Function

(part 1)

Programming Fundamentals Teaching Team - 2023

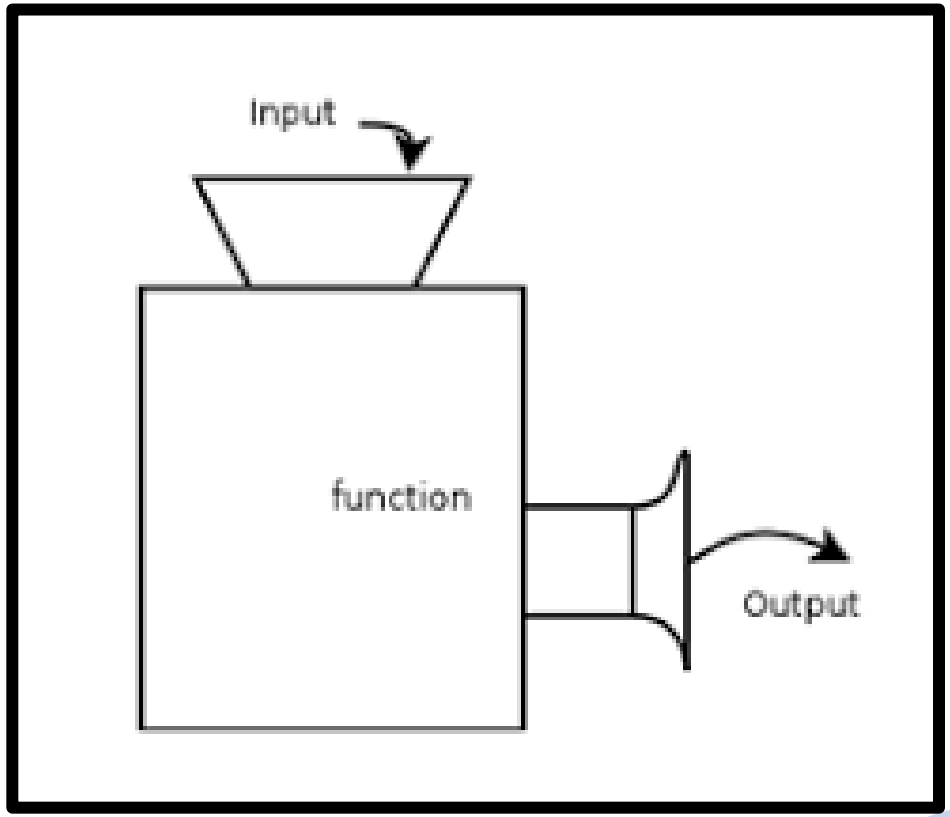
Tujuan

After finishing this lesson, students must be able to master the following competencies:

- the concept of Function
- the way to declare Functions
- the way to call Functions

Definition

- A **function** is several instructions which are grouped into one, standalone, that functions to complete a specific job/process.
- By using functions, the program can be arranged in a more structured (more modular) way and a more effective manner.



Definition

- **Modular**: a **group of statements** that is used to proceed certain tasks, **grouped separately**, has a **certain name**, so that if it is needed in a program to perform that tasks, then we can simply **call** the name of the function
- **Effective**: If that tasks are performed **repeatedly**, then rather than writing the tasks over and over, we can simply call the function name.

Function Declaration

```
static ReturnDataType functionName() {
        // statement
        // statement
}
```

- **Static** : A static function, so that it can be directly called in the main function which is also static
- **ReturnDataType:** the data type of the value that will be returned (output) after the function is executed
- **functionName()**: the name of the function that will be created

Example :

Function Declaration

```
static void beriSalam(){
    System.out.println("Halo! Selamat Pagi");
}
```

Function Call

```
public static void main(String[] args){
    beriSalam();
}
```

Function Type

1
- a.Function with Parameter
- b.Function without Parameter

2
- a.Function with a Return Value
- b.Function without any Return Value

Function Parameter

- Function needs parameter to get the input data comes from the outside of the function.
- Function can have no or many paramater
- The number of parameter of the function, depends on the need
- Parameter declared by using the following format:

  datatype parameterName.

