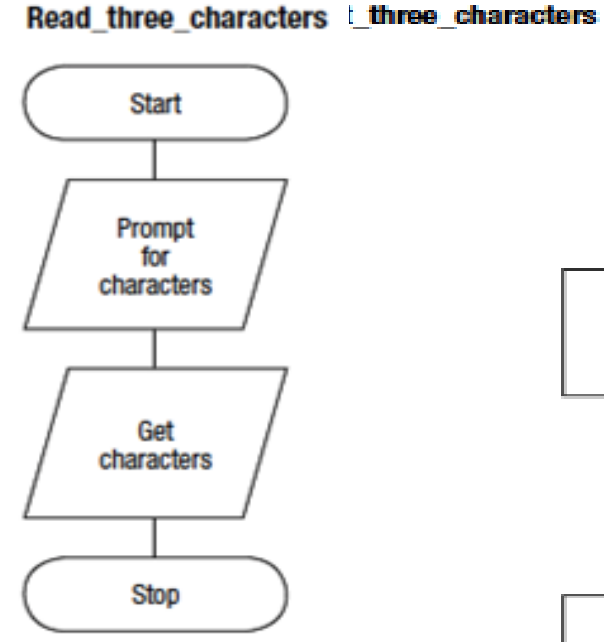
Hierarchy chart

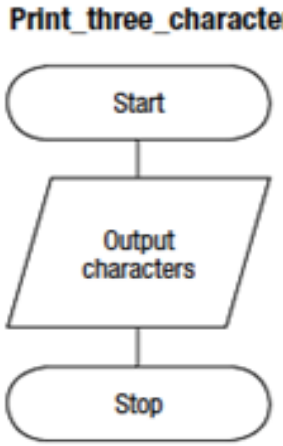


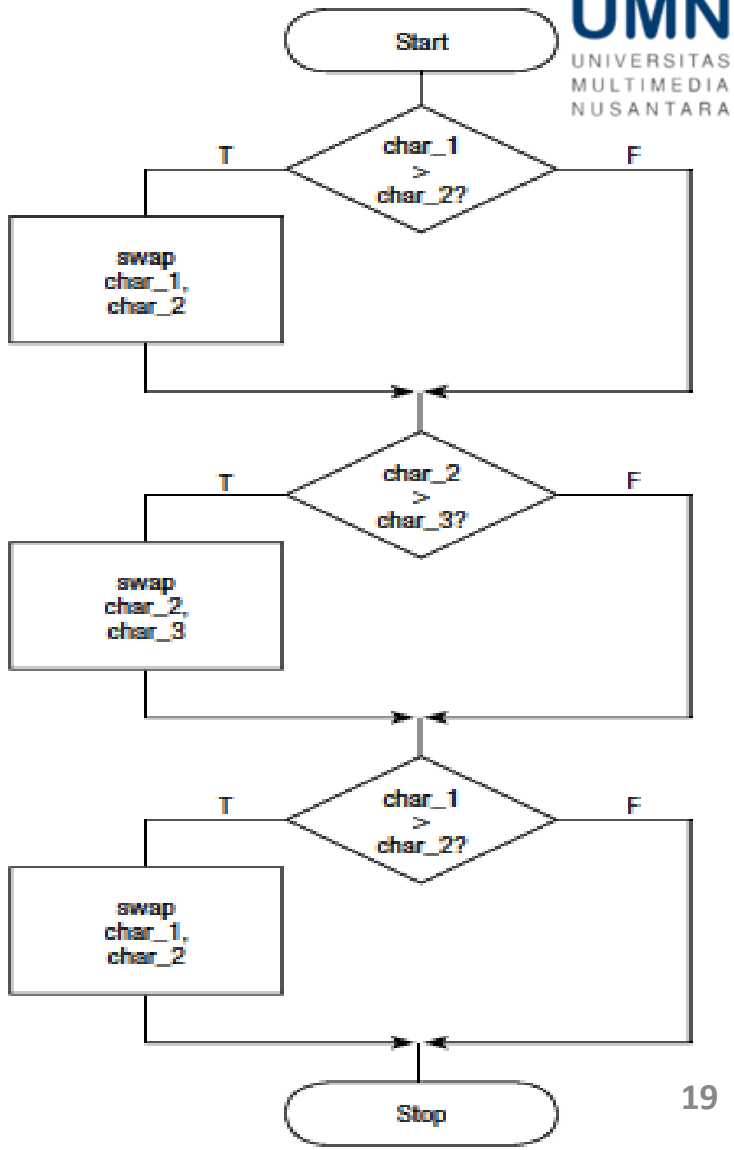
# Example 1: Process three characters

Solution Algorithm using **3 modules**



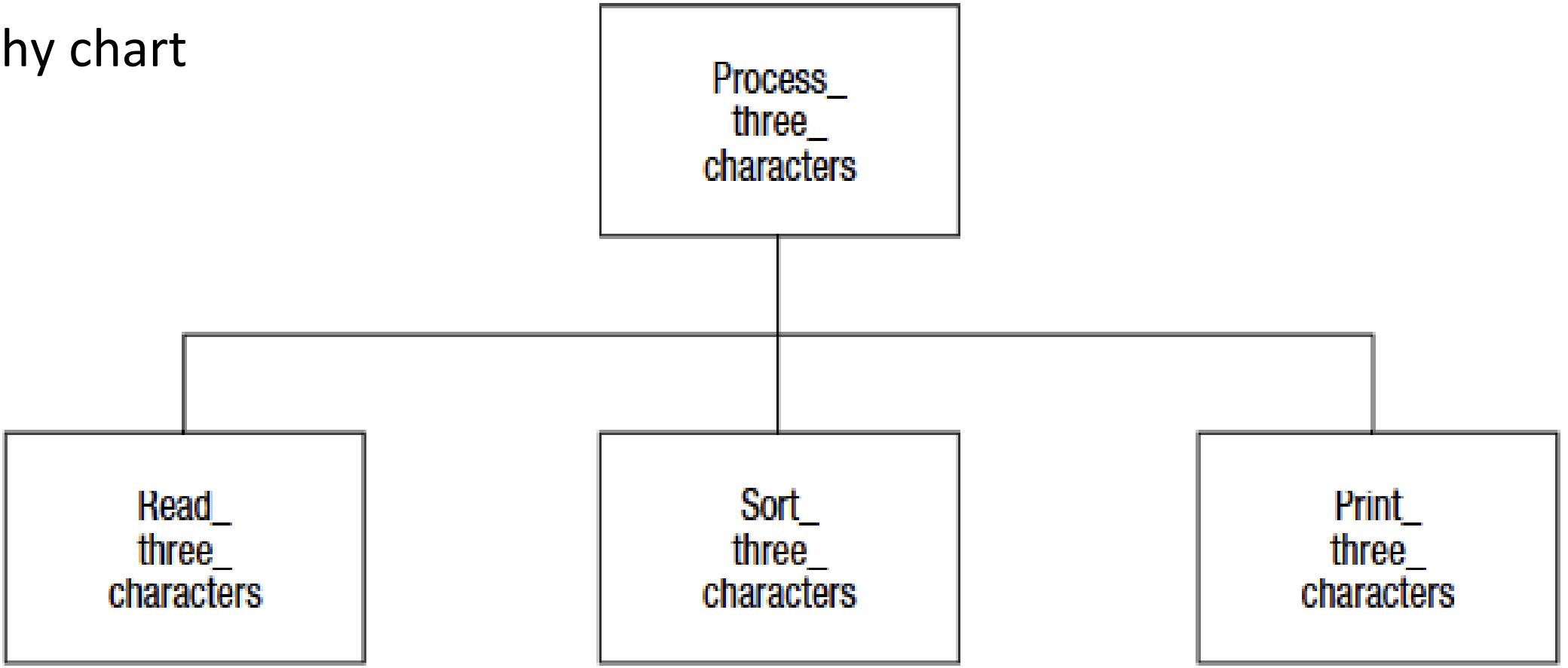



# Example 1: Process three characters (3 modules)

Hierarchy chart



# Steps in modularization

1. Define the problem **by dividing it into its three components**: input, output, and processing.
2. **Group the activities** into subtasks or functions to determine the modules that will make up the program.
3. **Construct a hierarchy chart** to illustrate the modules and their relationship to each other.
4. **Establish the logic** of the mainline of the algorithm in pseudocode / flowchart.
5. Develop the pseudocode/flowchart for **each successive module** in the hierarchy chart.
6. **Desk check** the solution algorithm.

