Array 1

Basic Programming Teaching Team 2022

Objectives

After studying this material, students should be able to:

- Understand the concept of 1-dimensional arrays
- Provide examples of the use of 1-dimensional arrays
- Solve simple searching and sorting case studies

Preface

- In mathematics, in a matrix that has matrix elements, the matrix elements are written using indexed variables.
- Suppose a matrix A [5,5] with dimension 5x5 will have matrix elements, namely a_{00} to a_{44}
- In computer programming, the implementation of indexed variables uses arrays. So that the array can be one or more dimensions.

Definition

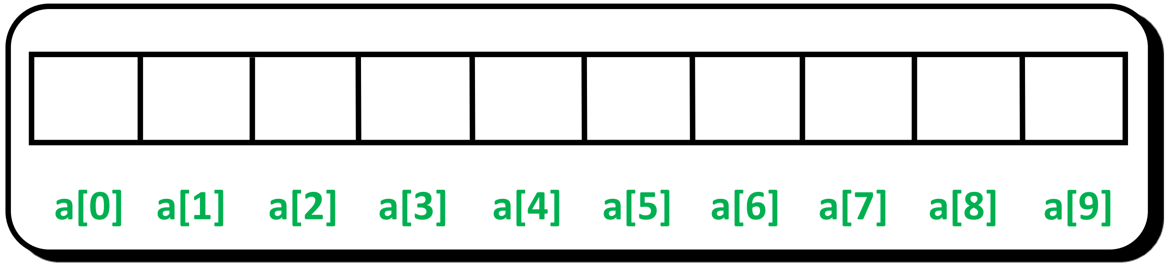
- An array is a complex variable with the **same data type**, using the **same name**, and having a **certain index**.
- In other words, an array is a set of values (elements) with the same data type, where each array element can be accessed using a unique index.

Array Properties

- Homogeneous
  - All elements in the array structure have the same data type.
- Random Access
  - Each element in the array structure can be accessed individually, directly to the desired element location, not necessarily through the first element.
- Is a reference variable

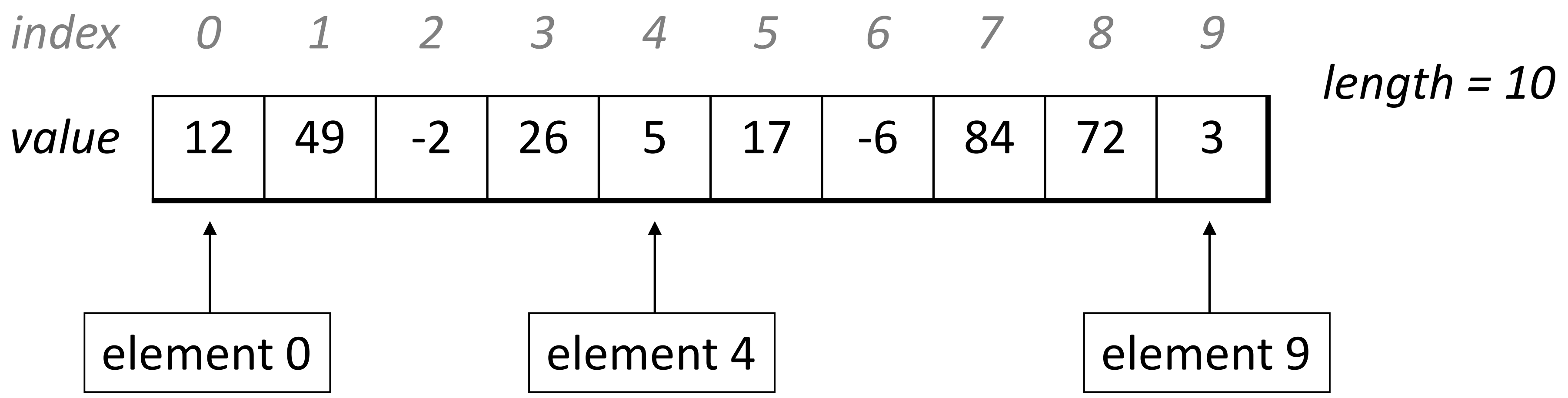
Array Visualization

- Suppose there is an array named a with 10 elements (N = 10), the array elements can be described as follows:



- The empty box shows the elements of the Array
- Each element has a numbering 0-9 (index)
- Array index starts at 0 and ends with N-1

Array Visualization



One Dimensional Array Declaration

- Declaration

```
    dataType arrayName[];
```

or

```
    dataType[] arrayName;
```

Example: `int a[]; int[] a;`

- **dataType** is the data type of the array to be created
- **arrayName** is the name of the array to be created

One Dimensional Array Instantiation

Array object instantiation:

- When an array is declared, only references from the array are created. Meanwhile, memory allocation is done using the **new** keyword
- How to instantiate array variables:

```
arrayName = new dataType[numberOfElements];
```

Example: `a = new int[10];`

One Dimensional Array

- The declaration and instantiation of an array object can be combined in an instruction as follows:

```
dataType[] arrayName = new dataType[numberOfElements];
```

or

```
dataType arrayName[] = new dataType[numberOfElements];
```

- Example:

```
int[] a = new int[10];
```

or

```
int a[] = new int[10];
```

Accessing Array Elements

- Refers to the index number

```
arrayName[index]
```

- Example:
  - Accessing an array variable **a** with index **i** can be written:
    ```
    a[i]
    ```
  - Index **i** can only be **0 or positive** with the maximum value is **numberOfElements - 1**

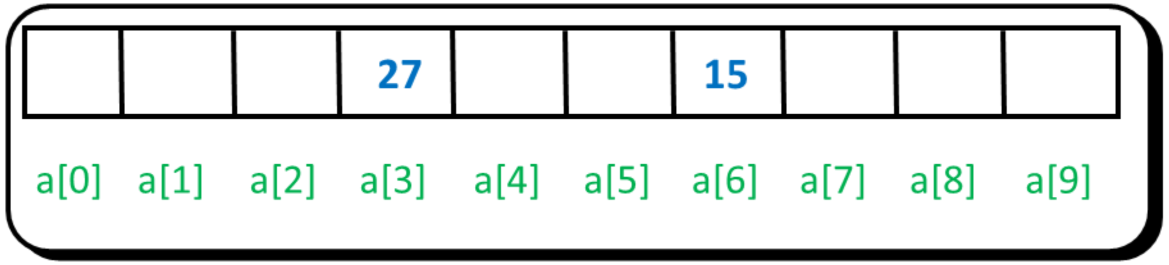
Accessing Array Elements

- Example:

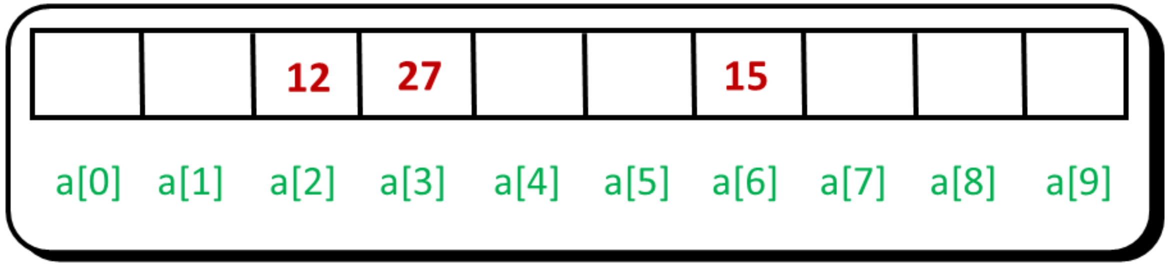
```
String[] cars = {"Volvo", "BMW", "Ford”};
System.out.println(cars[0]);  //displays Volvo
System.out.println(cars[2]);  //displays Ford
```

