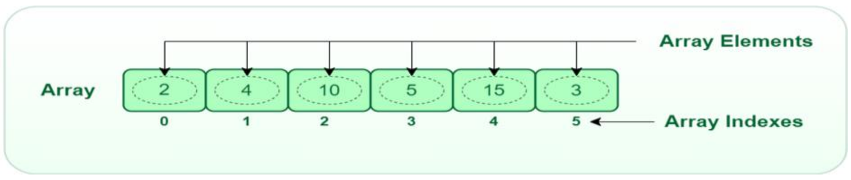
# LINEAR ARRAYS

- Linear Arrays also known as one dimensional array, is a collection of same data type elements stored in contiguous memory location.
- It allows efficient access and modification of elements using their position.
- Linear Arrays are fundamental in programming for organizing and manipulating data sequences.

## Definition

A linear array is a collection of elements of the same data type stored in contiguous memory locations, where each element is accessed using its index.



EXAMPLE:

arr= [2,4,10,5,15,3]

## Declaration Of Array

- Arrays can be declared in different ways in different languages.
- For better illustration below are some language specific array declarations

IN C language

```
 int arr[5];          // This array will store integer type element
char arr[10];         // This array will store char type element
float arr[20];        // This array will store float type element
```

In python

```
arr1 = [10, 20, 30]           # This array will store integer
arr2 = ['c', 'd', 'e']        # This array will store   characters
arr3 = [28.5, 36.5, 40.2]     # This array will store floating elements
```

Creating a Linear Array in python

- ► You can create an array by initializing a list with elements inside square brackets
- ► Example: array = [1, 2, 3, 4, 5]      creates a simple integer array.
- ► Lists can hold different data types, making them versatile for various applications.
- ► c = [1, 'hello', 3.14, True]                          # Mixed data types

<u>Program</u>

```
a = [1, 2, 3, 4, 5]                                   # List of integers
b = ['apple', 'banana', 'cherry']                     # List of strings
c = [1, 'hello', 3.14, True]                          # Mixed data types
print(a)
print(b)
print(c)
```

Types of Arrays

- ► Linear arrays are classified based on how elements are stored and accessed.

1. *Static Arrays*
2. D*ynamic Arrays*

## <u>Static Arrays</u>

- Size is fixed at compile time
- Memory is allocated when the program starts
- In this type of array, memory is allocated at compile time having a fixed size of it.
- We cannot alter or update the size of this array

EXAMPLE:   numbers = [10, 20, 30, 40, 50]

## <u>Dynamic Arrays</u>

- size can be changed during runtime

- **Memory is allocated and relocated as needed**

**In this type of array, memory is allocated at run time but not having a fixed size of it.**

```
 my_list = [1, 2, 3, "hello", 5.5]

   print(my_list)                              # Output: [1, 2, 3, 'hello', 5.5]
```

**append() method adds an element to the end of the list**

```
   my_list = [1, 2, 3]

  my_list.append(4)

 my_list.append("world")

  print(my_list)                                            # Output: [1, 2, 3, 4, 'world']
```

**insert() method inserts an element at a specific index.**

```
 my_list = [1, 2, 3]

my_list.insert(1, "new")                      # Insert "new" at index 1

print(my_list)                               # Output: [1, 'new', 2, 3]
```

## Creating a Linear Array (List)

A linear array (list) can be created in Python using square brackets[] and separating the elements with commas.

```
numbers = [10, 20, 30, 40, 50]
```

**Accessing Elements:**

Elements in a list can be accessed using their index,which starts at 0.

```
first_element = numbers[0]                  # first_element will be 10

third_element = numbers[2]                 # third_element will be 30

 last_element = numbers[-1]                # last_element will be 50
```

### Modifying Elements

Elements in a list can be modified by assigning a new value to the element at a specific index.

For eg: numbers = [10, 20, 30, 40, 50]

```
numbers[1] = 25                # numbers will now be [10, 25, 30, 40, 50]
```

## Representation of Linear Arrays in Memory

In memory, the elements are stored contiguously. If the array starts at memory address 1000 and each integer takes 4 bytes, the memory layout would be:

```
+--------+--------+--------+--------+--------+
| 10   | 20   | 30   | 40   | 50   |
+--------+--------+--------+--------+--------+
 1000    1004    1008    1012    1016     (Memory Address)
```

## Traversing Arrays in Python

In Python, arrays are usually represented using lists.Traversing an array means visiting each element one by one to perform some operation (printing, searching, calculations, etc.).

## Traversing using a for loop

```
A = [10, 20, 30, 40, 50]
for element in A:
    print(element)
```

## Searching

- Searching is the process of finding an element in a data collection
- Commonly performed on arrays, lists, and strings
- Objective: Check whether an element exists and find its position

## Linear Search

Linear search, also known as sequential search, is the simplest searching algorithm. It involves iterating through each element of a list or array until the target element is found or the end of the list is reached.

**Algorithm:**

- **Start at the beginning of the list.**
- **Compare each element with the target element.**

- **If a match is found, return the index of the element.**
- **If the end of the list is reached without finding a match, return -1 (or a similar indicator that the element is not found).**

**PROGRAM ON LINEAR SEARCH**

```
def linear_search(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1
arr = [10, 23, 45, 70, 11, 15]
x = 70
res = linear_search(arr, x)
if res != -1:
    print("Element found at index:", res)
else:
    print("Element not found in the array")
```

---

**Time Complexity of Linear Search**

- **Best Case: O(1) - The target element is found at the beginning of the list.**
- **Average Case: O(n) - On average, we need to examine half of the list.**
- **Worst Case: O(n) - The target element is at the end of the list or not present.**

**Advantages:**

- Simple to implement.
- Works on unsorted lists.

**<u>Disadvantages:</u>**

- Inefficient for large lists.