UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

IF 130
Programming Fundamentals

02 Pseudocode Struktur Kendali Pemilihan

Disampaikan oleh :
Dr. Maria Irmina Prasetiyowati, S.Kom,. M.T.
Januar Wahjudi, S.Kom., M.Sc.
Putri Sanggabuana Setiawan, S.Kom, M.T.I.
Alethea Suryadibrata, S.Kom., M.Eng.

Capaian Pembelajaran Mingguan Mata Kuliah (Sub-CPMK):

Sub- CPMK 0613: Mahasiswa mampu menyusun pseudocode dengan struktur kendali pemilihan, struktur kendali pengulangan, struktur kendali modularisasi (C3.)

Review

- Algorithm can be defined 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.
- There are 3 variations of **selection control structure**:
  1. Simple IF statement
  2. Null ELSE statement
  3. Nested IF statement
- The **case structure** is another way of expressing a nested IF statement.

How to write pseudocode?

- When designing a solution algorithm, it is necessary to keep in mind the fact that a computer will eventually perform the set of instructions you write.
- If you use **words** or **phrases** in the pseudocode that correspond to some basic computer operations, the translation from the pseudocode algorithm to a specific programming language becomes **quite simple**.
- There are **six basic computer operations**, and common words and keywords used to represent these operations in pseudocode.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

SIX BASIX COMPUTER OPERATIONS

1. A computer can receive information

- When a computer is required to receive information or input from a particular source, whether it be a terminal, a disk or any other device, the verbs **`Read`** and **`Get`** are used in the pseudocode.
- **`Read`** is usually used when the algorithm is to receive input from a record on a file
- **`Get`** is used when the algorithm is to receive input from the keyboard.

```
Read student name
Get system date
Read number_1, number_2
Get tax_code
```

2. A computer can put out information

- When a computer is required to supply information or output to a device, the verbs **`Print`**, **`Write`**, **`Put`**, **`Output`**, or **`Display`** are used in the pseudocode.
- **`Print`** is usually used when the output is to be sent to the printer.
- **`Write`** is used when the output is to be written to a file.
- If the output is to be written to the screen, the words **`Put`**, **`Output`**, or **`Display`** are used in the pseudocode.

```
Print 'Program Completed'
Write customer record to master file
Put out name, address and postcode
Output total_tax
Display 'End of data'
```

2. A computer can put out information (cont.)

- Usually an output **`Prompt`** instruction is required before an input **`Get`** instruction.
- The **`Prompt`** verb causes a message to be sent to the screen, which requires the user to respond, usually by providing input.

```
Prompt for student_mark
Get student_mark
```

3. A computer can perform arithmetic

- Most programs require the computer to perform some sort of mathematical calculation, or to apply a formula
- For these a programmer may use either actual **mathematical symbols** or the **words** for those symbols.
- To be consistent with high-level programming languages, the following symbols can be written in pseudocode:
  - **+** for add
  - **-** for substract
  - ***** for multiply
  - **/** for divide
  - **()** for parentheses

```
add number to total
total = total + number
```

4. A computer can assign a value to a variable or memory location

- There are 3 instances in which you may write pseudocode to assign a value to a variable or memory location:
  1. To give data an initial value in pseudocode, the verbs **`Initialize`** or **`Set`** are used.
  2. To assign a value as a result of some processing, the symbols **'='** or '←' are written.
  3. To keep a variable for later use, the verbs **`Save`** or **`Store`** are used.

```
Initialise total_price to zero
Set student_count to 0
total_price = cost_price + sales_tax
total_price ← cost_price + sales_tax
store customer_num in last_customer_num
```

5. A computer can compare two variables and select one of two alternative actions

- To represent this operation in pseudocode, special keywords are used: **`IF`**, **`THEN`**, and **`ELSE`**.
- The comparison of data is established in the **`IF`** clause, and the choice of alternatives is determined by the **`THEN`** or **`ELSE`** options.
- Only one of these alternatives will be performed.

```
IF student_attendance_status is part_time THEN
    add 1 to part_time_count
ELSE
    add 1 to full_time_count
ENDIF
```