Fill in Data on Array

- Filling data to array elements is done using assignment operators
- Example: `a[6] = 15;   a[3] = 27;`

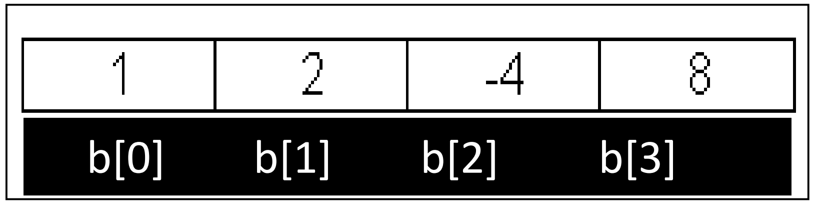


- Statement `a[2] = a[3] - a[6];` resulting:

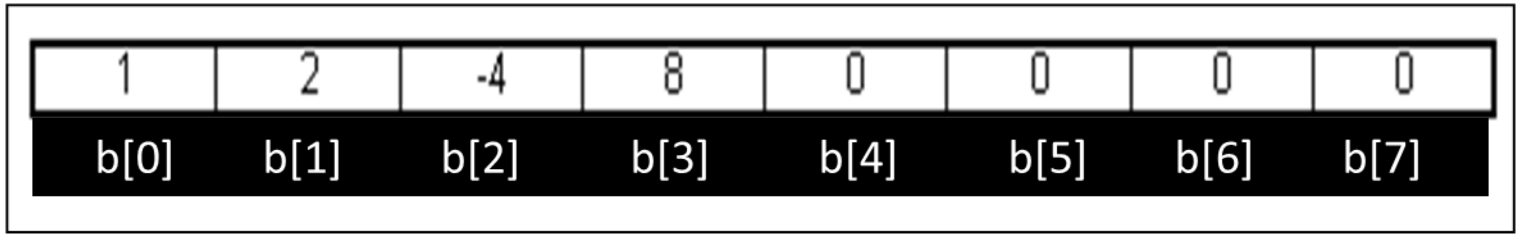


Array Initialization

- Arrays can be explicitly initialized when defined and may not be assigned a dimension value.
- Example: `int b[ ]={1, 2, -4, 8};`



- Example: `int b[]={1, 2, -4, 8,0,0,0,0};`



Example of Array Initialization

- `boolean results[] = { true, false, true, false };`
- `String[] cars = {"Volvo", "BMW", "Ford”};`
- `int[] myNum = {10, 20, 30, 40};`
- `double []grades = {100, 90, 80, 75};`
- `String days[] = { “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”, “Sunday”};`

Replacing Array Elements

- `String[] cars = {"Volvo", "BMW", "Ford”};`

Result:

Volvo	BMW	Ford
cars[0]	cars[1]	cars[2]

- `cars[0] = "Opel";`

Result:

Opel	BMW	Ford
cars[0]	cars[1]	cars[2]

Get the Length of an Array

- You can get the length of an array using

  arrayName`.length`

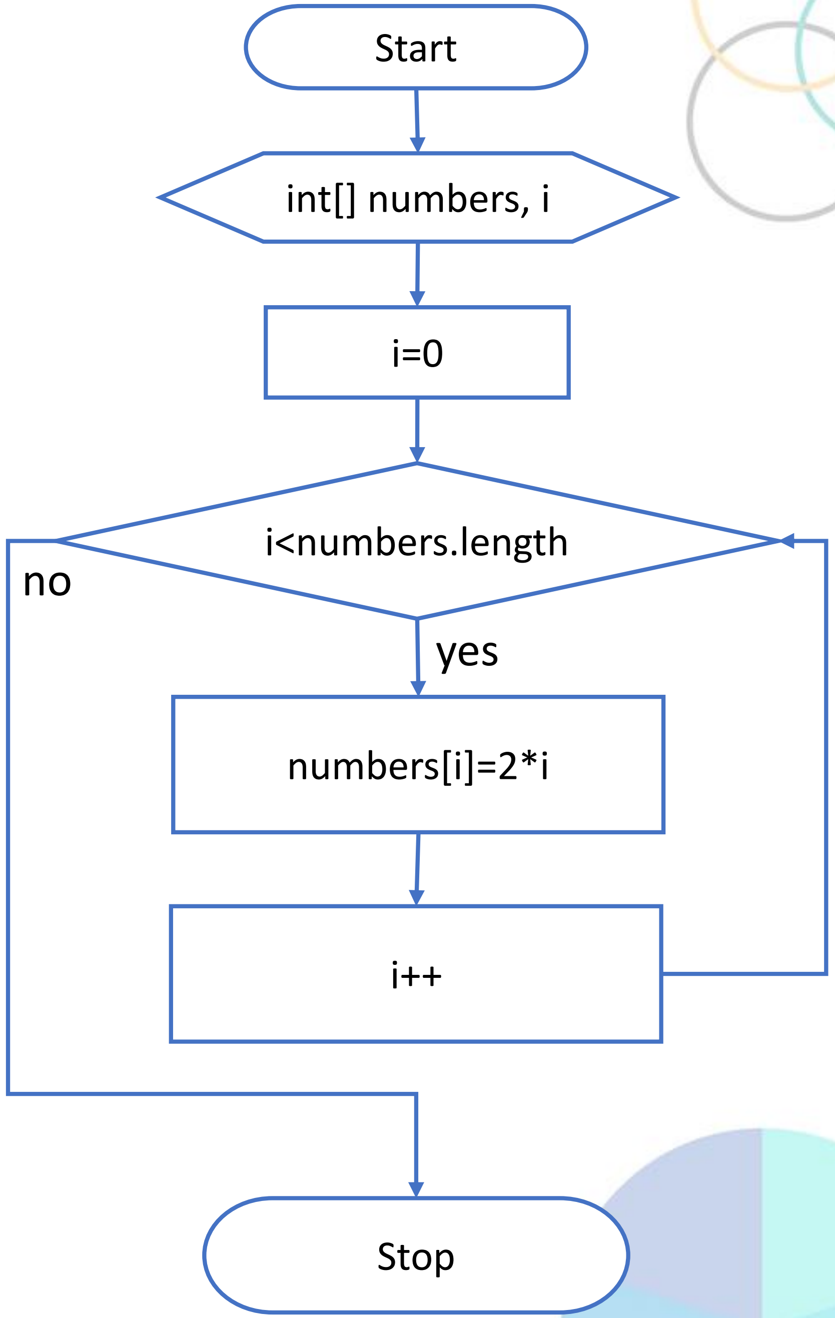
- Examples of using the length of an array:
  - What is the index of the last element of an array?
  - What is the index of the middle element of an array?