1.a. Function Without Parameter

- Function that does not need any input data that come from aoutside function. All the data needed by function, is already provide inside the function

No parameter inside the parenthesis

```
static void beriSalam(){
    System.out.println("Halo! Selamat Pagi");
}
```

1.b. Function with Parameter...(1)

- Parameter is a variable to get/hold the input data, passed from the outside function. The data passed to the parameter would be processed in the function. Thus, the parameter acts as *input* for the function
- Declaration:

```
static TypeDataKembalian namaFungsi(TipeData namaParameter,
    TipeData namaParameterLain){
    // statement
}
```

1.b. Function with Parameter...(2)

- **Parameter** : is a place for input of the function. It will get the input passed from the function call
- Parameter declared inside the *parenthesis* (...), after the function name.
- Parameter declared by using the following format

  datatype parameterName

- If there are more than one parameter, the parameters will be separated by a comma

1.b. Function with Parameter ...(3)

- **Example:**

Function with parameter declaration:

```
static void beriUcapan(String ucapan){
    System.out.println(ucapan);
}
```

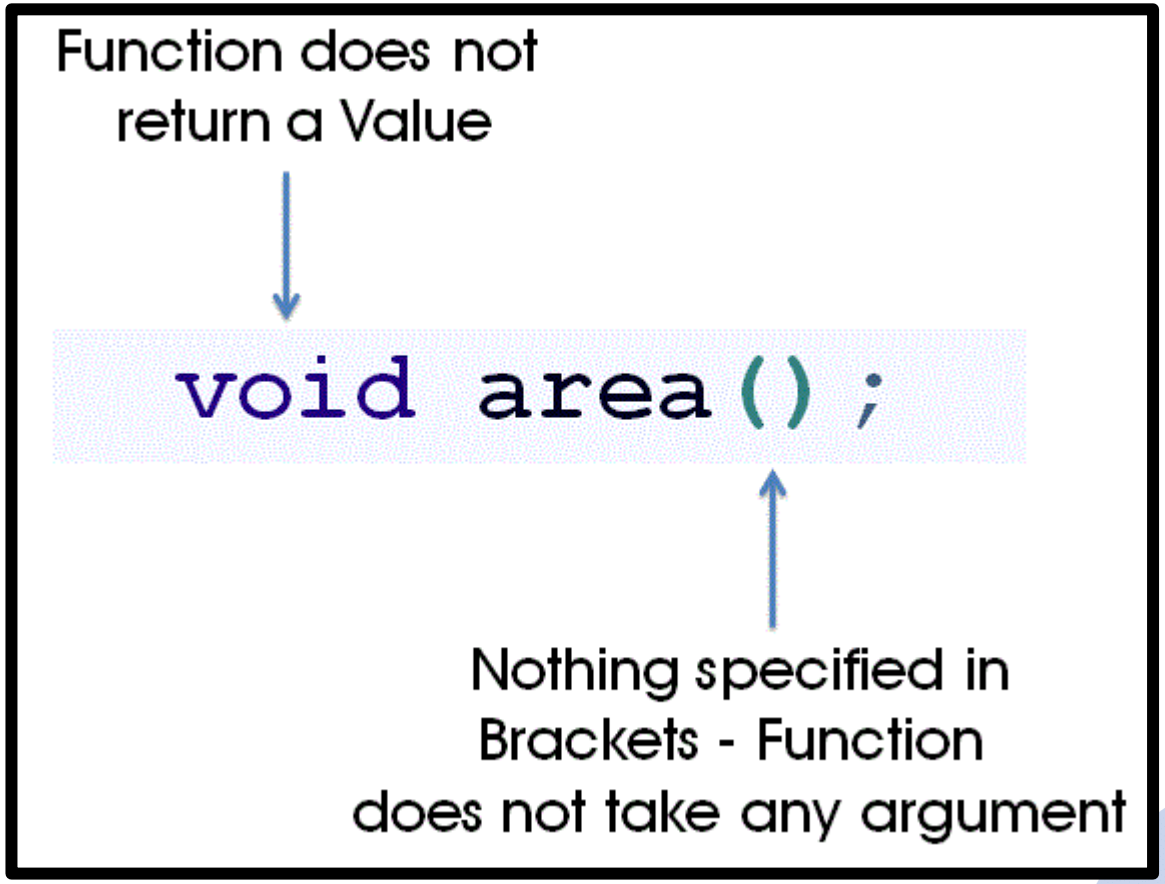
Function with parameter call:

```
String halo = "Hallo!";
beriUcapan(halo);
beriUcapan("Selamat datang di pemrograman Java");
```