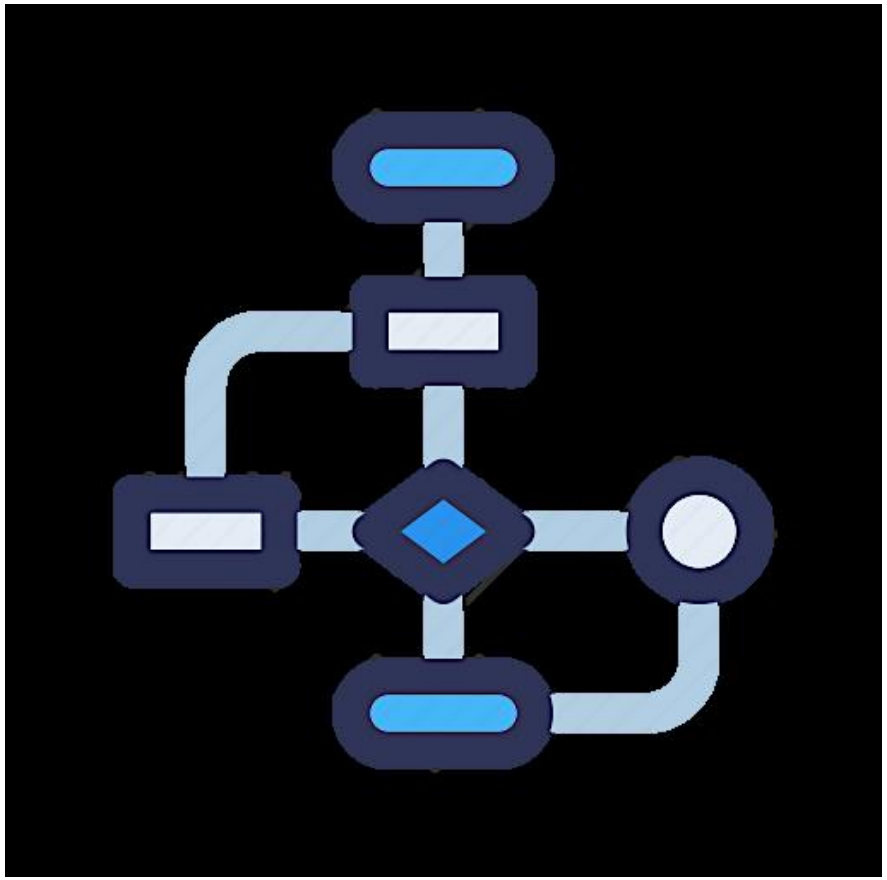
6. A computer can repeat a group of actions

- When there is a sequence of processing steps that need to be repeated, two special keywords, **DOWHILE** and **ENDDO**, are used in pseudocode.
- The condition for the repetition of a group of actions is established in the **DOWHILE** clause, and the actions to be repeated are listed beneath it.

```
DOWHILE student_total < 50
    Read student record
    Print student name, address to report
    add 1 to student_total
ENDDO
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

The Selection Control Structure



The Selection Control Structure

- The condition in the IF statement is based on a comparison of two items, and is usually expressed with one of the following relational operators:
  - **<** less than
  - **>** greater than
  - **=** equal to
  - **<=** less than or equal to
  - **>=** greater than or equal to
  - **<>** not equal to

1. Simple Selection (Simple IF Statement)

- Simple selection occurs when a choice is made between two alternative paths, depending on the result of a condition being true or false.
- The structure is represented in pseudocode using the keywords **IF, THEN, ELSE**, and **ENDIF**.

```
IF account_balance < $300 THEN
    service_charge = $5.00
ELSE
    service_charge = $2.00
ENDIF
```

2. Simple selection with null false branch (null ELSE statement)

- The null ELSE structure is a **variation** of the simple IF structure.
- It is used when a task is performed only when a particular condition is **true**.
- If condition is false, then **no processing** will take place and the IF statement will be bypassed.

```
IF student_attendance = part_time THEN
    add 1 to part_time_count
ENDIF
```

3. Combined selection (combined IF statement)

- A combined **IF** statement is one that contains multiple conditions, each connected with the logical operators **AND** or **OR**.
- If the connector **AND** is used to combine the conditions then both conditions must be true for the combined condition to be true.

```
IF student_attendance = part_time
AND student_gender = female THEN
    add 1 to female_part_time_count
ENDIF
```

```
IF student_attendance = part_time
OR student_gender = female THEN
    add 1 to female_part_time_count
ENDIF
```

3. Combined selection (combined IF statement) (cont.)

- More than two conditions can be linked **together** with the **AND** or **OR** operators.
- However, if both operators are used in the one IF statement, **parentheses** must be used to avoid ambiguity.