Array Loop

- We can use the length of the array, together with its index, to perform some operations using loops.
- For example, we can efficiently initialize an array.

```
int[] numbers = new int[8];
    for (int i = 0; i < numbers.length; i++){
      numbers[i] = 2 * i;
    }
```

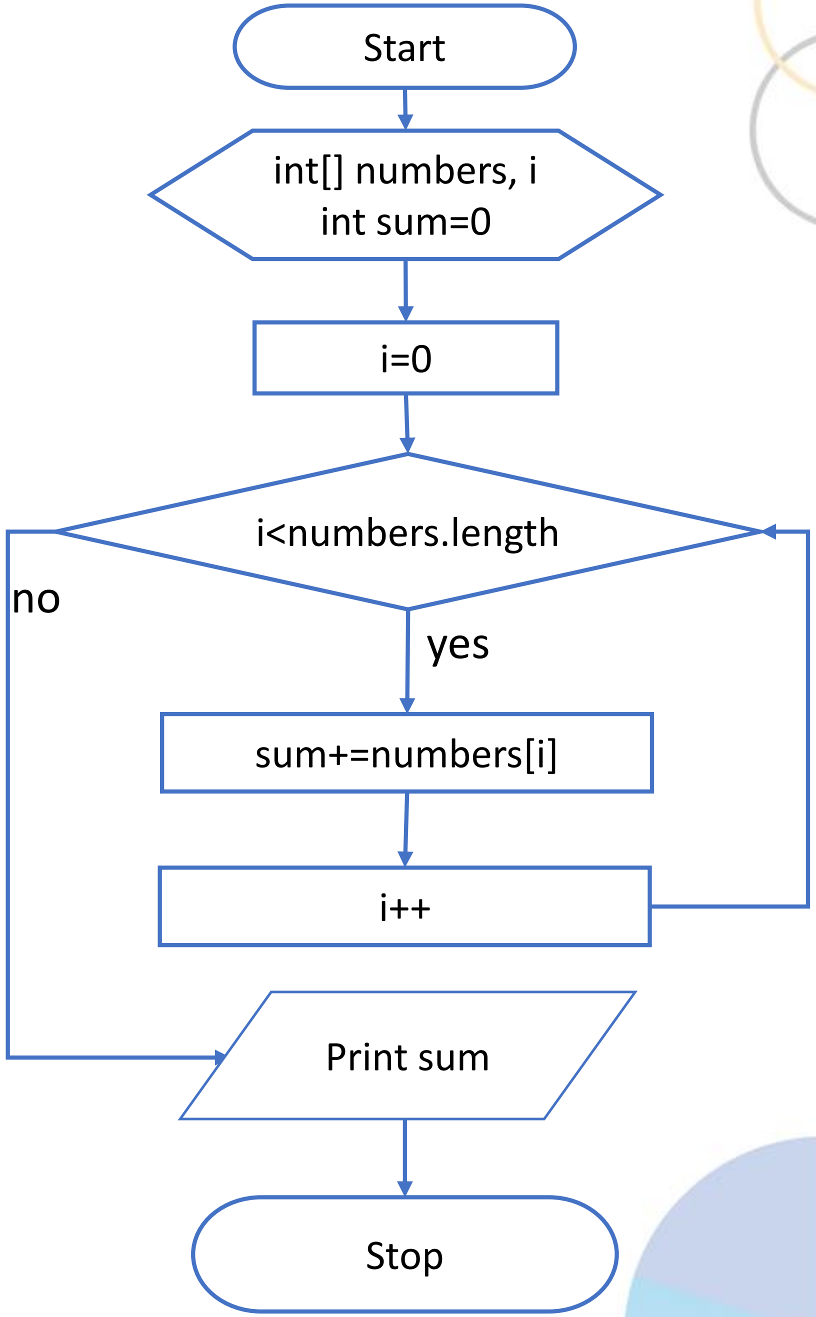
index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
value	0	2	4	6	8	10	12	14



Example of Array Loop

- Sum all the elements of the array

```
// assume that the user has created int[] numbers
int sum = 0;
for (int i = 0; i < numbers.length; i++)
{
        sum += numbers[i];
}
println(sum);
```



Array Loop

for-each loop

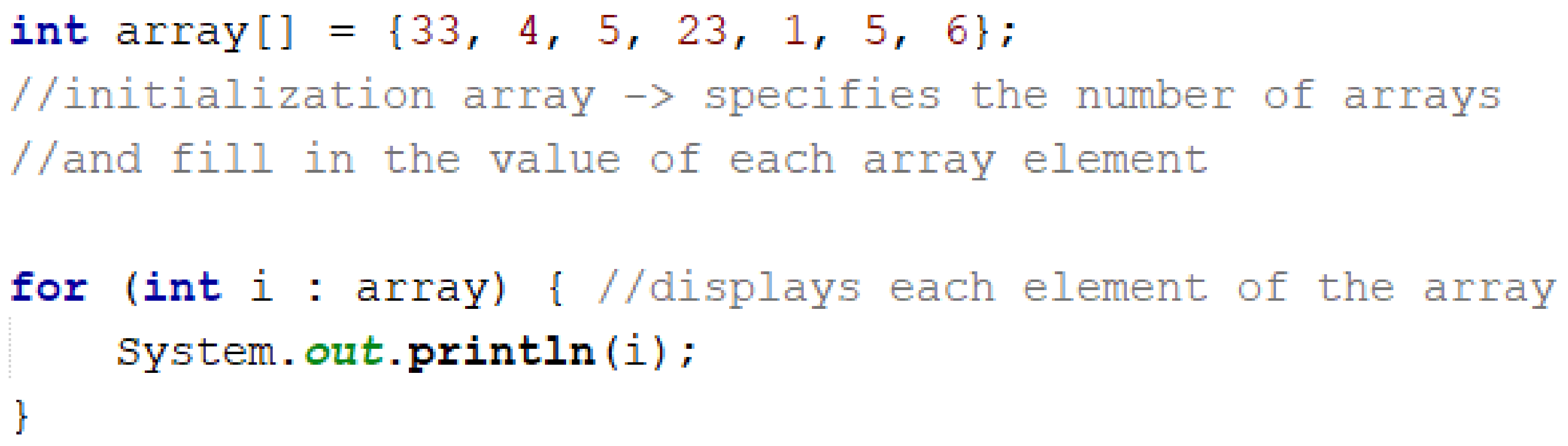
- This is another form of a for loop that is used to traverse arrays
- for-each loop reduces code significantly and there is no indexes or counters inside the loop
- Syntax:

```
for(dataType tempVar : arrayName){
    //statement
}
```

tempVar: temporary variable for looping process

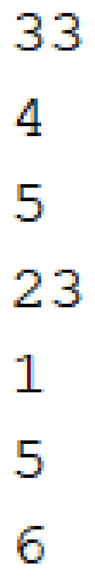
Example of Array Loop

- Access all array elements using the "**for-each**" loop



```
int array[] = {33, 4, 5, 23, 1, 5, 6};
//initialization array -> specifies the number of arrays
//and fill in the value of each array element

for (int i : array) { //displays each element of the array
    System.out.println(i);
}
```



```
33
4
5
23
1
5
6
```

Difference with or without Array

```
int number1 = 1;
int number2 = 2;
int number3 = 3;
```

<< without array

```
int number[] = {1, 2, 3};
```

<< with array

Array Usage Steps

1. Declare an array reference variable
2. Array element instantiation
3. Initialize array (if needed)
4. Manipulate array elements

Example of Incorrect Array Initialization

- Example: `int b[4] = { 1, 2, -4, 8, 9 };`
  - **ERROR** because the dimension value is smaller than the number of elements