# Example 2: 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.

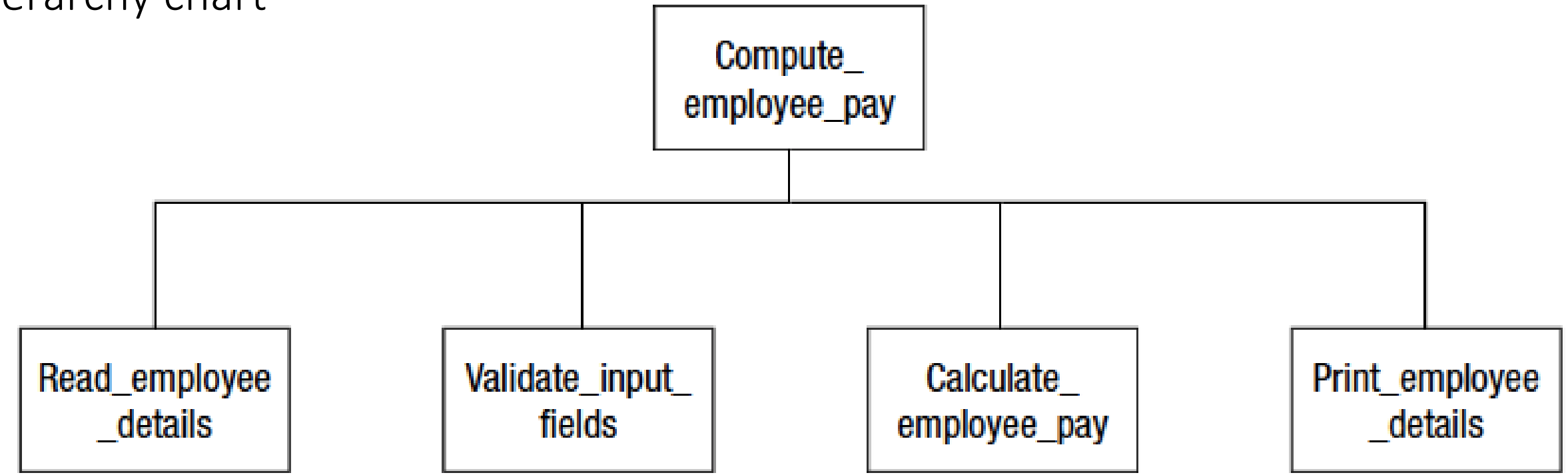
# Example 2: Calculate employee's pay

Defining diagram

| Input | Processing | Output |
|---|---|---|
| emp_no<br>pay_rate<br>hrs_worked | Read employee details<br>Validate input fields<br>Calculate employee pay<br>Print employee details | emp_no<br>pay_rate<br>hrs_worked<br>emp_weekly_pay<br>error_message |