```
IF (record_code = '23'
OR update_code = delete)
AND account_balance = zero THEN
    delete customer record
ENDIF
```

3. Combined selection (combined IF statement) (cont.)

- The **NOT** operator can be used for the logical negation of a condition

```
IF NOT (record_code = '23') THEN
    update customer record
ENDIF
```

```
IF NOT (record_code = '23'
    AND update_code = delete) THEN
    update customer record
ENDIF
```

4. Nested Selection (nested IF statement)

- Linear nested IF statement
  - Is used when a field is being tested for various values and a different action is to be taken for each value.
  - The form of nested IF is called **linear**, because each **ELSE** immediately **follows** the **IF** condition to which it corresponds.

```
IF record_code = 'A' THEN
    increment counter_A
ELSE
    IF record_code = 'B' THEN
        increment counter_B
    ELSE
        IF record_code = 'C' THEN
            increment counter_C
        ELSE
            increment error_counter
        ENDIF
    ENDIF
ENDIF
```

4. Nested Selection (nested IF statement)

Non-linear nested IF statement

- Occurs when a number of different conditions need to be satisfied before a particular action can occur.
- It is termed **non-linear** because the **ELSE** statement may be **separated** from the **IF** statement with which it is paired.
- If possible, replace a series of non-linear nested IF statements with a combined IF statement.

```
IF student_attendance = part_time THEN
    IF student_gender = female THEN
        IF student_age > 21 THEN
            add 1 to mature_female_pt_students
        ELSE
            add 1 to young_female_pt_students
        ENDIF
    ELSE
        add 1 to male_pt_students
    ENDIF
ELSE
    add 1 to full_time_students
ENDIF
```

Boolean Variables

- Boolean variables may contain only one of two possible values (true or false).
- When using the IF statement with a Boolean variable, the IF statement can be simplified in pseudocode.

```
IF valid_input_fields = true THEN
    statement
ENDIF
```

Can be written as..

```
IF valid_input_fields THEN
    statement
ENDIF
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Checking The Solution Algorithm

Desk Checking

- After a solution algorithm has been established, it must be tested for correctness.
- Desk checking involves tracing through the logic of the algorithm with some chosen test data.
- This '**playing computer**' not only helps to detect errors early, but also helps you to become familiar with the way program runs.

