Arrays in C

Introduction

What is an Array?

An **array** is a **collection of elements of the same data type** stored in **contiguous memory locations**.

Memory storage

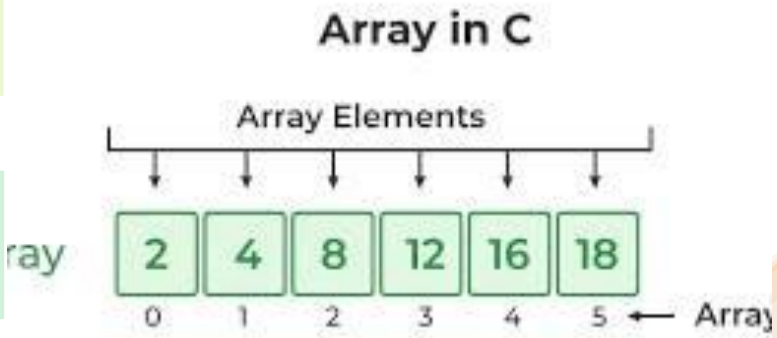
Elements are stored in contiguous memory locations

Element Access

Accessed using an index (starting from 0).

Example

Row of lockers storing numbers.



Need for Arrays

- To store multiple values under a single variable.
- Simplifies code (instead of multiple variables).
- Allows iteration using loops.
- Efficient for data manipulation.

Array Syntax

Syntax of Array

```
data_type array_name[size];
```

`data_type` → type of elements (int, float, char, etc.)

`array_name` → name of the array

`size` → number of elements

Example:

int numbers[5];

int → data type

numbers → array name

5 → size (number of elements)

Declaring and Initializing Arrays

1. DECLARATION ONLY: INT ARR[5];
2. DECLARATION WITH INITIALIZATION: INT ARR[5] = {10,20,30,40,50};
3. WITHOUT SPECIFYING SIZE: INT ARR[] = {10,20,30,40};
4. PARTIAL INITIALIZATION: INT ARR[5] = {1,2}; → {1,2,0,0,0}

Memory Representation of 1D Array

Example: int arr[5] = {10, 20, 30, 40, 50};

Index: 0 1 2 3 4

Value: 10 20 30 40 50

Elements are stored in contiguous blocks.

Accessing Array Elements

Use index starting from 0.

Example: printf("First element = %d", arr[0]);

arr[0] → first element

arr[2] → third element

Input & Output of Arrays

```
#include <stdio.h>
int main() {
    int arr[5];
    for(int i=0; i<5; i++) scanf("%d", &arr[i]);
    for(int i=0; i<5; i++) printf("%d ", arr[i]);
}
```

Types of Arrays

One-Dimensional Array: int arr[5];

Two-Dimensional Array: int matrix[2][2] = {{1,2},{3,4}};

Memory Representation: [0][0]=1, [0][1]=2, [1][0]=3, [1][1]=4

Multi-Dimensional Arrays: int arr[3][4][5];

Example – Sum of Array Elements

```
#include <stdio.h>
int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    int sum=0; for(int i=0;i<5;i++)
    sum+=arr[i];
    printf("Sum = %d", sum);
}
```

Advantages of Arrays

- Easy to store multiple values.
- Random access using index.
- Simplifies code with loops.
- Contiguous memory allocation.

Limitations of Arrays

- Fixed size (cannot grow/shrink).
- Same data type only.
- Insertion/Deletion requires shifting.
- No built-in bounds checking.

Real-Life Example

STORING MARKS OF 50 STUDENTS IN AN ARRAY.

EXAMPLE: INT MARKS[50];

INSTEAD OF DECLARING 50 SEPARATE VARIABLES, WE USE ONE ARRAY.

Summary

- Arrays store multiple values of same type.
- Access using index (0-based).
- Can be 1D, 2D, or multi-dimensional.
- Advantages: Easy access, compact code.
- Limitations: Fixed size, insert/delete.