2.a. Function without a Return Value

- Function with **void** datatype **does not need return** value.

```
static void functionName
(datatype parameterName)
{
    // statement
    // statement
}
```



2.b. Function with a Return Value...(1)

- A function **can return** an output/value so that it can be processed further.
- A function return a value by using **return** keyword.
- A function with **void** datatype **does not require** any **return** statement.
- A function with **non void** datatype, **must have return** statement. The value returned, **must match** with the function datatype. For example, if the data type of the function is int, then the returned value must be int value

2.b. Function with a Return Value…(2)

- Function Declaration

```
static TypeDataKembalian namaFungsi (TipeData namaParameter){
        // statement
        return variabelOutput;
}
```

2.b. Function with a Return Value...(3)

- **Example**

Function with a parameter and a return value:

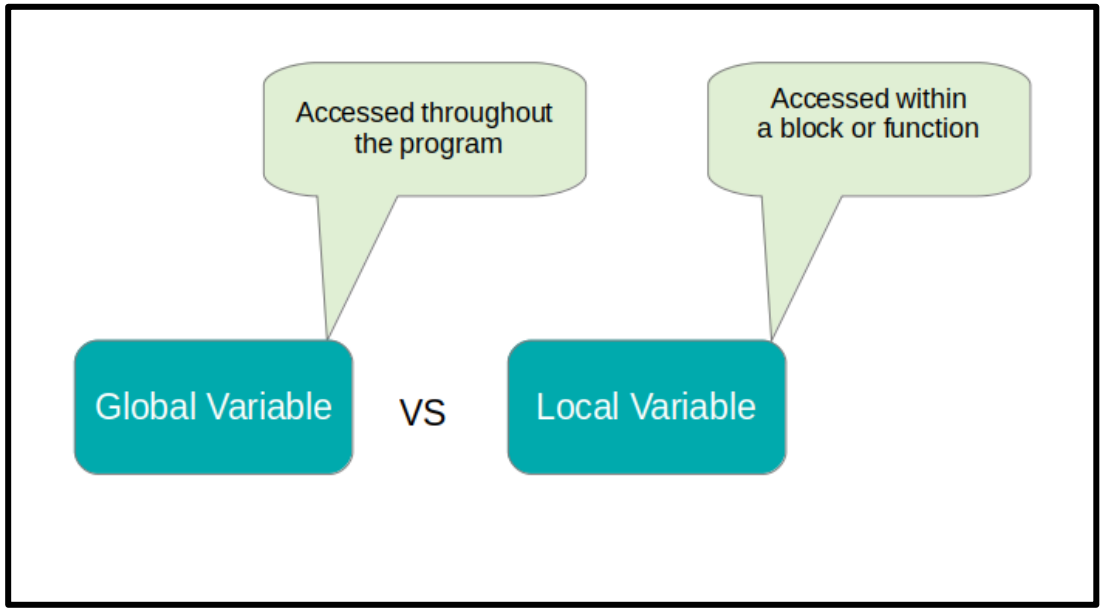
```
static int luasPersegi(int sisi){
    int luas = sisi * sisi;
    return luas;
}
```

Function call:

```
System.out.println("Luas Persegi dengan sisi 5 = " + luasPersegi(5));

int luasan = luasPersegi(6);
```

SCOPE OF VARIABLE... (1)

- Local Variables: variables that are declared within a function and can only be accessed or recognized from the function itself.
- Global Variables: Variables that are declared outside the function block and can be accessed or recognized from any function.
- Global variables in java are prefixed with **static** so that they can be called or accessed directly from main function