Steps in desk checking algorithm

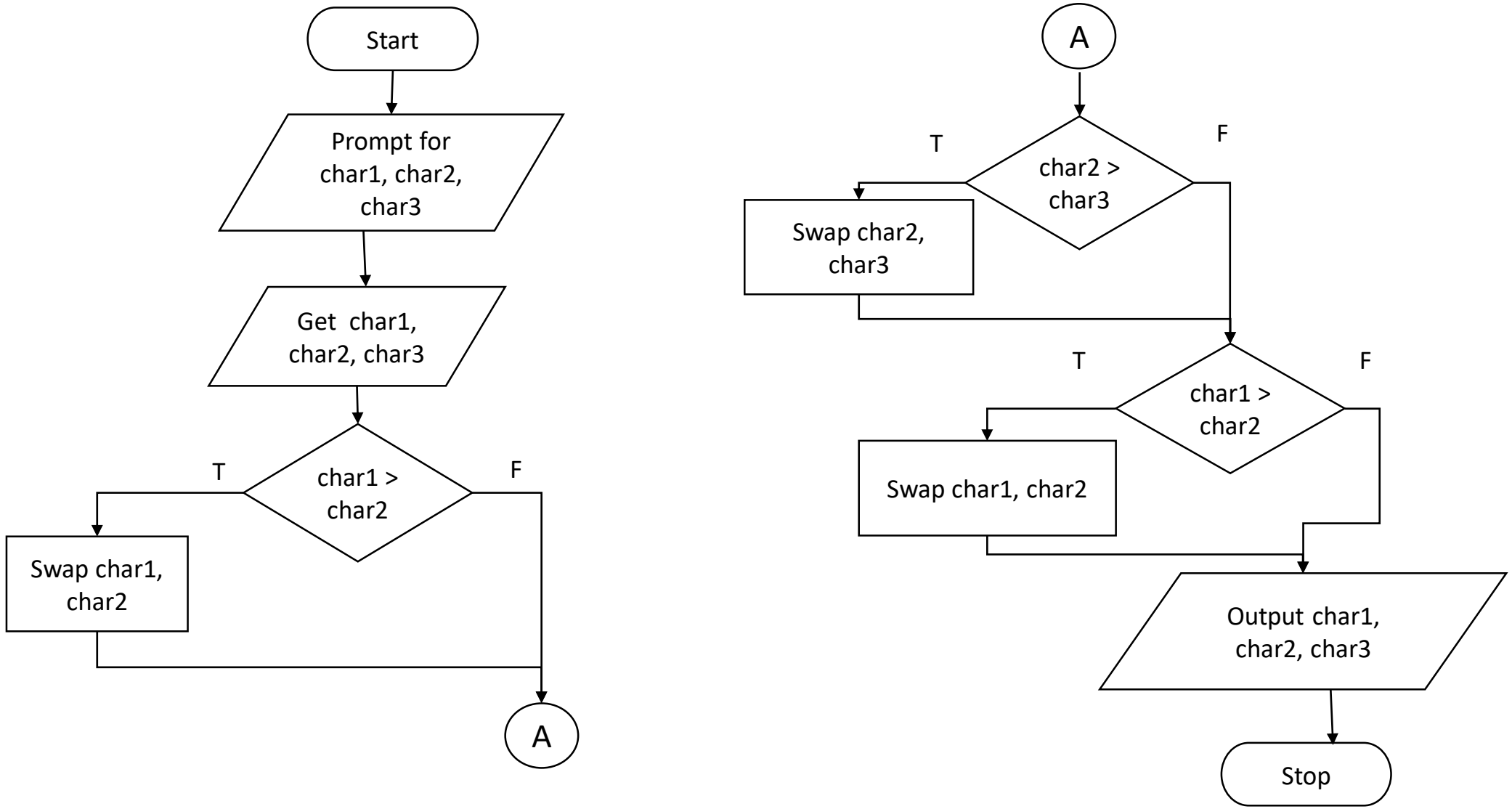
1. Choose simple **input test cases** that are valid. **Two** or **three** test cases are usually sufficient.
2. Establish what the **expected result** should be for each test case.
3. Make a **table** on a piece of paper of the relevant **variable** names within the algorithm.
4. Walk the first test case through the algorithm, **line by line**, keeping a step-by-step record of the contents of each variable in the table as the data passes through the logic.
5. **Repeat** the walk-through process using the other test data cases, until the algorithm has reached its logical end.
6. **Check** that the **expected result** established in Step 2 matches the actual result developed in Step 5.

Example 1: Read 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.

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: Read three characters



Example 1: Read three characters

Solution Algorithm

```
        Read_three_characters
1           Prompt the operator for char_1, char_2, char_3
2           Get char_1, char_2, char_3
3           IF char_1 > char_2 THEN
                temp = char_1
                char_1 = char_2
                char_2 = temp
            ENDIF
4           IF char_2 > char_3 THEN
                temp = char_2
                char_2 = char_3
                char_3 = temp
            ENDIF
5           IF char_1 > char_2 THEN
                temp = char_1
                char_1 = char_2
                char_2 = temp
            ENDIF
6           Output to the screen char_1, char_2, char_3
        END
```

Example 1: Read three characters

1 Input data

	First data set	Second data set
char_1	k	z
char_2	b	s
char_3	g	a

2 Expected results

	First data set	Second data set
char_1	b	a
char_2	g	s
char_3	k	z

Example 1: Read three characters

3 Desk check table

Line numbers have been used to identify each statement within the program. Note that when desk checking the logic each IF statement is treated as a single statement.

