Pseudocode and Flowchart

Programming Fundamentals Teaching Team 2023

Objectives

After studying this material, students should be able to:

1. Understand and explain pseudo-code
2. Describe and explain flowcharts including the its symbols
3. Understand and use pseudocode and flowcharts to model the algorithm

PSEUDOCODE

Pseudocode

- Pseudocode is a high-level description of a computer programming algorithm that uses **notation** that **corresponds** to **programming code**. The **notation of pseudocode** is **not related** to a **specific programming language**
- Pseudocode is used to design and to model algorithm (the steps to solve a logical problem). It could explain the algorithm clearly and easily understood by humans, not by machines.
- Written in a simple English or Indonesian
- **Syntax** or **notation** is used to describe **commands** (for example: “if”, “repeat”, “until”)

Pseudocode

The following syntaxes are the most basic notation in pseudocode:

1. Input the data (**Input**)
2. Output the data(**Output**)
3. Calculate or operate the data(**Compute**)
4. Save the data (**Store**)
5. Compare the data (**Compare**)
6. Repeat process (**Loop**)

1. Input

- The syntaxes that could be used to input data in pseudocode are **“Read”, “Get”,** or **“Baca”**
- Example:

  Read number

  Get taxCode

  Baca name

2. Output

- The following syntaxes are notation to output the data **“Print”, “Write”, “Put”, “Output”, “Display”** or **“Cetak”**
- Example:

  Print “Politeknik Negeri Malang”

  Cetak “Arithmetic Operation”

  Output Total

3. Computation

- To operate the data, the symbol of operator could be utilized. For example, to perform an arithmetic operation we can use:

+	add
-	substract
*	multiply
/	divide
()	bracket

- Statement **“Compute”, “Calculate”** or **“Hitung”** could be used as well.
- Example:

```
Compute  X = (1+2) * 3
Total = Total + 10
```

4. Store the Data

- There are 3 options to store the data:
  - Giving **initial value → "Initialize"** or **"Set"**
  - Giving a value as **the result of operation → "="**
  - Giving a **specific value** for a **variable → "Save"** or **"Store"**
- Example:

  Set age **to** 17

  total = price * amount

  Store 50 **to** Discount

5. Comparation

- One of the most fundamentals process to compare the data, and based on the comparation result (true/false) the program will select the further process
- The syntaxes are **“IF”, “THEN”** and **“ELSE” (**or **“JIKA”, “MAKA”, “SELAIN ITU”)**
- Example

```
IF condition=1 THEN
  Discount = 0.1 * price
ELSE
  Discount = 0.2 * price
ENDIF
```

6. Looping

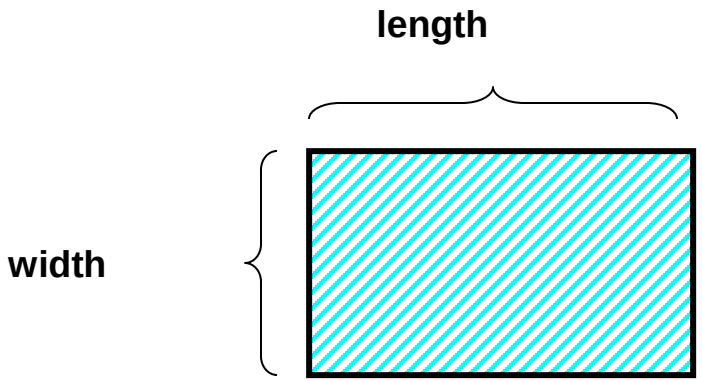
- If there is any process that needs to be run many times in a specific pattern, until reaching a specific conditions, we can us syntax → **“DOWHILE”** and **“ENDDO”**.
- Example

```
number = 0
DOWHILE number < 10
  output number
   number = number +1
ENDDO
```

Example

Create an algorithm using pseudocode to calculate the area of a rectangle, with the following formula:

area = length * width



Pseudocode Solution

Algorithm: rectangle area calculation

{input length and width and then calculate the area of rectangle}

Declaration :

length, width, area: int

Description :

1. print "Program to Calculate Area of a Rectangle"
2. print "Input length = "
3. read length;
4. print "Input width= "
5. read width
6. area = length * width
7. print "The area is = "
8. print area