- Example:
  - Initialize array after being defined incorrectly

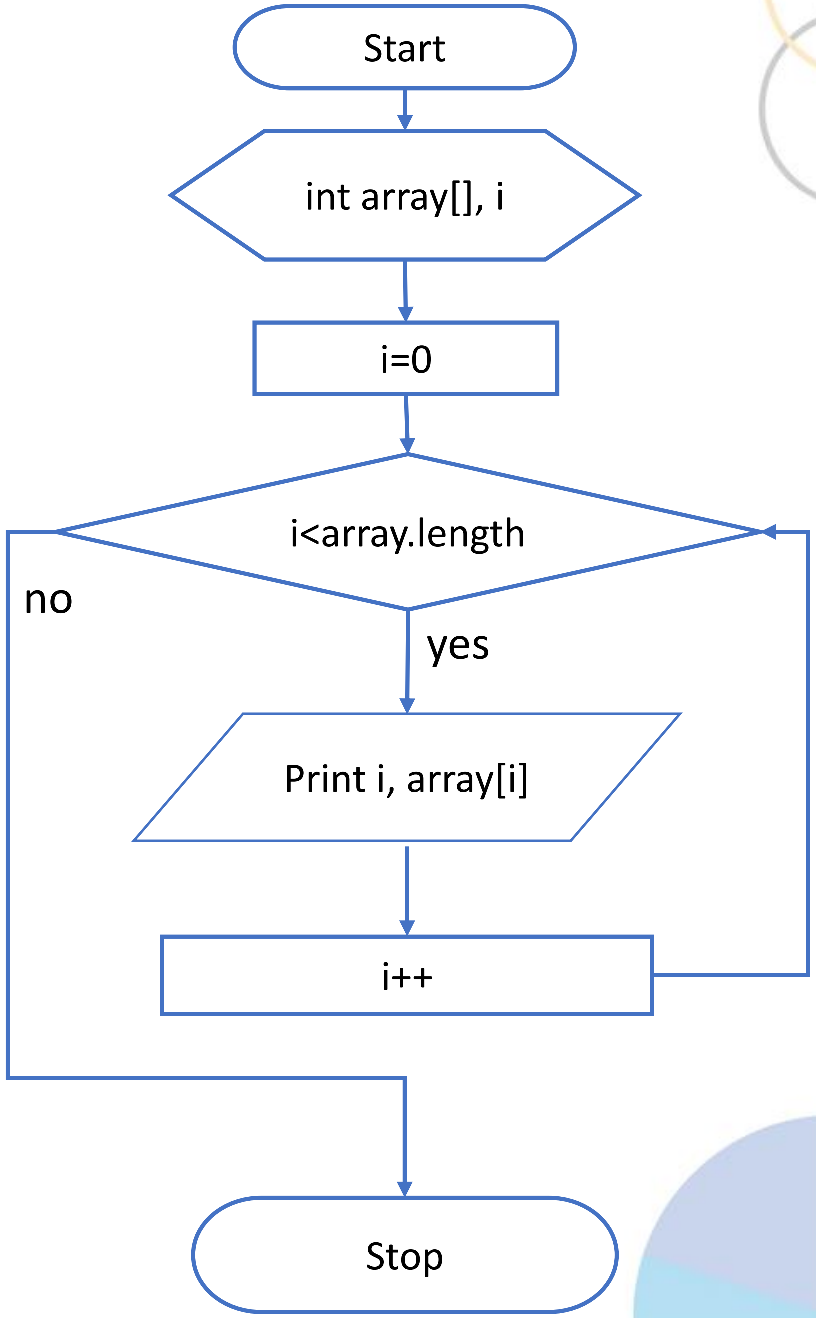
```
int b[5];
b[5]={0,0,0,0,0};
```

Examples of Using Arrays

Example 1

```
int array[]; //array declaration
array = new int[10]; //array instantiation
System.out.printf("%s%5s\n", "Index ", "Value");
//adds each element to the array and displays it
for (int i = 0; i < array.length; i++) {
    System.out.printf("%2d%5d\n", i, array[i]);
}
```

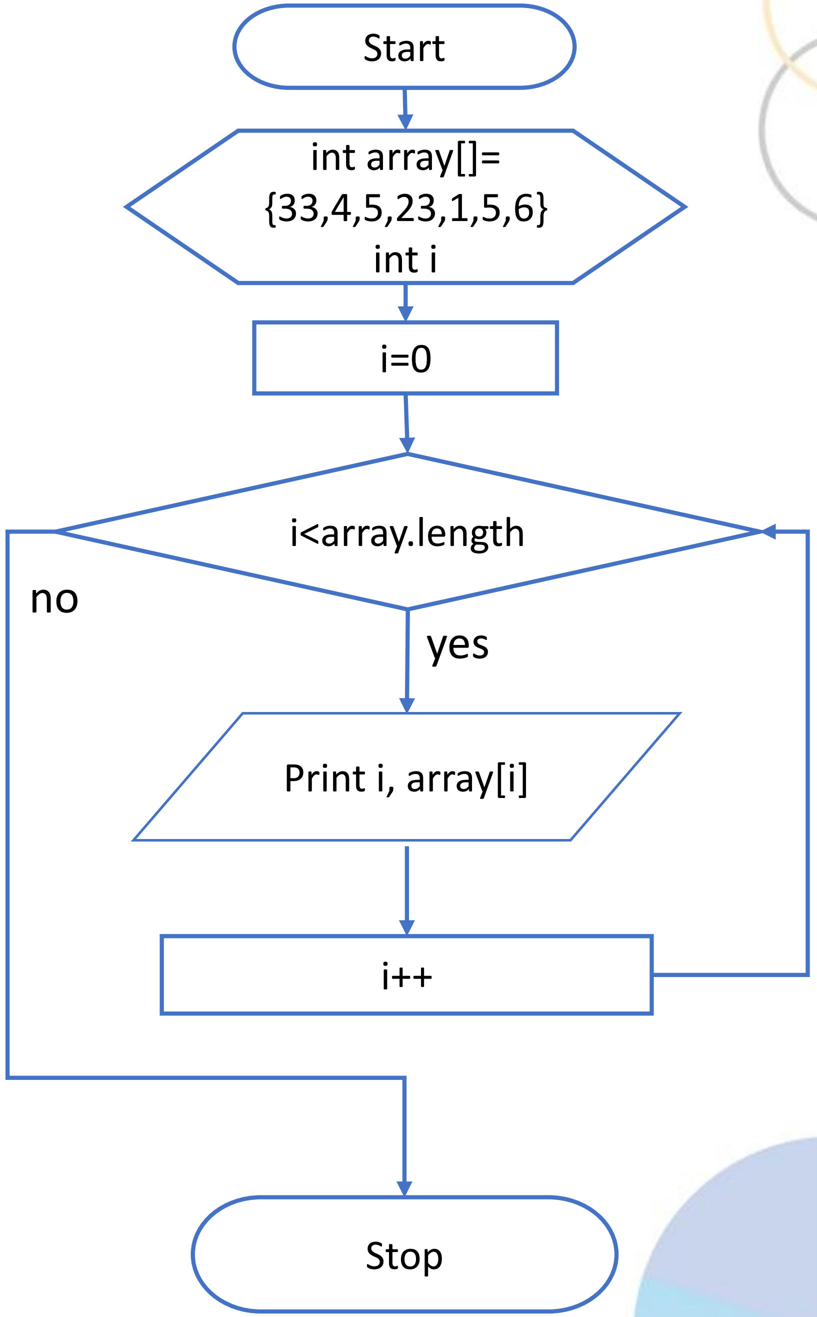
```
Index Value
 0     0
 1     0
 2     0
 3     0
 4     0
 5     0
 6     0
 7     0
 8     0
 9     0
```