Statement number	char_1	char_2	char_3	temp
First pass				
1, 2	k	b	g	
3	b	k		k
4		g	k	k
5				
6	output	output	output	
Second pass				
1, 2	z	s	a	
3	s	z		z
4		a	z	z
5	a	s		s
6	output	output	output	

Example 2: Process customer record

A program is required to read a customer's name, a purchase amount and a tax code. The tax code has been validated and will be one of the following:

0 – tax exempt (0 %)

1 – state sales tax only (3 %)

2 – federal and state sales tax (5 %)

3 – special sales tax (7 %)

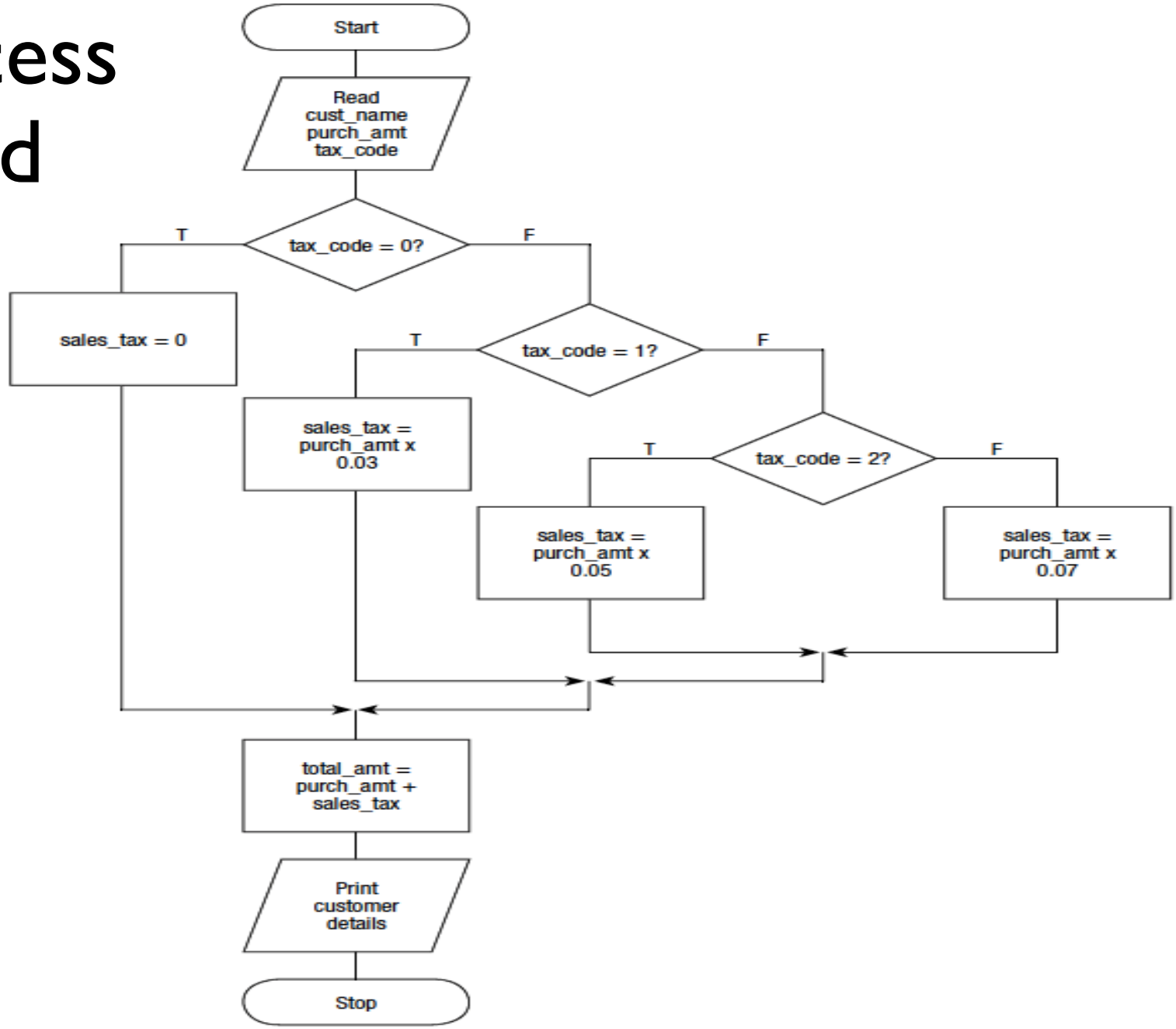
The program must then compute the sales tax and the total amount due, and print the customer's name, purchase amount, sales tax and total amount due.