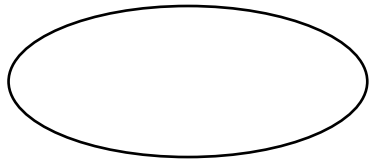
FLOWCHART

Flowchart

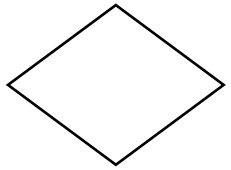
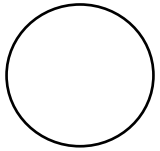
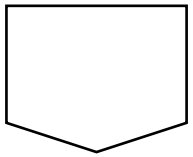
- If Pseudocode models the algorithm using a text-based description, a Flowchart models the algorithm using a graphical notation instead.
- Flowchart uses **graphical symbols** to **illustrate the steps of the processe**s and its **connectivity** to **solve the problem**
- One of algorithm modelling.
- There are 2 types of flowchart:

1. System Flowchart → steps of process in the system, that shows input, output, storage in correlation with data processing
2. Program Flowchart → steps of instructions that is illustrated by using a graphical notation, to model the algorithm of a program

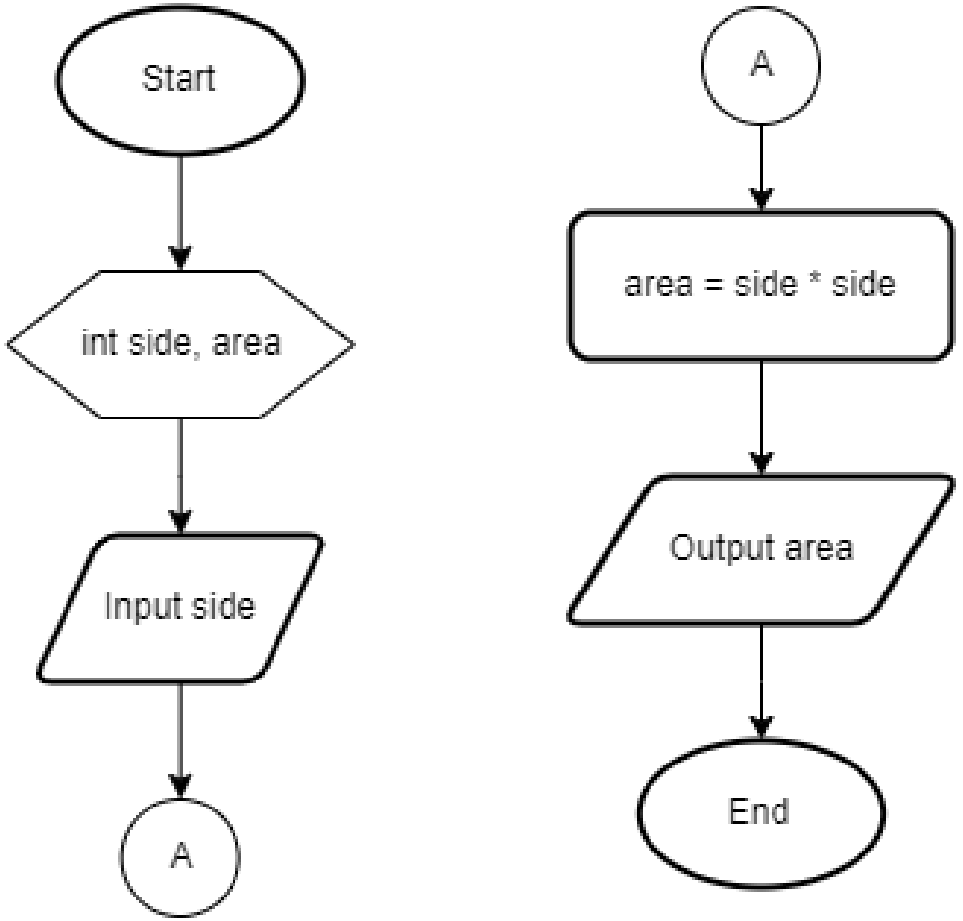
Flowchart Symbols

Symbol	Name	Description
	Terminator	Symbol for the beginning (start) and finish (end) of an algorithm
	Preparation	Symbol for preparing storage that will be used
	Input-output	Symbol for input in output
	Flow Line	Symbol that illustrates the flow of process

