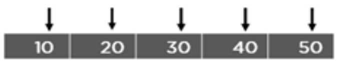
Basic Operations in Arrays

- **Traverse:** Visiting each element of an array in a specific order (e.g., sequential, reverse).
- **Insertion:** Adding a new element to an array at a specific index.
- **Deletion:** Removing an element from an array at a specific index.
- **Search**: Searches an element using the given index or by the value.
- **Update** – Updates an element at the given index.
- **Display:** Displays the contents of the array.

Traversing operation:



This operation is performed to traverse through the array elements. It prints all array elements one after another.

Algorithm

```
1 Start
2. Initialize an Array of certain size and datatype.
3. Initialize another variable 'i' with 0.
4. Print the ith value in the array and increment i.
5. Repeat Step 4 until the end of the array is reached.
6. End
```

Example

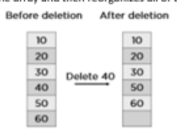
```
public class ArrayDemo {
  public static void main(String []args) {
    int LA[] = new int[5];
    System.out.println("The array elements are: ");
    for(int i = 0; i < 5; i++) {
      LA[i] = i + 2;
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
  }
}
```

Output

```
The original array elements are :
LA[0] = 1
LA[1] = 3
LA[2] = 5
LA[3] = 7
LA[4] = 8
```

Deletion operation

This operation removes an element from the array and then reorganizes all of the array elements.



Algorithm

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to delete an element available at the K^th^ position of LA.

```
1. Start
2. Set J = K
3. Repeat steps 4 and 5 while J < N
4. Set LA[J] = LA[J + 1]
5. Set J = J+1
6. Set N = N-1
7. Stop
```

Example

```
public class ArrayDemo {
  public static void main(String []args) {
    int LA[] = new int[3];
    int n = LA.length;
    System.out.println("Array Before Deletion:");
    for(int i = 0; i < n; i++) {
      LA[i] = i + 3;
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
    for(int i = 1; i<n-1; i++) {
      LA[i] = LA[i+1];
      n = n - 1;
    }
    System.out.println("Array After Deletion:");
    for(int i = 0; i < n; i++) {
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
  }
}
```

Output

```
Array Before Deletion:
LA[0] = 1
LA[1] = 3
LA[2] = 5
Array After Deletion :
LA[0] = 1
LA[1] = 5
```

Update operation

This operation is performed to update an existing array element located at the given index.

Algorithm

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to update an element available at the Kth position of LA.

```
1. Start
2. Set LA[K-1] = ITEM
3. Stop
```

Example

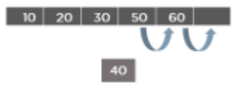
```
public class ArrayDemo {
  public static void main(String []args) {
    int LA[] = new int[5];

    int item = 15;
    System.out.println("The array elements are: ");
    for(int i = 0; i < 5; i++) {
      LA[i] = i + 2;
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
    LA[3] = item;
    System.out.println("The array elements after updation are: ");
    for(int i = 0; i < 5; i++)
      System.out.println("LA[" + i + "] = " + LA[i]);
  }
}
```

Output

```
The array elements are::
LA[0] = 1
LA[1] = 3
LA[2] = 5
LA[3] = 7
LA[4] = 8
The array elements after updation are:
LA[0] = 1
LA[1] = 3
LA[2] = 10
LA[3] = 7
LA[4] = 8
```

Insertion operation

This operation is performed to insert one or more elements into the array. As per the requirements, an element can be added at the beginning, end, or at any index of the array.



Algorithm

```
1. Start
2. Create an Array of a desired datatype and size.
3. Initialize a variable 'i' as 0.
4. Enter the element at ith index of the array.
5. Increment i by 1.
6. Repeat Steps 4 & 5 until the end of the array.
7. Stop
```

Example

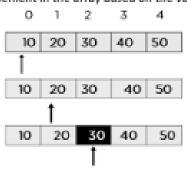
```
public class ArrayDemo {
  public static void main(String []args) {
    int LA[] = new int[3];
    System.out.println("Array Before Insertion:");
    for(int i = 0; i < 3; i++)
      System.out.println("LA[" + i + "] = " + LA[i]); //prints empty array
    System.out.println("Inserting Elements..");

    // Printing Array after Insertion
    System.out.println("Array After Insertion:");
    for(int i = 0; i < 3; i++) {
      LA[i] = i+3;
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
  }
}
```

Output

```
Array Before Insertion:
LA[0] = 0
LA[1] = 0
LA[2] = 0
Inserting elements..
Array After Insertion:
LA[0] = 2
LA[1] = 3
LA[2] = 4
LA[3] = 5
LA[4] = 6
```

Search operation

This operation is performed to search an element in the array based on the value or index.



