Function 1

Basic Programming Teaching Team 2022

Objectives

After studying this material, students should be able to:

- Understand the concept of function
- Master how to declare functions
- Master how to call functions

Function Definition

- A function is a number of instructions that are grouped together, stand alone, which aims to complete a particular task.
- Using functions, programs can be structured in a more structured (more modular) and more effective manner.

Function Definition

- **Modular**: A group of statements that function to carry out certain tasks, grouped individually and separately, given a specific name. If the function is required to perform the task, then we can call the function name.
- **Effective**: If the task is performed repeatedly in the program, it does not need to be written over and over again, but what is done is just calling the function.

Function Declaration

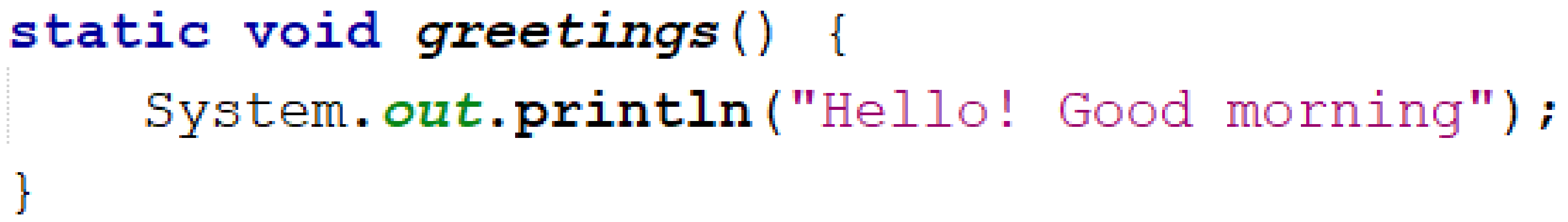
```
static ReturnDataType functionName() {
   //statement
   //statement
}
```

Information:

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

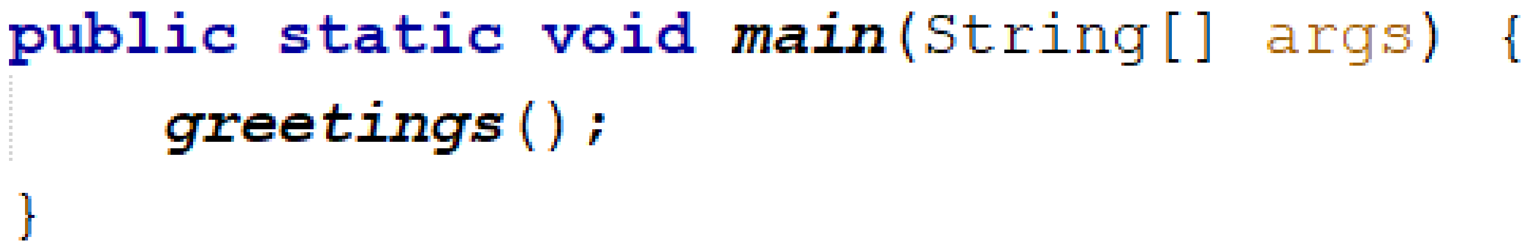
Example of Function

Function Creation



```java
static void greetings() {
    System.out.println("Hello! Good morning");
}
```

Function Call



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

Function with Parameters

- **Parameters** are variables that hold values to be processed in a function. Parameters serve as inputs to a function.
- Declaration:

```
static ReturnDataType functionName(DataType parameterName1,
DataType parameterName2) {
   //statement
   //statement
}
```

Function with Parameters

- Parameters serve as a place for input data to be processed in a function. The number of parameters according to needs. Each parameter consists of a data type and a parameter name, for example int a, float b, the writing method is the same as a variable declaration.
- Parameters are written between the parenthesis (...) after the function name.
- If there is more than one parameter, then separated by a comma and each parameter must describe its data type.