SCOPE OF VARIABLE... (2)

```
public class Fungsi1 {

    static int a = 10, b = 5;
    static double c;

    static int Kali() {
        int hasilKali = a * b;
        return hasilKali;

    }

    static void Tambah() {
        int hasilTambah = a + b;
        System.out.println("Hasil Tambah adalah " + hasilTambah);
    }

    public static void main(String[] args) {
        System.out.println("Hasil Kali adalah " + Kali());
        Tambah();

    }

}
```

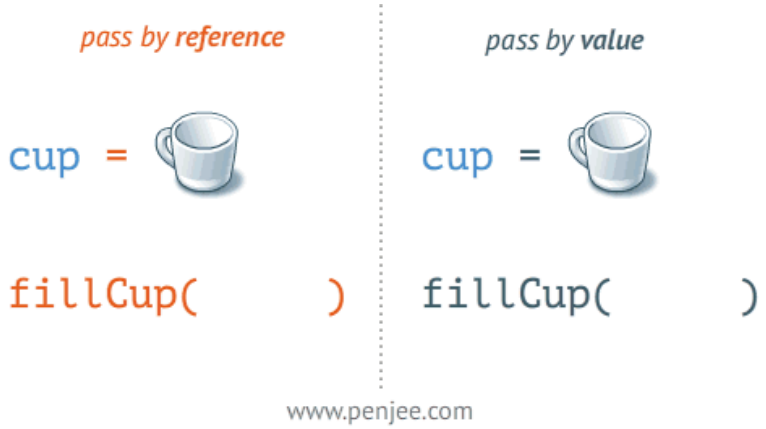
Global Variables

Local Variable

Local Variable

Pass by Value Vs Pass by Reference Function

- **Pass by Value**
  - In pass by value, the **actual value** of the argument is passed to the function.
  - The function receives **a copy of the value**, and any changes made to the parameter inside the function do not affect the original value outside the function.
  - This is a common approach in languages like C, C++, and Java for passing primitive data types (e.g., integers, floating-point numbers).
- **Pass by Reference**
  - In pass by reference, instead of passing the actual value, the function receives a **reference (or a memory address)** to the variable.
  - Any changes made to the parameter inside the function directly affect the original value outside the function.
  - This is common in languages like C++ with references or in Java for passing array and reference/object datatype



Pass by Value Vs Pass by Reference Function

```java
public class PassbyValue {
    static void UbahNilai(int j){
        j=33;
    }

    public static void main(String[] args) {
        int i=10;
        System.out.println(i);
        UbahNilai(i);
        System.out.println(i);
    }
}
```

```
run:
10
10
```

```java
public class PassbyRef {
    static void UbahArray (int[] arr){
        for (int i=0; i<arr.length; i++){
            arr[i]=i+50;
        }
    }

    public static void main(String[] args) {
        int[] umur = {10, 11, 12};

        for (int i = 0; i < umur.length; i++) {
            System.out.println(umur[i]);
        }

        UbahArray(umur);
        for (int i=0; i<umur.length; i++){
            System.out.println(umur[i]);
        }
    }

}
```

```
run:
10
11
12
50
51
52
```

A Function can CALL Other Functions

```
public class FungsiCall {
    public static void main(String[] args) {
        int hasil=Hitung(5,2);
        System.out.println("Hasil Akhirnya adalah "+hasil);
    }

    static int Tambah(int x, int y){
        int Z=x+y;
        return Z;
    }

    static int Hitung (int c, int d){
        int E;
        c*=2;
        d*=2;
        E=Tambah(c,d);
        return E;
    }
}
```

```
run:
Hasil Akhirnya adalah 14
BUILD SUCCESSFUL (total time: 2 seconds)
```

Two Functions can CALL Each Other

```
public class FungsiCallFungsi {
    public static void main(String[] args) {
        int hasil=Hitung(5,2);
        System.out.println("Hasil Akhirnya adalah "+hasil);
    }

    static int Tambah(int x, int y){
        int Z=x+y;
        while (Z<50){
            x+=2;
            y+=2;
            Z=Hitung(x,y);
        }
        return Z;
    }

    static int Hitung (int c, int d){
        int E;
        c*=2;
        d*=2;
        E=Tambah(c,d);
        return E;
    }
}
```

```
run:
Hasil Akhirnya adalah 80
BUILD SUCCESSFUL (total time: 2 seconds)
```