Example 2: Process customer record

Defining diagram

Input	Processing	Output
cust_name	Read customer details	cust_name
purch_amt	Compute sales tax	purch_amt
tax_code	Compute total amount	sales_tax
	Print customer details	total_amt

Example 2: Process customer record



Example 2: Process customer record

Solution algorithm

```
        Process_customer_record
1           Read cust_name, purch_amt, tax_code
2           IF tax_code = 0 THEN
                sales_tax = 0
            ELSE
                IF tax_code = 1 THEN
                    sales_tax = purch_amt * 0.03
                ELSE
                    IF tax_code = 2 THEN
                        sales_tax = purch_amt * 0.05
                    ELSE
                        sales_tax = purch_amt * 0.07
                    ENDIF
                ENDIF
            ENDIF
3           total_amt = purch_amt + sales_tax
4           Print cust_name, purch_amt, sales_tax, total_amt
        END
```

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Example 2: Process customer record

1 Input data

	First data set	Second data set
purch_amt	$10.00	$20.00
tax_code	0	2

2 Expected results

	First data set	Second data set
sales_tax	0	$1.00
total_amt	$10.00	$21.00

3 Desk check table

Statement number	purch_amt	tax_code	sales_tax	total_amt
First pass				
1	$10.00	0		
2			0	
3				$10.00
4	print		print	print
Second pass				
1	$20.00	2		
2			$1.00	
3				$21.00
4	print		print	print