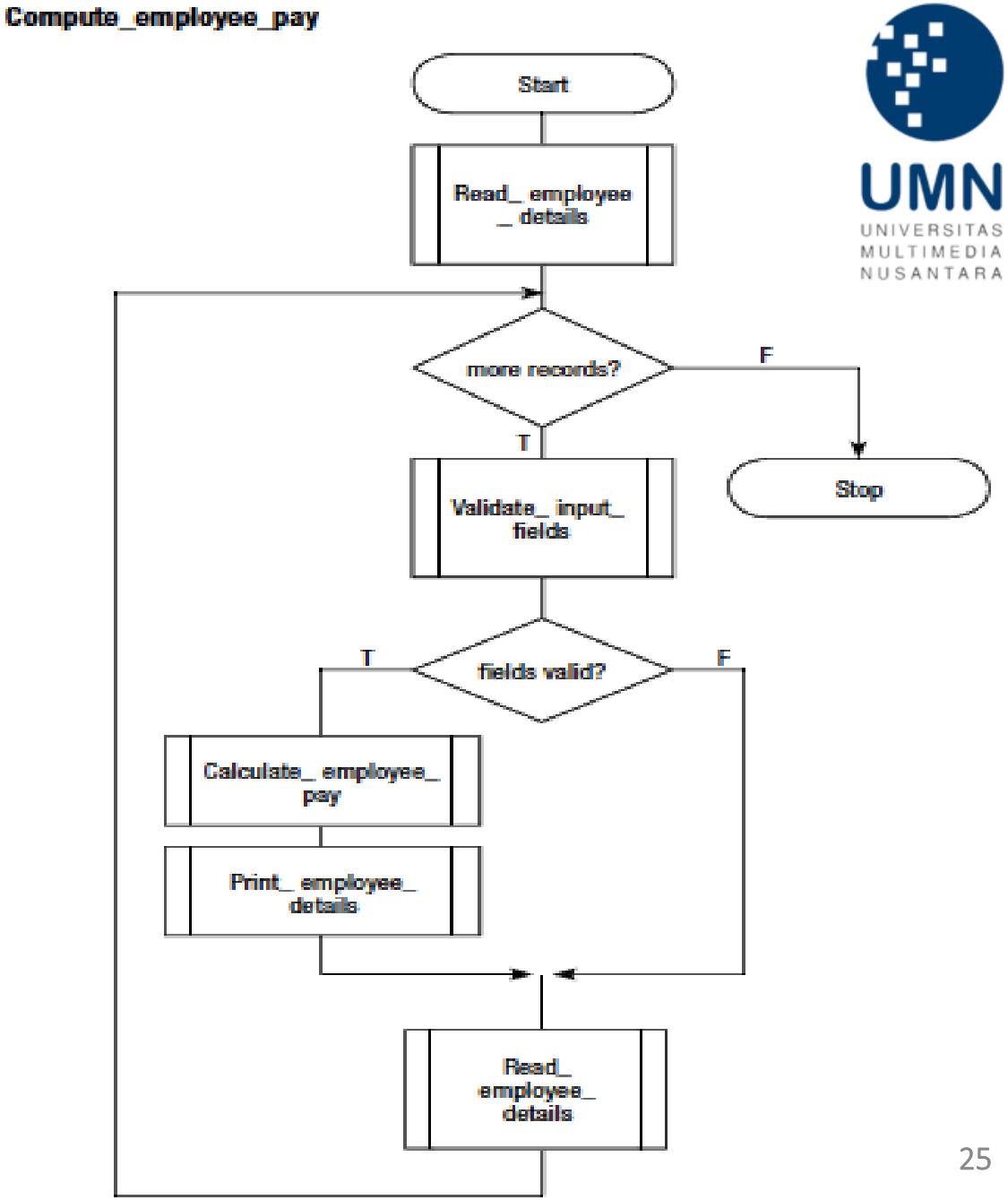
# Example 2: Calculate employee's pay

Hierarchy chart



# Example 2: Calculate employee's pay

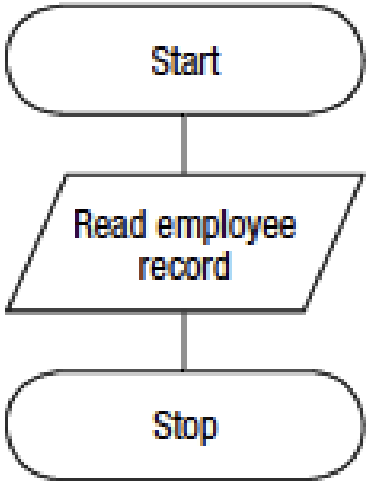
Solution Algorithm



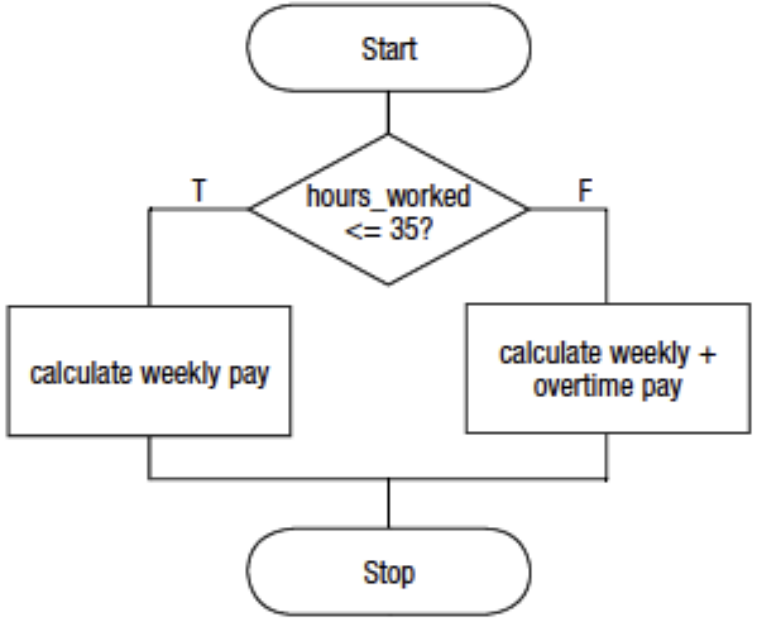
# Example 2: Calculate employee's pay

## Solution Algorithm

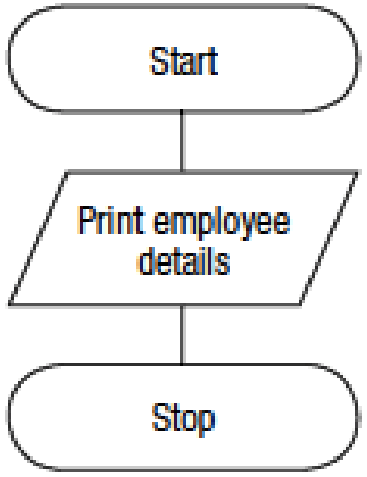
**Read_employee_details**



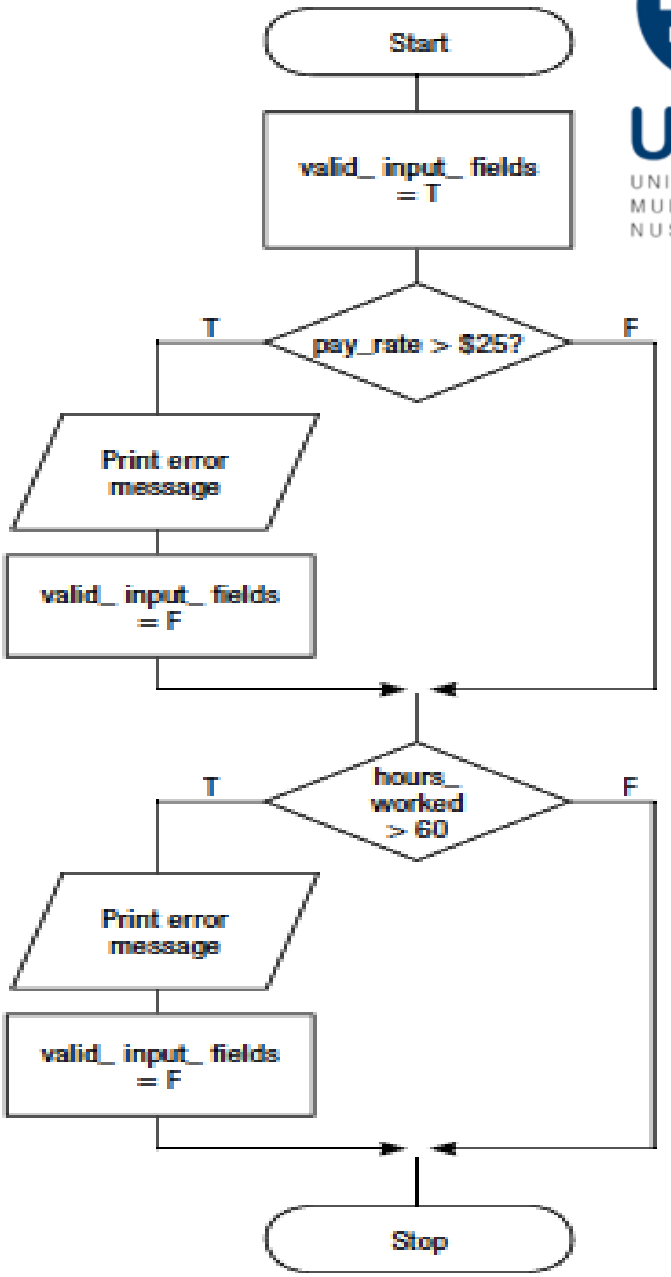

**Calculate_employee_pay**




**Print_employee_details**




**Validate_input_fields**



# Practice 1

Design an algorithm **in modular flowchart** 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 flowchart** 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 flowchart** 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. Definition of modular programming
2. Modular pseudocode
3. Modular Desk checking
4. Exercises

# 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.