Java Varargs (Variable Arguments)

- In Java, "varargs" (variable-length argument lists) is a feature that allows a method to accept **a variable number of arguments/parameters**.
- This feature was introduced in Java 5 as part of the enhanced for loop and is denoted by an **ellipsis (...)** followed by the argument type in the function parameter.
- Declaration:

```
accessModifier methodName(datatype… arg) {
      // method body
}
```

Example

```java
public class ContohVarargs1 {
    static void tampilIsi(int ...a){
        System.out.println("Jumlah parameter ada "+ a.length);
        System.out.println("isinya : ");

        for(int i=0; i<a.length; i++){
            System.out.println("Parameter ke-"+i+" : "+a[i]);
        }

        System.out.println();
    }

    public static void main(String args[]){
        tampilIsi(10);  // hanya ada satu parameter
        tampilIsi(4,5,3); // ada 3 parameter
    }

}
```

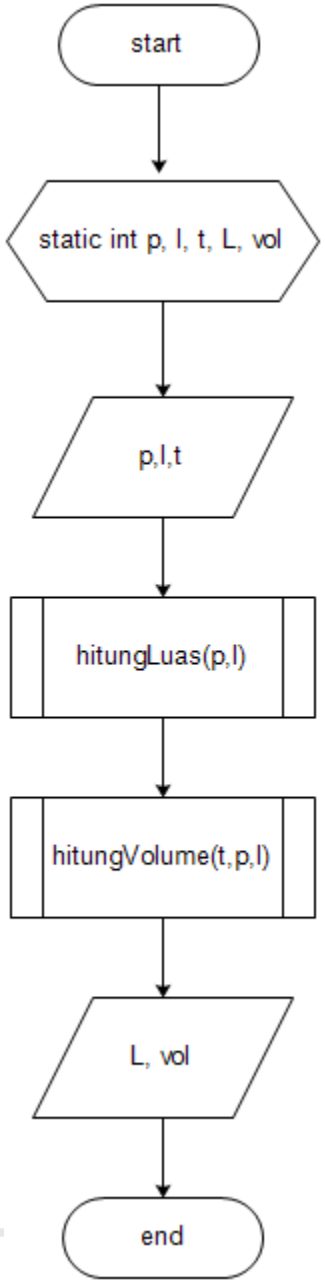
```
run:
Jumlah parameter ada 1
isinya :
Parameter ke-0 : 10

Jumlah parameter ada 3
isinya :
Parameter ke-0 : 4
Parameter ke-1 : 5
Parameter ke-2 : 3
```

Function Flowchart

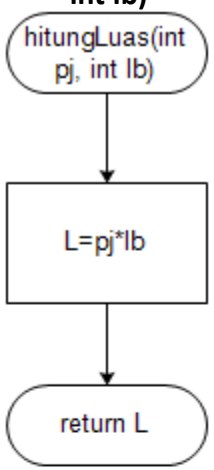
Example 1

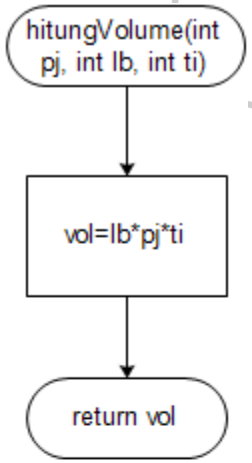
- flowchart to calculate the **area** of a square and the **volume** of a cuboid using functions.

Flowchart : main()



Flowchart : hitungLuas (int pj, int lb)