Function with Parameters

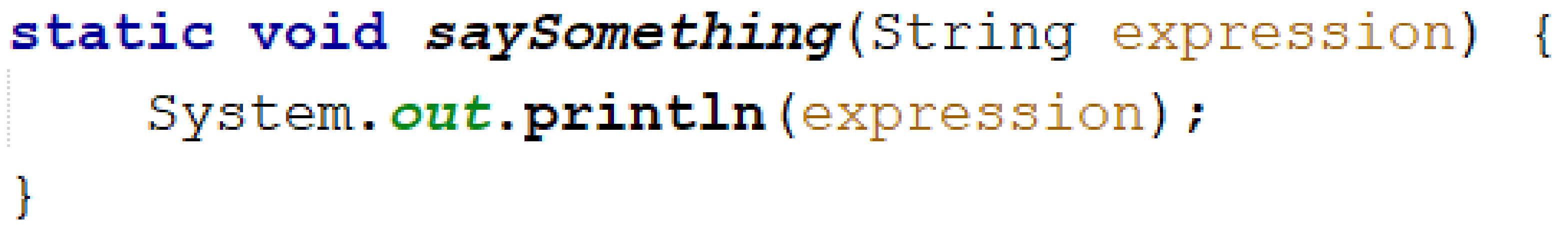
- Functions require parameters when the function requires data that comes from outside the function to be processed in the function.
- The function **may** have no parameters at all.
- The number of function parameters that a function can have according to needs, and there is no maximum limit.
- When declaring a function, write parameters as follows:

  dataType parameterName

JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

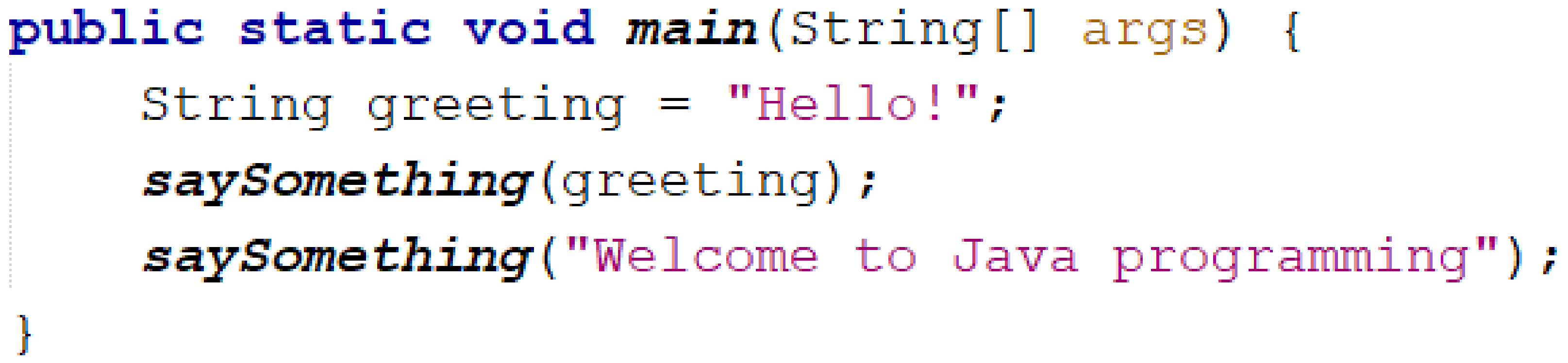
Function with Parameters

Function creation with parameters



```java
static void saySomething(String expression) {
    System.out.println(expression);
}
```

Call the function and provide parameter values



```java
public static void main(String[] args) {
    String greeting = "Hello!";
    saySomething(greeting);
    saySomething("Welcome to Java programming");
}
```

Function that Return Value

- A function can return the output value so that it can be processed in the next process.
- Returns the value of the function using the **return** keyword.
- Functions that have a function data type **other than void that require return**. Functions **with the void data type do not require return**.
- **The value returned** from a function must **match the function's data type**. For example, if the function's data type is int, the returned value must be int as well.

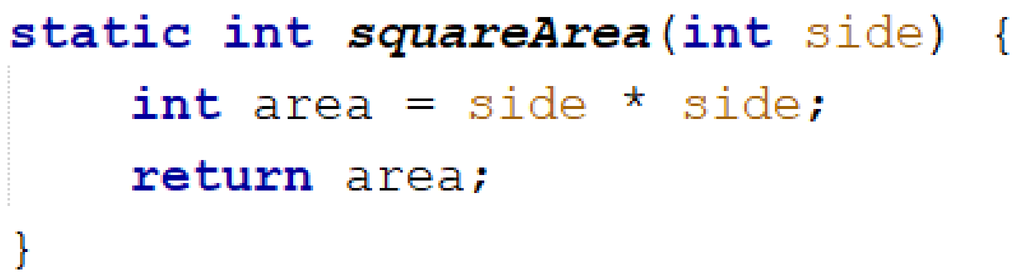
Function that Return Value

- A function can return the output value so that it can be processed in the next process.
- Returns the value of the function using the **return** keyword.
- Function declaration:

```
static ReturnDataType functionName(DataType parameterName){
   //statement
   return outputVariable;
}
```