Example 2

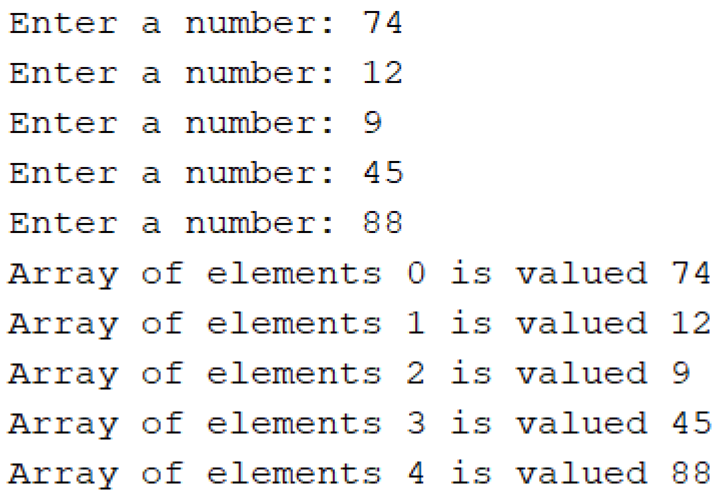
```java
int array[] = {33, 4, 5, 23, 1, 5, 6};
//initialization array -> specifies the number of arrays
//and fill in the value of each array element
System.out.printf("%s\t%s\n", "Index ", "Value");
for (int i = 0; i < array.length; i++) {
    System.out.printf("%d\t%d\n", i, array[i]);
} //displays each element of the array
```

```
Index    Value
0        33
1        4
2        5
3        23
4        1
5        5
6        6
```

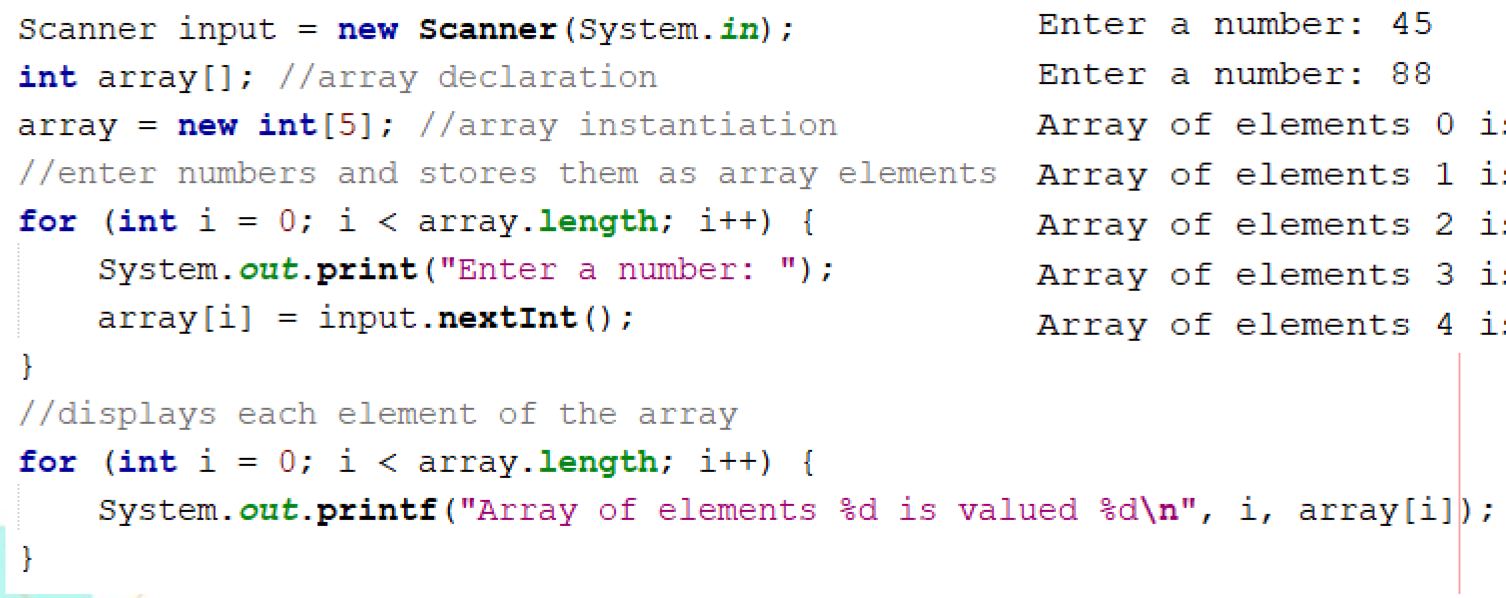


Example 3

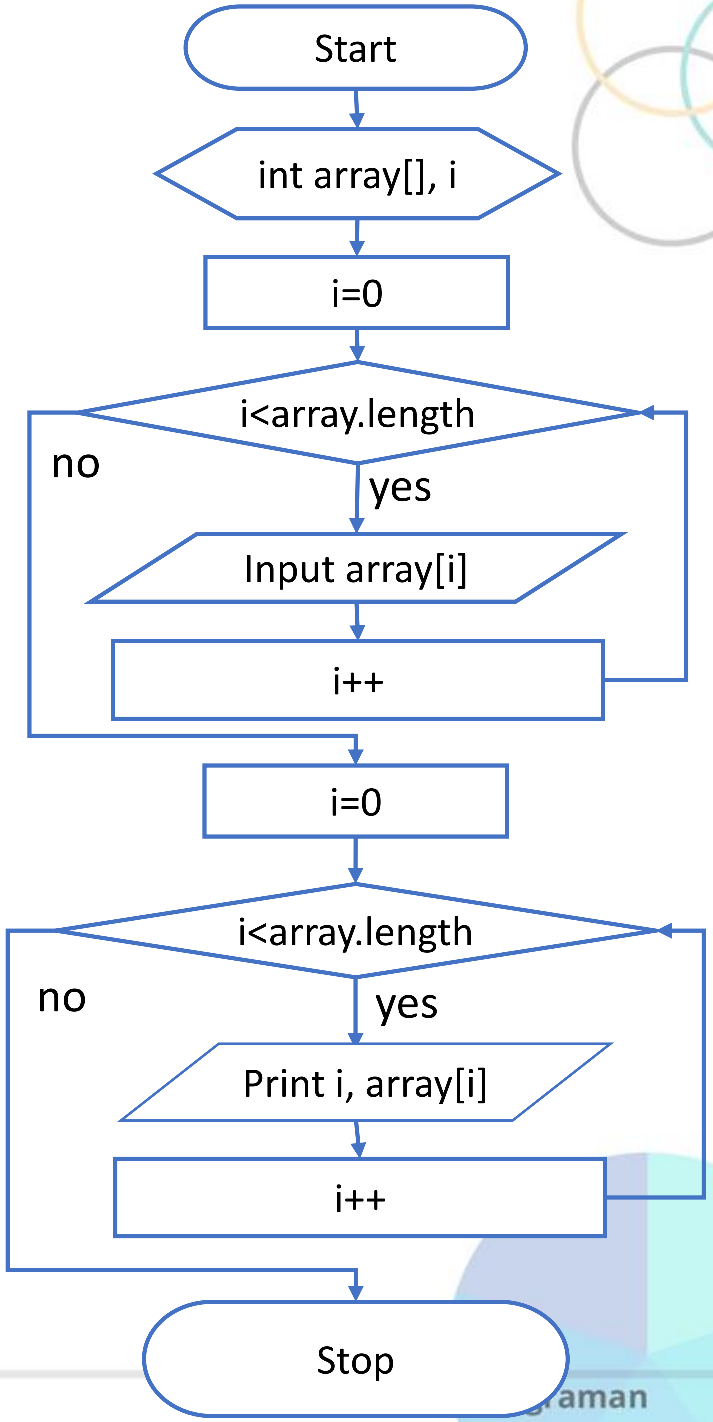
- The program asks for input of 5 numbers then displays the 5 numbers



```
Enter a number:  74
Enter a number:  12
Enter a number:  9
Enter a number:  45
Enter a number:  88
Array of elements 0 is valued 74
Array of elements 1 is valued 12
Array of elements 2 is valued 9
Array of elements 3 is valued 45
Array of elements 4 is valued 88
```



```java
Scanner input = new Scanner(System.in);
int array[]; //array declaration
array = new int[5]; //array instantiation
//enter numbers and stores them as array elements
for (int i = 0; i < array.length; i++) {
    System.out.print("Enter a number: ");
    array[i] = input.nextInt();
}
//displays each element of the array
for (int i = 0; i < array.length; i++) {
    System.out.printf("Array of elements %d is valued %d\n", i, array[i]);
}
```