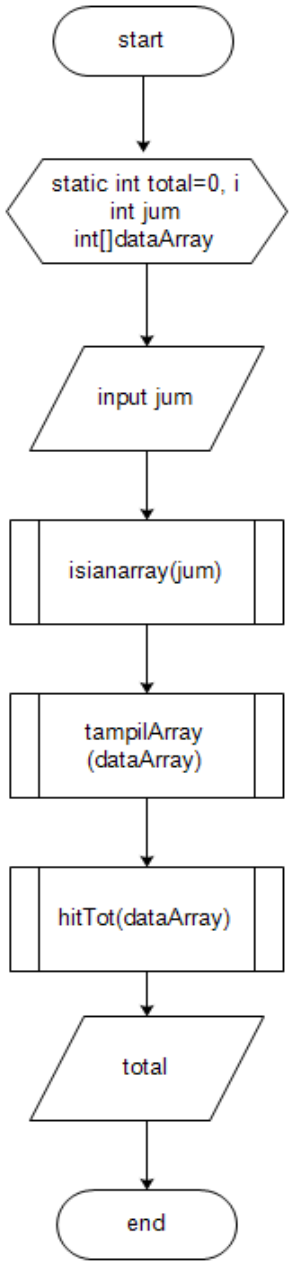
Flowchart : hitungVolume (int pj, int lb, int ti)

Example 2

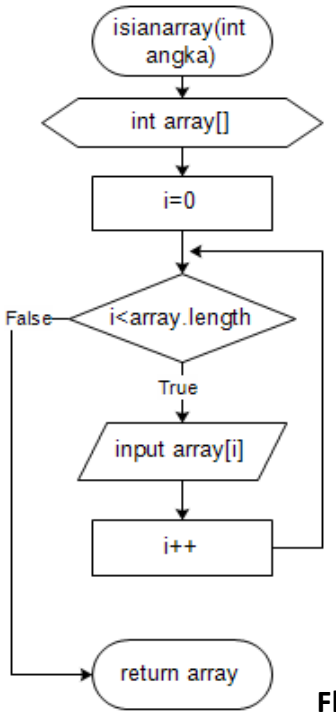
- flowchart to calculate the **total grade** of N students, that consists of 3 functions:
  - Input function N grades (N is the number of grades entered) → **isianarray()**
  - Show grades → **tampilArray()**
  - Total grades → **hitTot()**

Example 2

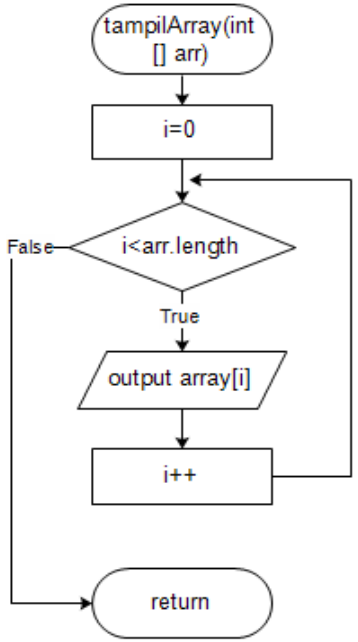
Flowchart : main()



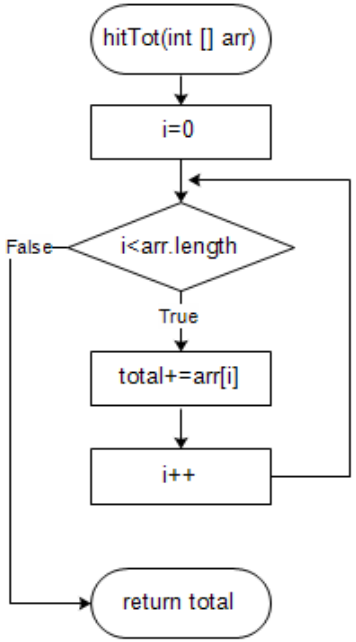
Flowchart : isianarray (angka)



Flowchart : tampilArray (int[]arr)



Flowchart : hitTot (int[]arr)



Assignments:

1. Create a flowchart to show a series of even numbers between 1-100, using function!
2. A fruit shop selling its fruit in 1 week of sales yields the following data:

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Apple	20	19	25	20	10	0	10
Orange	30	30	40	10	15	20	25
Grape	5	0	20	25	10	5	45
Kiwi	50	0	7	8	0	30	60
Guava	15	10	16	15	10	10	5

Create a flowchart to display the day when the biggest number of fruit sold, and to display the best-selling fruit sold at the 1st to 7th day. The flowchart will consist of 4 functions:

a) Function to input fruit sales data

b) Function to display all fruit data sold

c) Function to find on what day when the biggest number of fruit sold

d) Function to display the best-selling fruit in each day