Flowchart Symbols

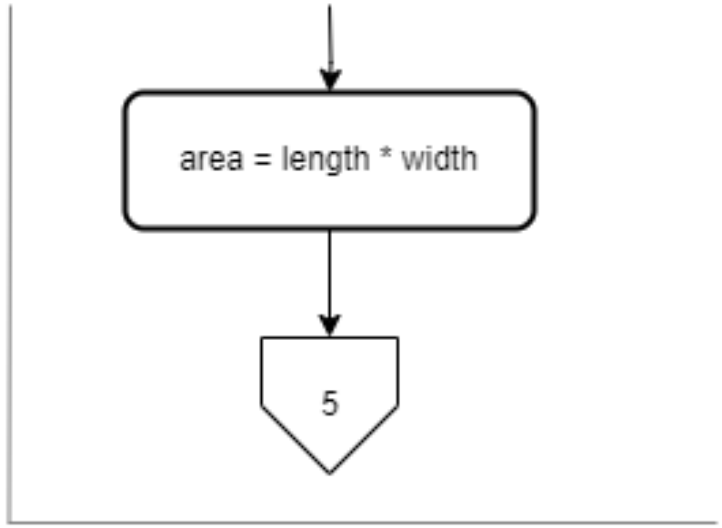
Symbol	Name	Description
	Process	Symbol to illustrate the process
	Branching/Condition Selection	To check or select the condition (it results true or false)
	On Page Connector	Connecting the flowchart in the same page
	Off Page Connector	Connecting the flowchart in the different page
	Predefined Process	Symbol that illustrates the sub-process or sub-program, function, procedure

Example for On Page Connector

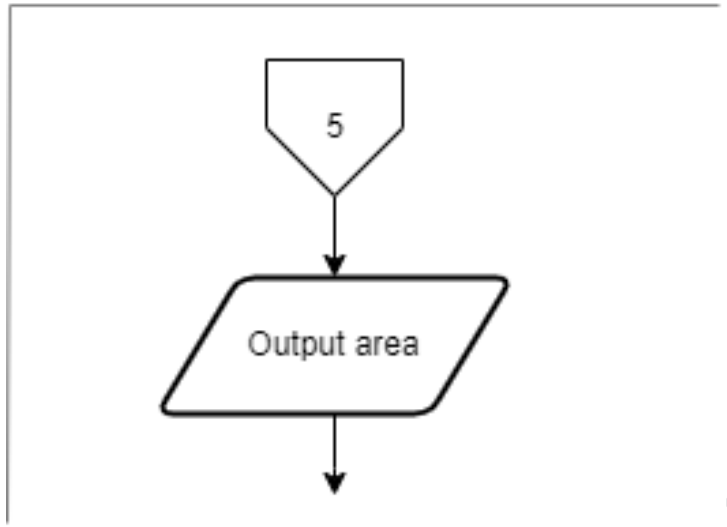


Example for Off Page Connector

end of page 1



beginning of page 2



Example of Week 3

Mr. Adi has a rectangular garden. Mr. Adi wants to make a wooden fence to surround the garden.

Before creating a program to help Mr. Adi calculate the perimeter of his garden, help Mr. Adi to identify variables and data types along with the algorithm!

Example

1. Algorithm

Input: length, width

Output: perimeter

Process:

1. input length, width
2. Perimeter = 2 x (length + width)
3. Output perimeter

2. Variable Identification

Variable	Data type
length	int
width	int
perimeter	int

PSEUDOCODE

Algorithm : Perimeter Calculation

{Input length and width of the rectangle and calculate the perimeter}!

Declaration :

length, width, perimeter : int

Description :

1. **read** length
2. **read** width
3. perimeter = length * width
4. **print** perimeter

Example

PSEUDOCODE

Algorithm : Perimeter Calculation

{Input length and width of the rectangle and calculate the perimeter}!

Declaration :

length, width, perimeter : int

Line 6,7,8

Description :

1. **read** length
2. **read** width
3. perimeter = 2 * (length + width)
4. **print** perimeter

Line 10,11

Line 12

Line 13

```
1   import java.util.Scanner;
2
3   public class Example1 {
4       public static void main(String[] args) {
5           Scanner input = new Scanner(System.in);
6           int length;
7           int width;
8           int perimeter;
9
10          length = input.nextInt();
11          width = input.nextInt();
12          perimeter = 2 * (length + width);
13          System.out.println(perimeter);
14      }
15  }
```

Example

```
1   import java.util.Scanner;
2
3   public class Example1 {
4       public static void main(String[] args) {
5           Scanner input = new Scanner(System.in);
6           int length;
7           int width;
8           int perimeter;
9
10          length = input.nextInt();
11          width = input.nextInt();
12          perimeter = 2 * (length + width);
13          System.out.println(perimeter);
14      }
15  }
```

Line 6,7,8

Line 10,11

Line 12

Line 13