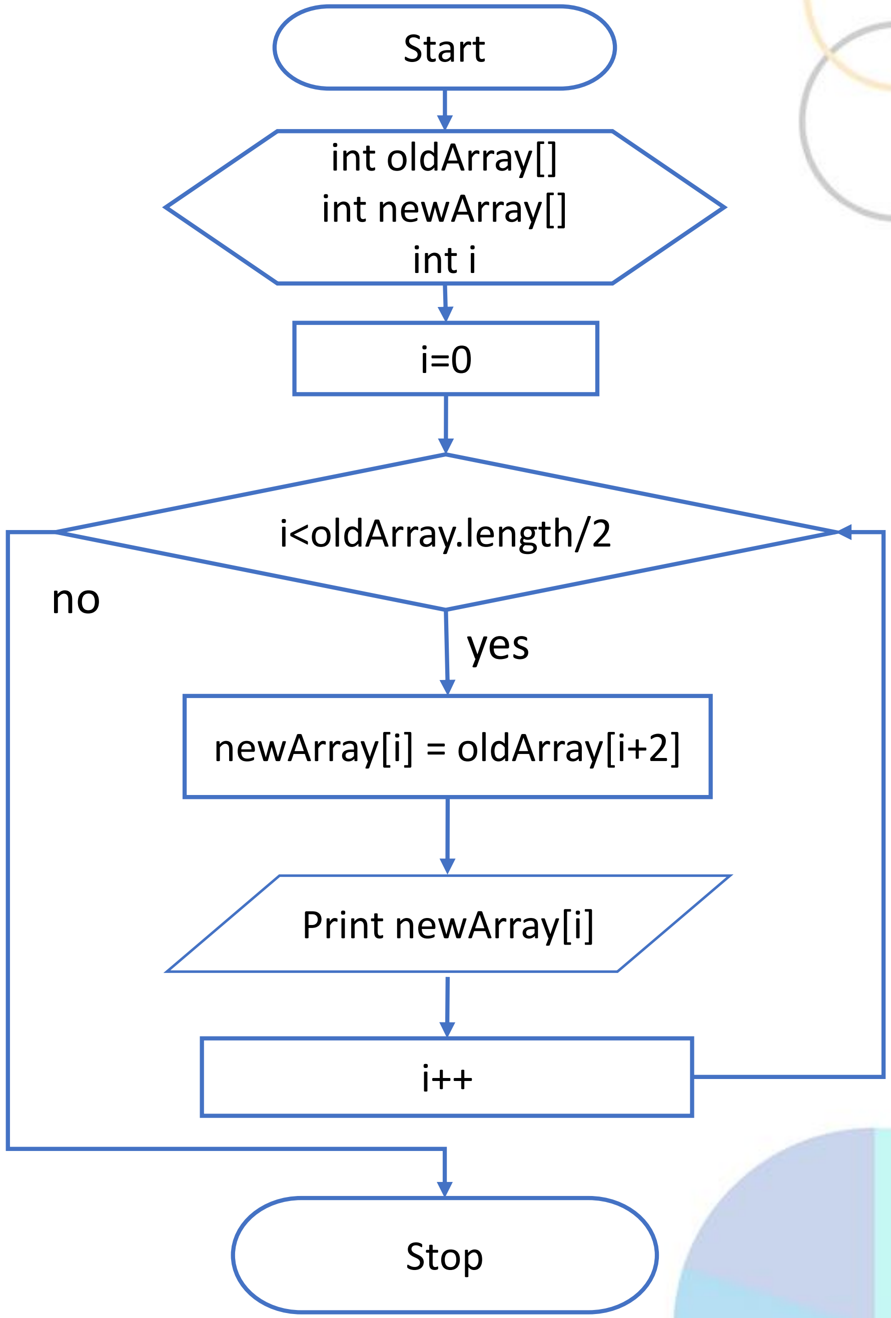


Example 4

- Make a copy of the array contents

```
int[] oldArray = {1, 3, 6, 7, 9};
int[] newArray = new int[5];
//the loop is only performed half the length of oldArray
for (int i = 0; i < oldArray.length / 2; i++) {
    newArray[i] = oldArray[i + 2];
    System.out.print(newArray[i] + " ");
}
```

```
6 7
```