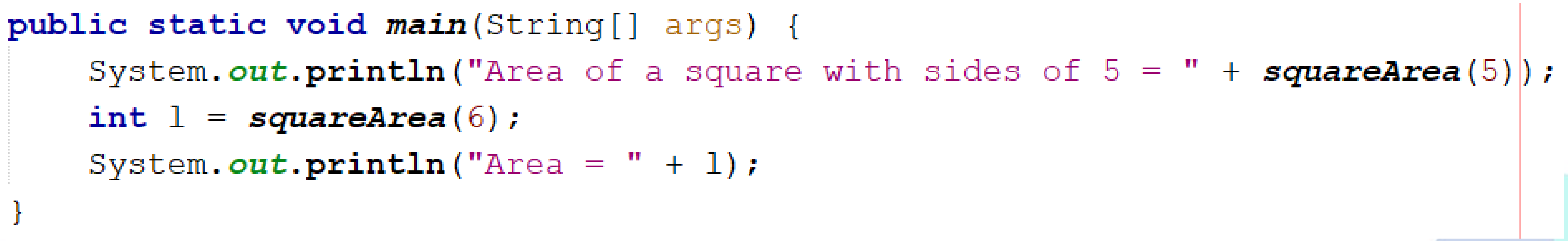
Function that Return Value

Creating a function with parameters and return value



```java
static int squareArea(int side) {
    int area = side * side;
    return area;
}
```

Function Call and assign parameter values

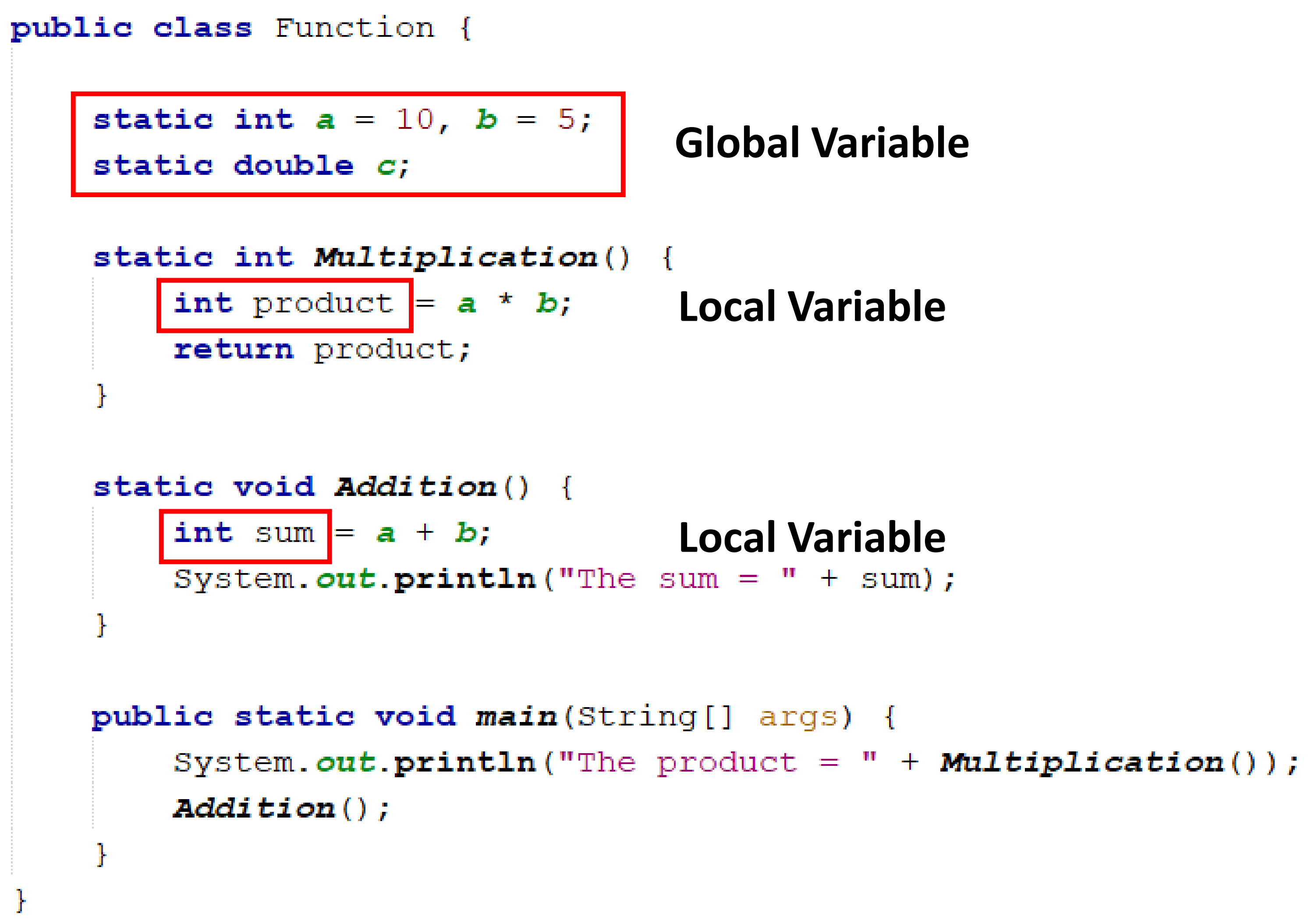


```java
public static void main(String[] args) {
    System.out.println("Area of a square with sides of 5 = " + squareArea(5));
    int l = squareArea(6);
    System.out.println("Area = " + l);
}
```

