Tingkat III
Rekayasa Perangkat Lunak Kripto
2019

P2 Perancangan Program Dengan PseudoCode

Metode Perancangan Program

MATERI YANG DIPELAJARI

1. Memahami konsep perancangan program dengan pseudocode.
2. Mampu mengidentifikasi permasalahan dalam bentuk pseudocode
3. Mampu mengkonversi pseudocode ke dalam bahasa pemrograman

INTRO PSEUDOCODE

DEFINITION OF PSEUDOCODE

- Informal way of programming description that does not require any strict programming language syntax
- Sometimes used as a detailed step in the process of developing a program
- One of the popular representation of Algorithm besides Flowchart

An algorithm is *"An effective procedure for solving a problem in a finite number of steps.*

PSEUDOCODE CONVENTION

- Statement are written in simple English
- Each instruction is written on a separate line
- Keywords and indentation are used to signify particular control structures.
- Each set of instructions is written from top to bottom, with only one entry and one exit.
- Groups of statements may be formed into modules, and that group given a name.

SIX BASIC COMPUTER OPERATIONS

1. A computer can receive information
2. A computer can put out information
3. A computer can perform arithmetic
4. A computer can assign a value to a variable or memory location
5. A computer can compare two variables and select one of two alternate actions
6. A computer can repeat a group of actions

A computer can receive information

Verb used:

- **Read** → used when the algorithm is to receive the input from a record on a file
- **Get** → 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
```

A computer can put out information

Verb used:

- **Print** → used when the output is to be sent to the printer
- **Write** → used when the output is to be written to a file
- **Put, Output, Display** → used when the output is to be written to the screen
- **Prompt** → required before an input instruction Get, causes the message to be sent to the screen which requires the user responds, usually by providing input

```
Print `Program Completed´
Write customer record to master file
Put out name, address and postcode
Output total_tax
Display ´End of data´
Prompt for student_mark
Get student_mark
```

A computer can perform arithmetic

Verb used:

- **Compute**
- **Calculate**

Symbols used: +, -, *, /, ()

```
Add number to total
      Total = total + number

Divide total_marks by student_count
      Sales_tax = cost_price * 0.10
      Compute C = (F – 32) * 5/9
```

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

Three cases :

1. To give data an initial value in pseudocode, the verbs **Initialise** 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.

```
Initialize 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
```

A computer can compare two variables and select one of two alternate actions

Keyword used: **IF, THEN, ELSE**

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

A computer can repeat a group of actions

Keyword used : **DOWHILE – ENDDO, REPEAT-UNTIL, FOR-ENDFOR, WHILE-END WHILE**

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

REPEAT
 statement
UNTIL (condition)
```

```
WHILE (condition)
 statement
END WHILE

FOR (var = startValue;
testValue ; var = stop value)
ENDFOR
```

EXERCISE

- **Buat Pseudocode untuk penilaian predikat nilai dimana Predikat A (85-100), B (75-84), C (65-74), D (55-64), E (45-54), F (<45)**

ALGORITHM SOLUTION DESIGN

HOW TO WRITE SOLUTION

1. A name should be given to the algorithm, which is describe the function of algorithm
2. An END statement used to indicate the algorithm is complete
3. All processing steps between the algorithm name and END statement should be indented for readability.
4. Each processing step in the defining diagram relates directly to one or more statements in the algorithm.

EXAMPLE

- A program is required to read three numbers, add them together and print their total.

```
Add_three_numbers
   Read number1, number2, number3
   Total = number1 + number2 + number3
   Print total
END
```

HOW TO WRITE FUNCTION

1. Function headers should appear as:

```
FunctionName (parameters: field1, field2, etc. )
```

2. Returns in functions should appear as:

```
Return (field1)
```

3. Function footer should appear as:

```
ENDFUNCTION
```

4. Calls to Functions should appear as:

```
CALL FunctionName (arguments: field1, field2, etc.)
```

EXAMPLE

- A program is required to read three numbers, add them together and print their total.

```
Add_three_numbers (parameters number1, number2, number3)
   Initialize Total to zero
   Total = number1 + number2 + number3
   Return (Total)
ENDFUNCTION

CALL Add_three_numbers(n1,n2,n3)
```

EXAMPLE

```
Sum_three_numbers

Iniziate total to zero
Add_three_numbers (parameters number1, number2, number3)
   Initialize Total to zero
   Total = number1 + number2 + number3
   Return (Total)
ENDFUNCTION

Total = Add_three_numbers(n1,n2,n3)

Print Total

END
```

EXERCISE

- Buat Pseudocode untuk penilaian predikat nilai dimana Predikat A (85-100), B (75-84), C (65-74), D (55-64), E (45-54), F (<45)
- **Ubah Pseudocode di atas dalam format solution**
- **Buat pseudocode dengan menggunakan function pada kode utama**

DESK CHECKING

DEFINITION

Tracing through the logic of the algorithm with some chosen data..

