✅ UNIT – III : Arrays and Strings in C

PART – A : ARRAYS

---

1. Arrays – Introduction

Definition

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

Why Arrays?

- Store multiple values using one variable
- Easy data processing
- Efficient memory usage

---

2. Declaring and Referencing Arrays

Syntax

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

Example

```
int marks[5];
```

Initialization

```
int marks[5] = {10, 20, 30, 40, 50};
```

Accessing Array Elements

```
marks[0];   // First element
marks[4];   // Last element
```

---

3. Array Subscripts

Meaning

Index value used to access array elements.

Important Points

- Index starts from **0**
- Last index = size – 1

Example

```
int a[3] = {10, 20, 30};
printf("%d", a[1]);  // Output: 20
```

---

4. Using for Loop for Sequential Access

Example

```
int i, a[5];

for(i = 0; i < 5; i++) {
   scanf("%d", &a[i]);
}

for(i = 0; i < 5; i++) {
   printf("%d ", a[i]);
}
```

Use

- Reading multiple values
- Printing elements

- Processing data

---

5. Using Array Elements as Function Arguments

Example

```
void display(int arr[]) {
  for(int i = 0; i < 5; i++)
    printf("%d ", arr[i]);
}

int main() {
  int a[5] = {1,2,3,4,5};
  display(a);
}
```

---

6. Array Arguments

Explanation

When an array is passed to a function, **only the base address is passed**.

Syntax

```
void functionName(int arr[], int size)
```

---

7. Searching an Array

Linear Search

Searches element one by one.

Algorithm

1. Start from index 0
2. Compare each element
3. Stop if match found

Example

```
int search(int a[], int n, int key) {
  for(int i=0; i<n; i++) {
    if(a[i] == key)
      return i;
  }
  return -1;
}
```

---

8. Sorting an Array

Bubble Sort Example

```
for(int i=0; i<n-1; i++) {
  for(int j=0; j<n-i-1; j++) {
    if(a[j] > a[j+1]) {
      int temp = a[j];
      a[j] = a[j+1];
      a[j+1] = temp;
    }
  }
}
```

---

9. Parallel Arrays
Definition
Two or more arrays with **related data stored at the same index**.
Example
int roll[3] = {1,2,3};
int marks[3] = {80,90,85};
Usage
printf("%d %d", roll[i], marks[i]);

---

10. Enumerated Types (enum)
Definition
enum assigns names to integer constants.
Syntax
enum days {SUN, MON, TUE, WED};
Example
enum days d;
d = MON;

---

11. Multidimensional Arrays
Definition
Arrays having more than one dimension.
Syntax
int a[2][3];
Example
int a[2][2] = {{1,2},{3,4}};
Accessing Elements
a[0][1]; // Output: 2

Text Diagram (2D Array)
| 1 | 2 |
| 3 | 4 |

✅ **PART – B : STRINGS**

---

12. String Basics
Definition
A string is an array of characters ending with **'\0' (null character)**.
Declaration
char name[10];
Initialization
char name[] = "C Language";

---

13. String Library Functions
Header File
#include <string.h>

Function	Use
strlen()	Find length
strcpy()	Copy string
strcat()	Join strings
strcmp()	Compare strings

---

14. String Assignment and Substrings
String Copy
strcpy(dest, src);
Substring Example
strncpy(dest, src, n);

---

15. Longer Strings – Concatenation
Example
char s1[20] = "Hello";
char s2[] = "World";
strcat(s1, s2);

---

16. Whole-Line Input
Using gets() (Not Safe)
gets(str);
Using fgets() (Safe)
fgets(str, 50, stdin);

---

17. String Comparison
strcmp()
if(strcmp(s1, s2) == 0)
  printf("Equal");

---

18. Arrays of Pointers
Definition
An array that stores **addresses of strings**.
Example
char *names[] = {"Ram", "Sita", "Ravi"};
Access
printf("%s", names[0]);

Text Diagram
names[0] → "Ram"
names[1] → "Sita"
names[2] → "Ravi"

Important Keywords (For Exams)

- Array
- Index
- Linear search
- Bubble sort

- Pointer
- String manipulation
- Enumeration
- Multidimensional array