Scope of Variable

- **Local variables** are variables declared in a function, and can only be accessed or recognized from within the function itself.
- **Global variables** are variables declared outside the function block, and can be accessed or recognized from any function.
- Global variables in Java are prefixed **static** so they can be called directly.

Scope of Variable



```
public class Function {

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

    static int Multiplication() {
        int product = a * b;
        return product;
    }

    static void Addition() {
        int sum = a + b;
        System.out.println("The sum = " + sum);
    }

    public static void main(String[] args) {
        System.out.println("The product = " + Multiplication());
        Addition();
    }
}
```

Global Variable

Local Variable

Local Variable

Pass by Value Vs Pass by Reference Function

- **Pass by Value** sends a parameter based on **the value of the original variable** which will be linked to the caller's function parameter.
- **Pass by Reference** sends parameters based on the **address of a certain value**, therefore if a value is changed from the original address, there will also be changes to the value of the parameter called.

Pass by Value Vs Pass by Reference Function

```java
public class PassByValue {

    static void changeValue(int j) {
        j = 33;
    }

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

```
run:
10
10
```

```java
public class PassByReference {

    static void changeArray(int[] arr) {
        for (int i = 0; i < arr.length; i++) {
            arr[i] = i + 50;
        }
    }

    public static void main(String[] args) {
        int[] age = {10, 11, 12};
        for (int i = 0; i < age.length; i++) {
            System.out.println(age[i]);
        }
        changeArray(age);
        for (int i = 0; i < age.length; i++) {
            System.out.println(age[i]);
        }
    }
}
```

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

A Function can CALL Another Function

```
public class FunctionCall {

    public static void main(String[] args) {
        int result = Calculate(5, 2);
        System.out.println("The final result = " + result);
    }

    static int Addition(int x, int y) {
        int z = x + y;
        return z;
    }

    static int Calculate(int c, int d) {
        int e;
        c *= 2;
        d *= 2;
        e = Addition(c, d);
        return e;
    }
}
```

```
run:
The final result = 14
```

Two Functions can CALL Each Other

```java
public class FunctionCallFunction {

    public static void main(String[] args) {
        int result = Calculate(5, 2);
        System.out.println("The final result = " + result);
    }

    static int Addition(int x, int y) {
        int z = x + y;
        while (z < 50) {
            x += 2;
            y += 2;
            z = Calculate(x, y);
        }
        return z;
    }

    static int Calculate(int c, int d) {
        int e;
        c *= 2;
        d *= 2;
        e = Addition(c, d);
        return e;
    }
}
```

```
run:
The final result = 80
```

Java Varargs (Variable Arguments)

- Varargs are used by placing parameters in an array and that array will become the parameters of the function.
- If we don't know the exact number of parameters of a function, we can use the **Variable Length Argument** (Varargs).
- Varargs can only be used on several parameters that have the same data type
- Declaration:

```
static ReturnDataType functionName(DataType... arg){
   //statement
}
```

Example of Varargs