Step in desk Checking an algorithm

1. Choose valid simple input test case (2-3 enough)
2. Establish what the expected result should be.
3. Make a table of relevant variable names
4. Checking the test case line by line, step by step
5. Repeat process 4 for other test case
6. Check if expected result 2 matches with actual result 5

EXAMPLE

- A program is required to read three numbers, add them together and print their total.

```
Add_three_numbers
   Read number1, number2, number3
   Total = number1 + number2 + number3
   Print total
END
```

DESK CHECKING

1. Choose two sets input test data.

Set 1: 10,20, 30 and Set 2: 40, 41, 42

	Data Set 1	Data Set 2
Number 1	10	40
Number 2	20	41
Number 3	30	42

DESK CHECKING

2. Establish the expected result for each test case

	Data Set 1	Data Set 2
Total	60	123

DESK CHECKING

3. Set up a table of relevant variable names, and pass each test data set statement by statement.

Statement number	number1	number2	number3	total
First Pass				
1	10	20	30	
2				60
3				Print
Second Pass				
1	40	41	42	
2				123
3				Print

DESK CHECKING

4. Checking the test case line by line, step by step

Statement number	number1	number2	number3	total
First Pass				
1	10	20	30	
2				60
3				Print
Second Pass				
1	40	41	42	
2				123
3				Print

5. Check the expected results (60 and 123) match the actual results.

EXERCISE

- Buat Pseudocode untuk penilaian predikat nilai dimana Predikat A (85-100), B (75-84), C (65-74), D (55-64), E (45-54), F (<45)
- Ubah Pseudocode di atas dalam format solution
- Buat pseudocode dengan menggunakan function pada kode utama
- **Buat Desk Checking**

Levels of Testing

1

Unit Testing

By Developer

2

Integration Testing

By Developer & Tester

3

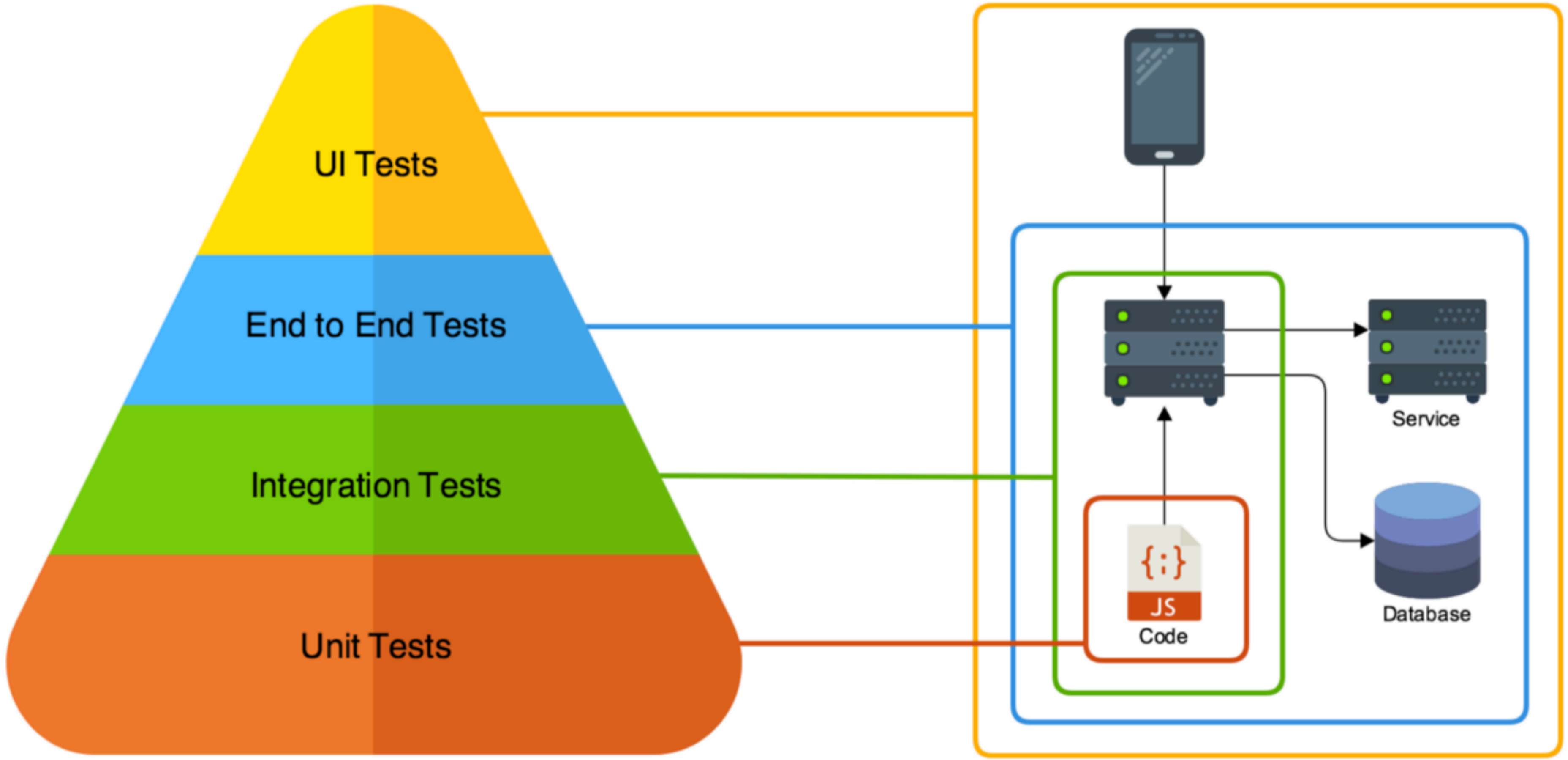
System Testing

By Tester

4

User Acceptance Testing

By End User / Customer


UI Tests
End to End Tests
Integration Tests
Unit Tests
Service
Database
JS
Code

SELESAI

Next : SDLC