Algorithm

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to find an element with a value of ITEM using sequential search.

```
1. Start
2. Set J = 0
3. Repeat steps 4 and 5 while J < N
4. IF LA[J] is equal ITEM THEN GOTO STEP 6
5. Set J = J +1
6. PRINT J, ITEM
7. Stop
```

Example

```
public class ArrayDemo{
  public static void main(String []args){
    int LA[] = new int[5];
    System.out.println("Array:");
    for(int i = 0; i < 5; i++) {
      LA[i] = i + 3;
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
    for(int i = 0; i < 5; i++) {
      if(LA[i] == 6)
        System.out.println("Element " + 6 + " is found at index " + i);
    }
  }
}
```

Output

```
Array:
LA[0] = 1
LA[1] = 3
LA[2] = 6
LA[3] = 7
LA[4] = 8

Element 6 is found at index 2
```

Display Operation

This operation displays all the elements in the entire array using a print statement.

Algorithm

Consider LA is a linear array with N elements. Following is the algorithm to display an array elements.

```
1. Start
2. Print all the elements in the Array
3. Stop
```

Example

```
public class ArrayDemo {
  public static void main(String []args) {
    int LA[] = new int[5];
    System.out.println("The array elements are: ");
    for(int i = 0; i < 5; i++) {
      LA[i] = i + 2;
      System.out.println("LA[" + i + "] = " + LA[i]);
    }
  }
}
```

Output

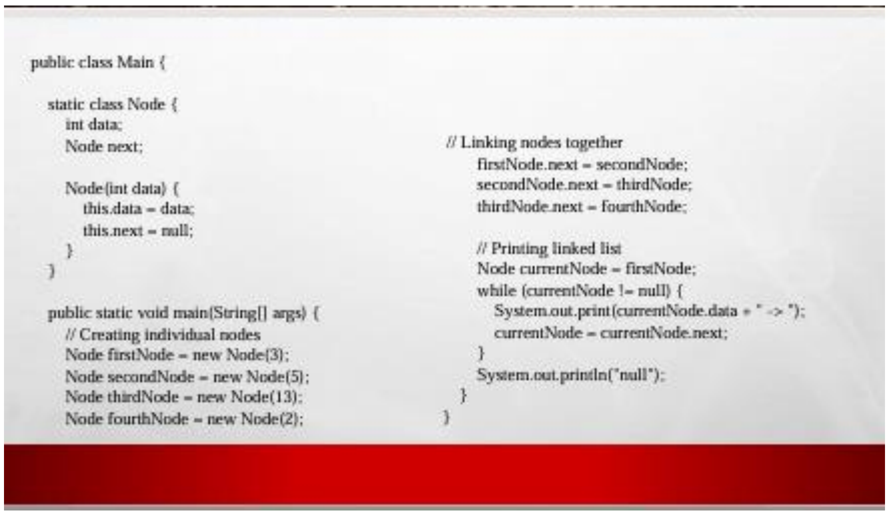
```
The array elements are:
LA[0] = 1
LA[1] = 3
LA[2] = 5
LA[3] = 7
LA[4] = 8
```

Ex. 1

```
package linkedlistsample;

import java.util.LinkedList;

public class LinkedListSample {


   public static void
main(String[] args) {
   LinkedList <String> program =
new LinkedList <>();

   program.add("BSCS");
   program.add("BSIT-Elec");
   program.add("BSM");
   System.out.println(program);


   program.add(2, "BSIT-
FoodTech");

   System.out.println(program);

   program.remove("BSIT-
FoodTech");

   System.out.println(program);

   program.set(0,"BSED-Math");

   System.out.println(program);

   }

}
```

Ex. 2

```
package linkedlista1;

import java.util.LinkedList;
import java.util.Scanner;

/**
 *
 * @author RESTIFICAR
 */
public class LinkedListA1 {

   public static void
main(String[] args) {

       Scanner scan = new
Scanner(System.in);
       LinkedList<Integer> num =
new LinkedList();

       System.out.print("\nEnter
Number of Elements: ");
       int num_elem =
scan.nextInt();
       System.out.println("\n");
       for(int i = 1; i <=
num_elem; i++){
           System.out.print("Eleme
nt ["+i+"]: ");
           int numbers =
scan.nextInt();
           num.add(numbers);
       }

       int number_size =
num.size();
       int mid_num =
number_size / 2;


       System.out.println("\nMid
dle number:
"+num.get(mid_num));

   }

}
```



```
public class Main {

  static class Node {
    int data;
    Node next;

    Node(int data) {
      this.data = data;
      this.next = null;
    }
  }

  public static void main(String[] args) {
    // Creating individual nodes
    Node firstNode = new Node(3);
    Node secondNode = new Node(5);
    Node thirdNode = new Node(13);
    Node fourthNode = new Node(2);

  // Linking nodes together
    firstNode.next = secondNode;
    secondNode.next = thirdNode;
    thirdNode.next = fourthNode;

    // Printing linked list
    Node currentNode = firstNode;
    while (currentNode != null) {
      System.out.print(currentNode.data + " -> ");
      currentNode = currentNode.next;
    }
    System.out.println("null");
  }
}
```