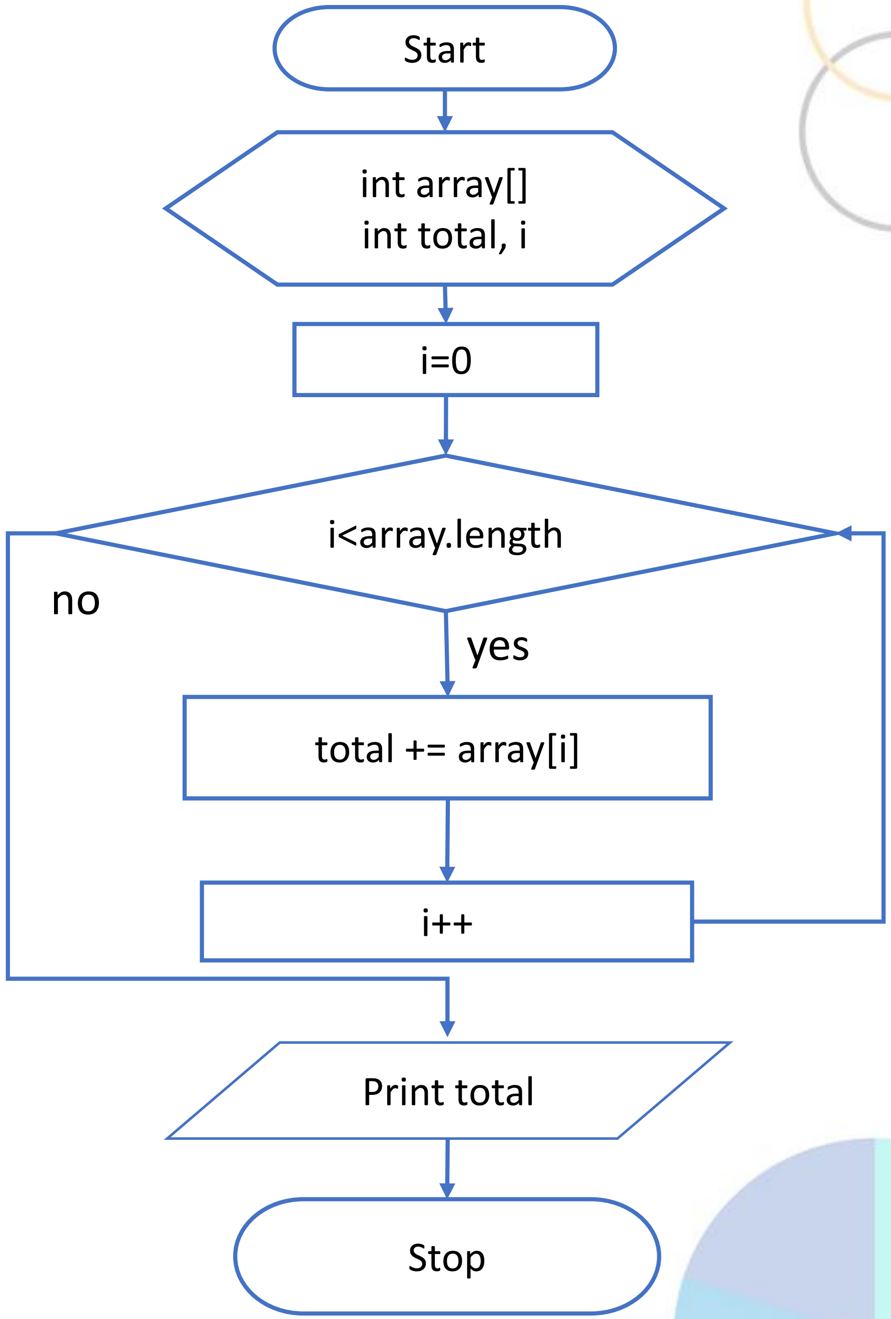


Example 5

- Array Summation

```java
int array[] = {33, 4, 5, 23, 1, 5, 6};
int total = 0;
//adds the value of each element to total
for (int i = 0; i < array.length; i++) {
    total += array[i];
}
System.out.println(total);
```

```
77
```



Searching & Sorting

Enrichment Material

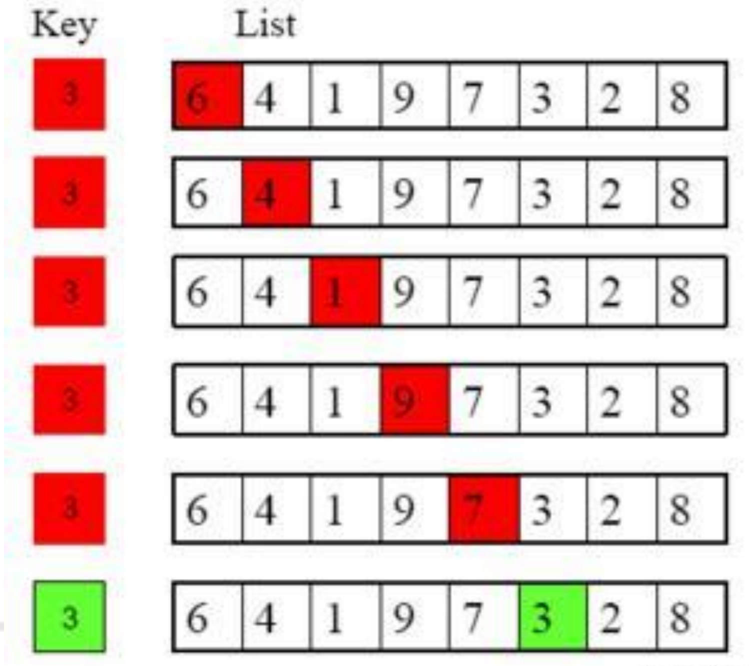
Searching

- One of the most common ways to perform array operations is searching
- Searching is performed to **find a specific value** for an element in array
- One of the easiest searching algorithms is **Linear Search**

Searching

- Suppose that in an array, you want to find the index position of an array element.
- In Linear Search, compare the "key" or number you want to find, with each element in the array.

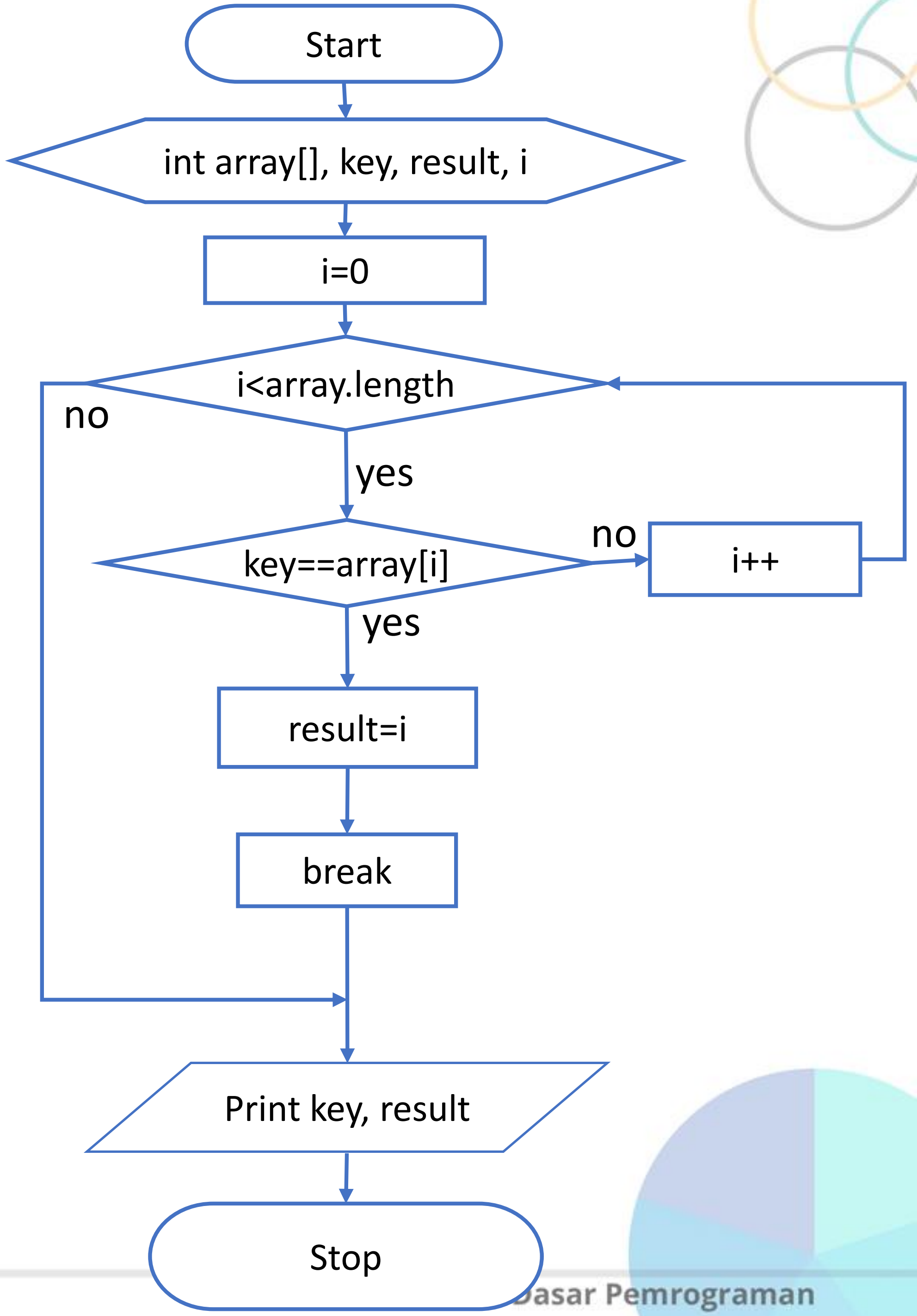
- The key you want to find is 3
- A loop is used to compare each array element
- The number 3 is in the 5th index.
- So once found, looping will stop