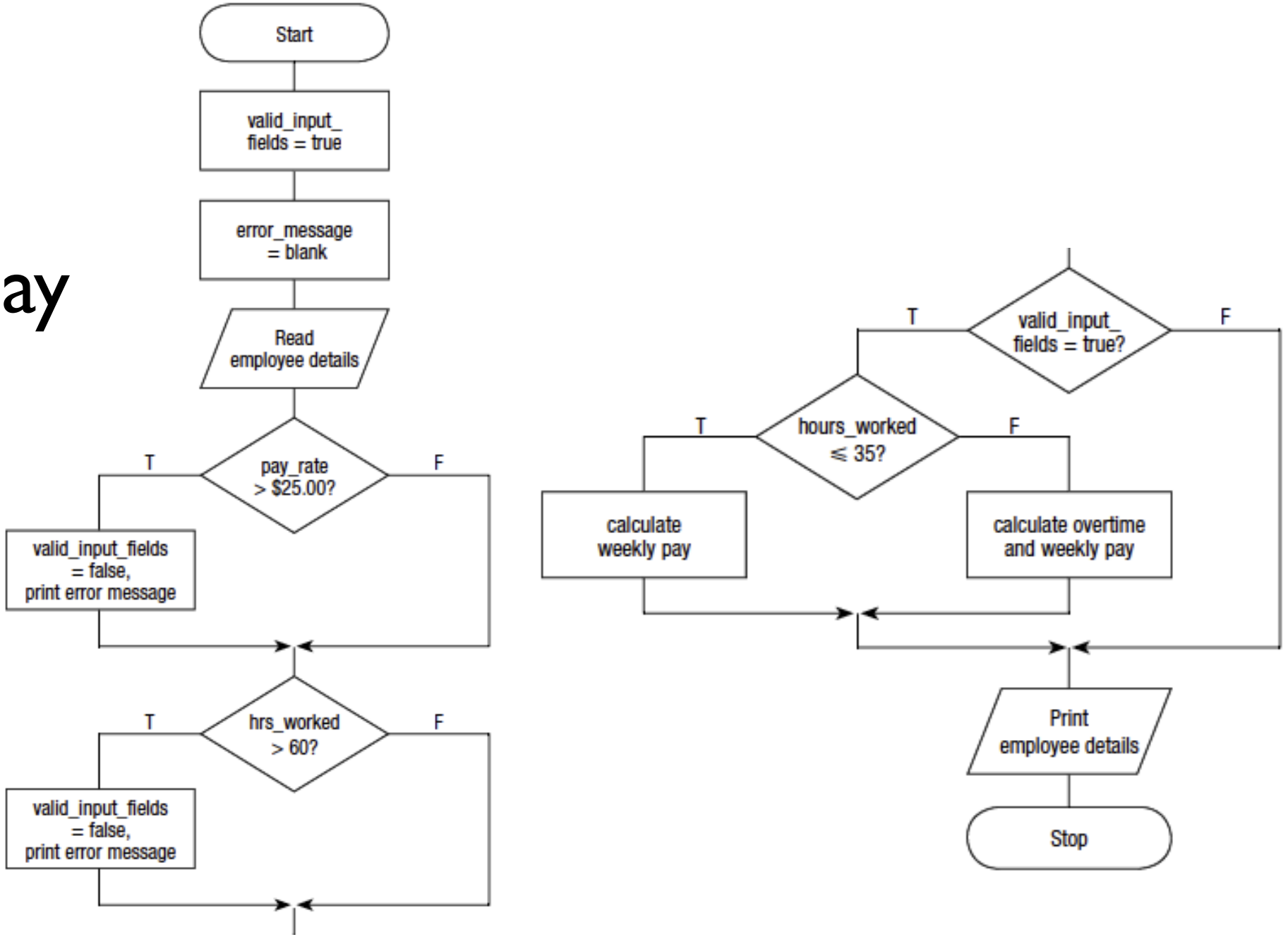
Example 3: Calculate employee's pay

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 3: Calculate employee's pay

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 3: Calculate employee's pay



Example 3: Calculate employee's pay

Solution algorithm

```
        Compute_employee_pay
1           Set valid_input_fields to true
2           Set error_message to blank
3           Read emp_no, pay_rate, hrs_worked
4           IF pay_rate > $25 THEN
                error_message = 'Pay rate exceeds $25.00'
                Print emp_no, pay_rate, hrs_worked, error_message
                valid_input_fields = false
            ENDIF
5           IF hrs_worked > 60 THEN
                error_message = 'Hours worked exceeds 60'
                Print emp_no, pay_rate, hrs_worked, error_message
                valid_input_fields = false
            ENDIF
6           IF valid_input_fields THEN
                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
                Print emp_no, pay_rate, hrs_worked, emp_weekly_pay
            ENDIF
        END
```

Example 3: Calculate employee's pay

1 Input data

	First data set	Second data set
pay_rate	$10.00	$40.00
hrs_worked	40	35

2 Expected results

	First data set	Second data set
pay_rate	$10.00	$40.00
hrs_worked	40	35
emp_weekly_pay	$425.00	–
error_message	blank	Pay rate exceeds $25.00