```java
public class FunctionVarargs {

    static void showContents(int... a) {
        System.out.println("Number of parameters = " + a.length);
        System.out.println("The contents are ");
        for (int i = 0; i < a.length; i++) {
            System.out.println("Parameter " + i + ": " + a[i]);
        }
        System.out.println("");
    }

    public static void main(String[] args) {
        showContents(10);
        showContents(4, 5, 8);
    }
}
```

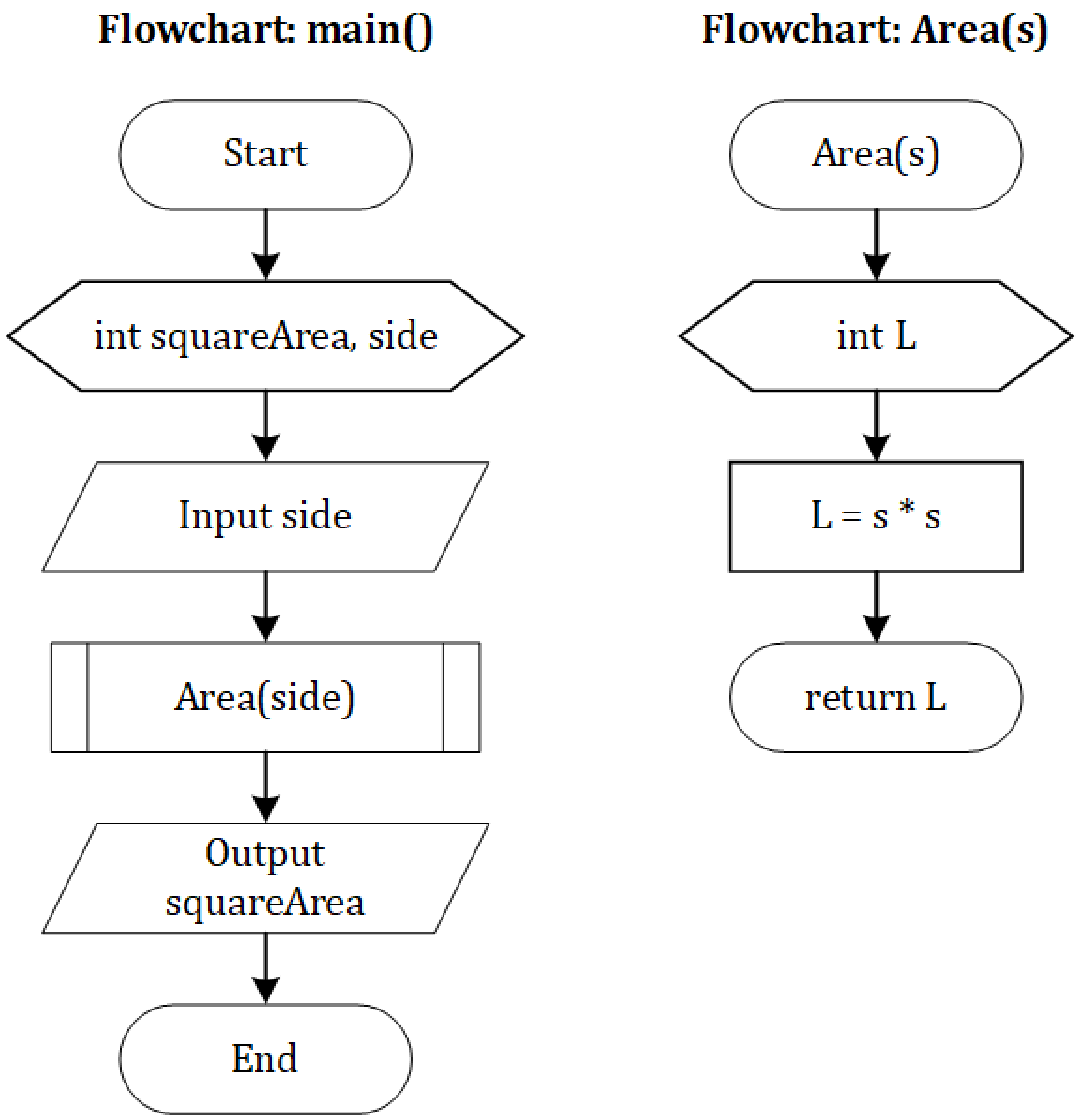
```
run:
Number of parameters = 1
The contents are
Parameter 0: 10

Number of parameters = 3
The contents are
Parameter 0: 4
Parameter 1: 5
Parameter 2: 8
```

Example of Function Flowchart

- Create a flowchart to calculate the area of a square. To calculate area, use the function.

Example of Function Flowchart



Assignment

1. Create a flowchart to calculate the surface area and volume of the block (surface area and volume of the block using functions)!
2. Create a flowchart to display an even number series between 1-100 (the series displayed using the function)!
3. Create a flowchart to input student names and grades (using an array) then calculate the class average score. The functions consist of:
   a. The function displays student data
   b. The function calculates the class average grade