Searching

```
int array[] = {6, 4, 1, 9, 7, 3, 2, 8};
int key = 3;
int result = 0;
for (int i = 0; i < array.length; i++) {
    if (key == array[i]) {
        result = i;
        break;
    }
}
System.out.println("Key " + key + " is in index " + result);
```

```
Key 3 is in index 5
```

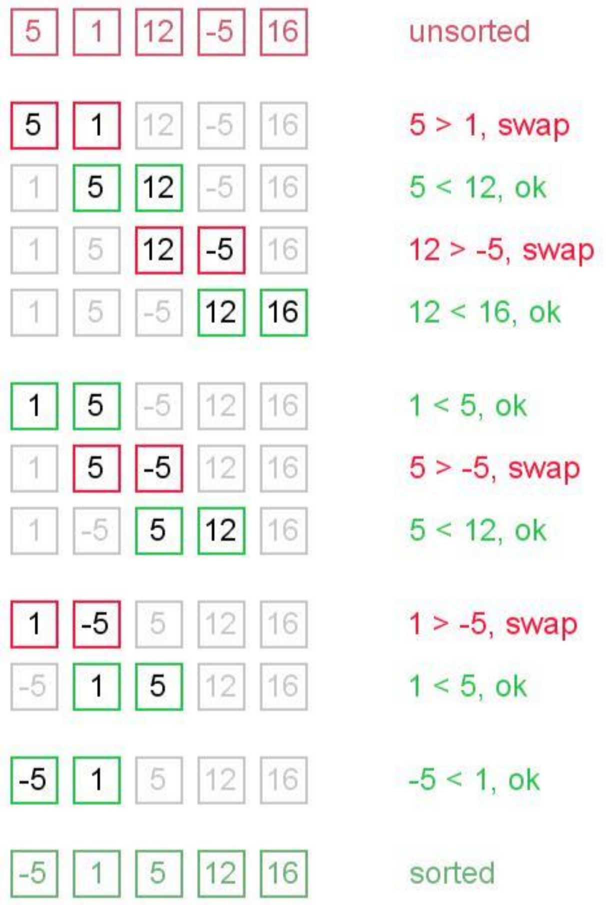


Sorting

- Sorting is the process of **sorting array elements** from smallest to largest (ascending) or vice versa (descending)
- One of the easiest sorting algorithms is **Bubble Sort**

Sorting

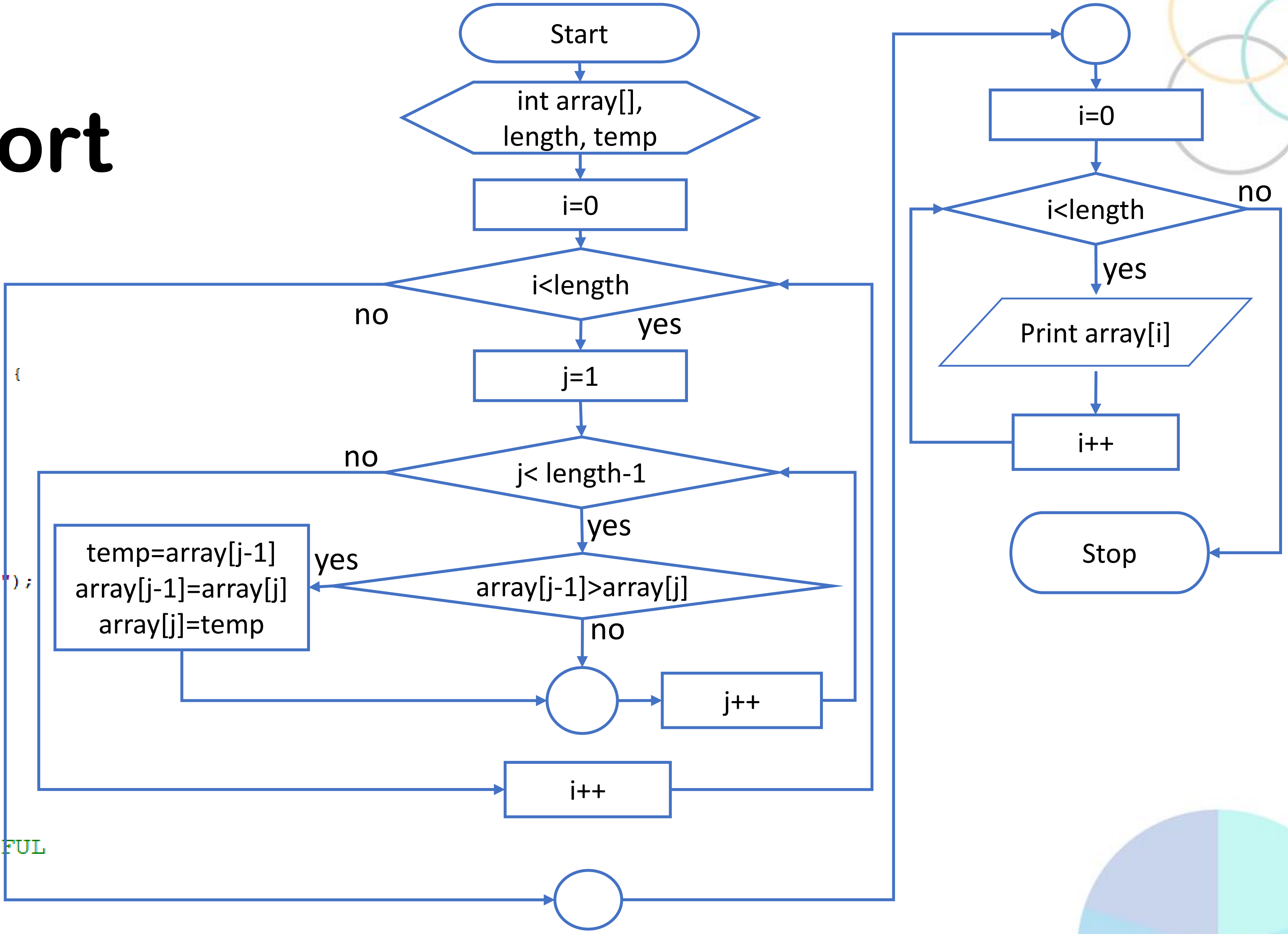
- In Bubble Sort, looping is performed from the first element to the last element of the array.
- Then each element is compared with the next element.
- If that element is bigger than the next element, it will be swapped.



Bubble Sort

```
int array[] = {6, 4, 1, 9, 7, 3, 2, 8};
int length = array.length;
int temp;
for (int i = 0; i < length; i++) {
    for (int j = 1; j < length - 1; j++) {
        if (array[j - 1] > array[j]) {
            temp = array[j - 1];
            array[j - 1] = array[j];
            array[j] = temp;
        }
    }
}
System.out.println("Result after sorting");
for (int i = 0; i < length; i++) {
    System.out.print(array[i] + " ");
}
```

```
Result after sorting
1 2 3 4 6 7 9 8 BUILD SUCCESSFUL
```



Assignment

1. Create a flowchart of variable array filling with 50 elements length using looping!
2. Create a flowchart to fill array elements with 5 elements, then display the contents of array in reverse order!