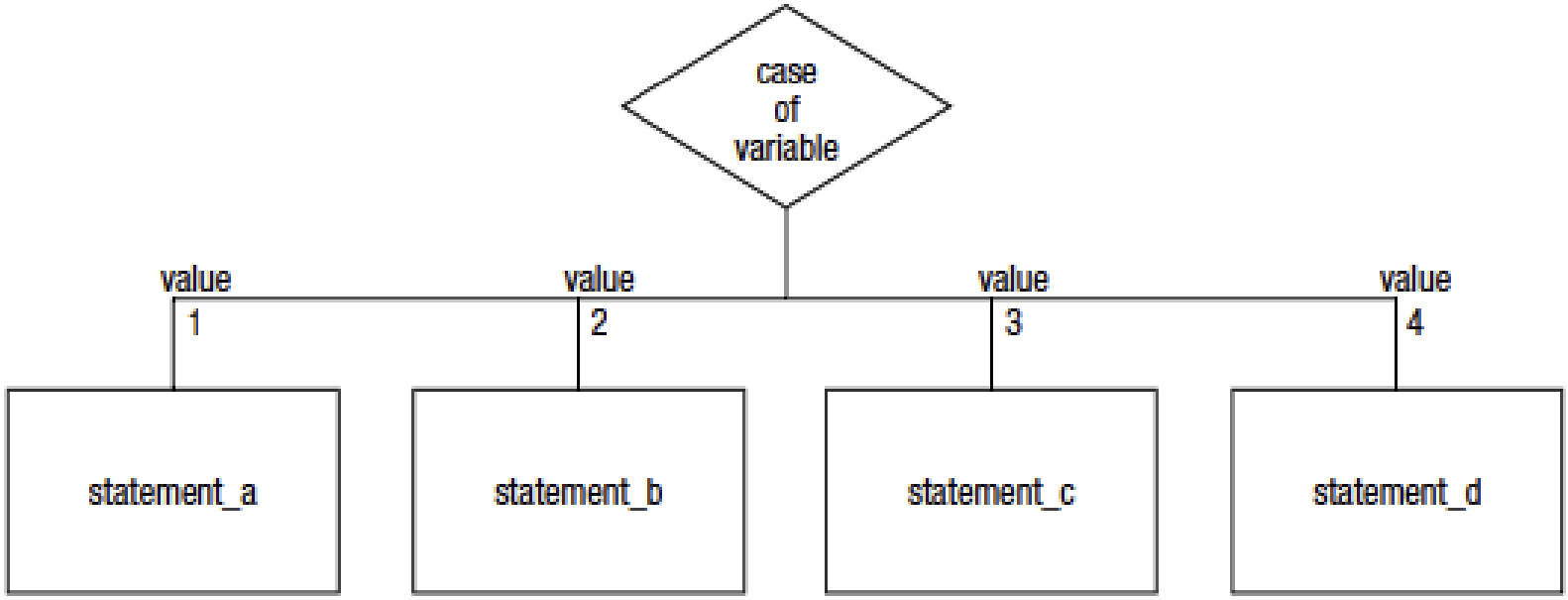
Example 3: Calculate employee's pay

3 Desk check table

Statement number	pay_rate	hrs_ worked	overtime_ hrs	overtime_ pay	emp_ weekly_ pay	valid_ input_ fields	error_ message	Print
First pass								
1						true		
2							blank	
3	$10.00	40						
4								
5								
6			5	75.00	425.00			Print fields
Second pass								
1						true		
2							blank	
3	$40.00	35						
4						false	Pay rate exceeds $25.00	Print message
5								
6								

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

The Case Structure



The Case Structure

- The case control structure in pseudocode is another way of expressing a linear nested IF statement.
- It is used in pseudocode for 2 reasons:
  - It can be translated into many high-level languages
  - it makes the pseudocode easier to write and understand

```
CASE OF single variable
    value_1 : statement block_1
    value_2 : statement block_2
        .
        .
        .
    value_n : statement block_n
    value_other : statement block_other
ENDCASE
```

Lets us look at the example used earlier in this chapter..

```
IF record_code = 'A' THEN
    increment counter_A
ELSE
    IF record_code = 'B' THEN
        increment counter_B
    ELSE
        IF record_code = 'C' THEN
            increment counter_C
        ELSE
            increment error_counter
        ENDIF
    ENDIF
ENDIF
```

We can now rewrite the above linear nested IF statement with a case statement..

```
CASE OF record_code
    'A'     : increment counter_A
    'B'     : increment counter_B
    'C'     : increment counter_C
    other   : increment error_counter
ENDCASE
```

Example : Process customer record

A program is required to read a customer's name, a purchase amount and a tax code. The tax code has been validated and will be one of the following:

0 – tax exempt (0 %)

1 – state sales tax only (3 %)

2 – federal and state sales tax (5 %)

3 – special sales tax (7 %)

The program must then compute the sales tax and the total amount due, and print the customer's name, purchase amount, sales tax and total amount due.

Example: Process customer record

Defining diagram

Input	Processing	Output
cust_name	Read customer details	cust_name
purch_amt	Compute sales tax	purch_amt
tax_code	Compute total amount	sales_tax
	Print customer details	total_amt

Example: Process customer record (In Case Structure Pseudocode)

```
        Process_customer_record
1           Read cust_name, purch_amt, tax_code
2           CASE OF tax_code
                0 : sales_tax = 0
                1 : sales_tax = purch_amt * 0.03
                2 : sales_tax = purch_amt * 0.05
                3 : sales_tax = purch_amt * 0.07
            ENDCASE
3           total_amt = purch_amt + sales_tax
4           Print cust_name, purch_amt, sales_tax, total_amt
        END
```

Example : Process customer record

1 Input data

	First data set	Second data set
purch_amt	$10.00	$20.00
tax_code	0	2

2 Expected results

	First data set	Second data set
sales_tax	0	$1.00
total_amt	$10.00	$21.00

3 Desk check table

Statement number	purch_amt	tax_code	sales_tax	total_amt
First pass				
1	$10.00	0		
2			0	
3				$10.00
4	print		print	print
Second pass				
1	$20.00	2		
2			$1.00	
3				$21.00
4	print		print	print

Practice 1

Design an algorithm **in 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 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 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%

NEXT WEEK'S OUTLINE

1. Definition of repetition control structure
2. Kind of repetition control structure
3. Flowchart of repetition control structure
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, 2022, 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**.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

INFORMATIKA
UMN

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.