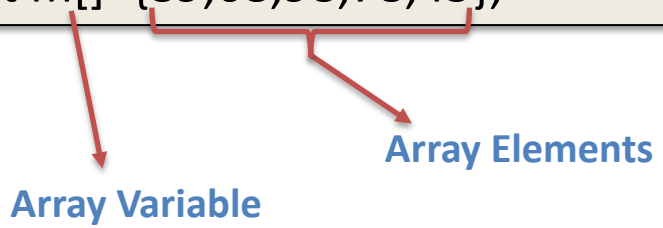
Array

- **Array** is the collection of similar type of data with contiguous memory allocation.

Without Array

int m1=89;
int m2=68;
int m3=98;
int m4=78;
int m5=45;

With Array
int m[]={89,68,98,78,45};



Steps for Array Creation

Step 1- Array Declaration

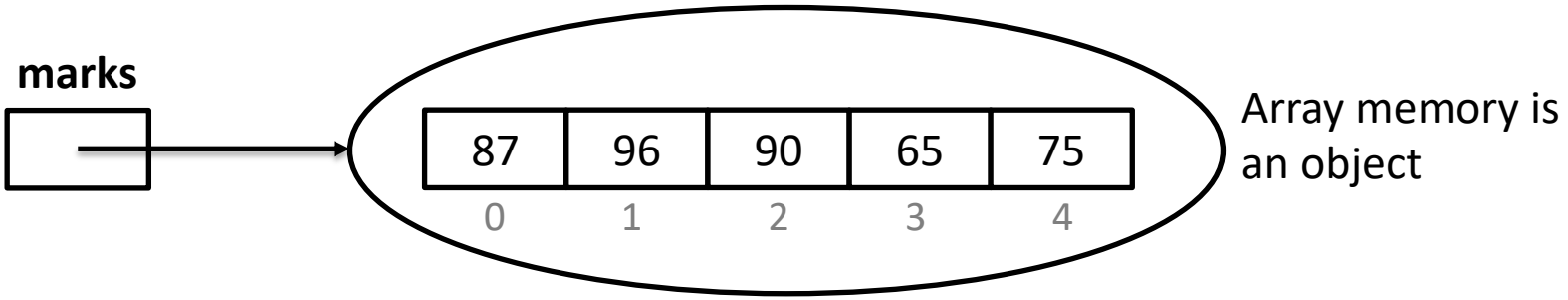
int marks[]; or **int []marks;**

Step 2- Array Memory Creation

marks=new int[5];

Step 3- Array Initialization:

```
marks[0]=87;
marks[1]=96;
marks[2]=90;
marks[3]=65;
marks[4]=75;
```



Note:

Array indexing starts from 0. Because by default array variable refers the first block of the memory. That's why to access first block, we need to add 0 to current address and to access second block, we need to add 1.

Array Creations

incapp

Array Creation:

```
int marks[]= new int[5];
marks[0]=87;
marks[1]=96;
marks[2]=90;
marks[3]=65;
marks[4]=75;
```

Array Creation:

```
int marks[]= new int[ ]{87,96,90,65,75};
```

Array Creation:

```
int marks[]= new int[5]{87,96,90,65,75};
```

Array Creation:

```
int marks[]= {87,96,90,65,75};
```

Array Creation:

```
int marks[];
marks={87,96,90,65,75};
```

Accessing an Array

We can access array using two types of loop controls:

1. Normal for loop
2. for-each loop

Normal for loop

```
int marks[]={89,68,98,78,45};
for(int i=0; i<marks.length ; i++) {
        System.out.println(marks[i]);
}
```

For-Each loop

```
int marks[]={89,68,98,78,45};
for(int x : marks) {
        System.out.println(x);
}
```

Note:
'marks.length' property return the number of elements in array.

Operations on Array

incapp

- The operations that can be performed on array are:

Traversing	Visiting every element
Insertion	Adding a new element
Deletion	Removing the existing element
Searching	Checking the existence of a particular element
Sorting	Arranging the elements in a particular order
Merging	Combining two or more arrays

Note: I will explain the insertion, traversing and deletion operation.
All the remaining operations will be explained in chapters of Searching and Sorting.

Traversing Operation on Array

- **Algo for Traversing an Array**

Let **arr** is array, **length** is the number of elements present in the array, **i** is the index for iteration and then the algorithm is given below:

1. Create array **arr.**
2. Read array **arr**.
3. Check, if length > 0, then:
    a) Repeat for i = 0 to length – 1
        Print, **arr**[i]
4. Otherwise,
    a) Print, No element to traverse.
5. Exit.

Insertion Operation on Array

- **Algo for Insertion at End of Array**

Let **arr** and **newArr** are the arrays, **length** is the number of elements present in the array, **value** will store the value to be inserted at the end of array, then the algorithm is given below:

1. Create array **arr.**
2. Read array **arr**.
3. Read **value**.
4. Create new array **newArr** with one more element.
5. Copy all elements of old array to new array **newArr[index]**=**arr[index]**
6. Set **newArr**[**length-1**] = **value**.
7. Print, new array **newArr**.
8. Exit.

Insertion Operation on Array

- **Algo for Insertion at Specified index of Array**

Let **arr** and **newArr** are the arrays, **length** is the number of elements present in the array, **value** will store the value to be inserted and **index** will be that specified index of array where the vale to be inserted, **i** is the index for iteration:

1. Create and Read array **arr**
2. Read **value**.
3. Read **index**.
4. Check if index ≥ 0 and index ≤ length – 1
   a) Create new array **newArr** with one more element.
   b) Copy all elements of old array to new array **newArr[index]**=**arr[index]**
   c) Shift all elements right of **index** element by one position .
      Repeat for i = length-1 to index
      - newArr[i] = newArr[i - 1];
   a) inserting **value** at position **index :** newArr[index] = value
   b) Print, array **newArr**.
5. Otherwise,
   a) Print, invalid index.
6. Exit.

Deletion Operation on Array

- **Algo for Deletion from End of Array**

Let **arr** and **newArr** are the arrays, **length** is the number of elements present in the array, then the algorithm is given below:

1. Create array **arr.**
2. Read array **arr**.
3. Create new array **newArr** with one less element.
4. Copy all elements of old array to new array **newArr[index]**=**arr[index]**
5. Print, new array **newArr**.
6. Exit.

Deletion Operation on Array

- **Algo for Deletion from Specified index of Array**

Let **arr** and **newArr** are the arrays, **length** is the number of elements present in the array, and **index** will be that specified index of array where the vale to be deleted, **i** is the index for iteration:

1. Create array **arr.**
2. Read array **arr** and **index**.
3. Check if index ≥ 0 and index ≤ length – 1
   a) Repeat **for** i = index to length – 1
      Set **arr**[i] = **arr**[i+1]
   b) Create new array **newArr** with one less element.
   c) Copy all elements of old array to new array **newArr[index]**=**arr[index]**
   d) Print, new array **newArr**.
4. Otherwise,
   a) Print, invalid index.
5. Exit.