Unit 5: Array, Pointer and String

5.1 Concept of Array

◇ Definition:

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

◇ Why Array?

- Store multiple values in one variable
- Easy to process large data
- Access using index

◇ Example:

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

Here, `marks` can store 5 integers.

5.2 Array Declaration, Access and Initialization

◇ Declaration

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

Example:

```
int arr[5];
```

◇ Initialization

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

OR

```
int arr[] = {10, 20, 30};
```

◇ Accessing Array Elements

Array index starts from **0**

```
printf("%d", arr[0]);   // First element
```

Program Example

```
#include <stdio.h>

int main()
{
    int arr[3] = {5, 10, 15};
    int i;

    for(i = 0; i < 3; i++)
    {
        printf("%d\n", arr[i]);
    }

    return 0;
}
```

5.3 Multi-Dimensional Array

Array with more than one dimension.

2D Array Declaration

```
int matrix[2][3];
```

Initialization

```
int matrix[2][2] = {
    {1, 2},
    {3, 4}
};
```

Example Program

```
#include <stdio.h>

int main()
{
    int i, j;
    int mat[2][2] = {{1,2},{3,4}};

    for(i = 0; i < 2; i++)
    {
        for(j = 0; j < 2; j++)
        {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

5.4 Concept of Pointer

Definition:

A **pointer** is a variable that stores the address of another variable.

◇ Example:

```
int a = 10;
int *p = &a;
```

5.5 Pointer Address, Dereference, Declaration, Assignment, Initialization

◇ Declaration

```
int *ptr;
```

◇ Initialization

```
int a = 5;
int *ptr = &a;
```

◇ Address Operator (&)

Gets address of variable.

```
&variable
```

◇ Dereference Operator (*)

Access value stored at address.

```
*ptr
```

Example Program

```
#include <stdio.h>

int main()
{
    int a = 10;
    int *p = &a;

    printf("Value of a = %d\n", a);
    printf("Address of a = %p\n", &a);
    printf("Pointer value = %p\n", p);
    printf("Value using pointer = %d\n", *p);

    return 0;
}
```

5.6 Pointer Arithmetic

We can perform arithmetic operations on pointers.

Allowed operations:

- Increment (++ / --)
- Addition (+)
- Subtraction (-)

Example:

```
#include <stdio.h>

int main()
{
    int arr[3] = {10, 20, 30};
    int *p = arr;

    printf("%d\n", *p);      // 10
    p++;
    printf("%d\n", *p);      // 20
```

```
    return 0;
}
```

Pointer moves according to data type size.

5.7 Array and Pointer

Array name acts as a pointer to the first element.

```
int arr[3] = {10,20,30};
```

- `arr` → address of first element
- `arr[0]` = `*(arr + 0)`
- `arr[1]` = `*(arr + 1)`

Example

```
#include <stdio.h>

int main()
{
    int arr[3] = {1,2,3};
    int *p = arr;
    int i;

    for(i = 0; i < 3; i++)
    {
        printf("%d ", *(p + i));
    }

    return 0;
}
```

5.8 String

In C, a string is a character array ending with `\0` (null character).

```
char str[6] = "Hello";
```

Memory:
H e l l o \0

◇ Reading String

```
scanf("%s", str);
```

OR

```
fgets(str, sizeof(str), stdin);
```

Example

```
#include <stdio.h>

int main()
{
    char name[20];

    printf("Enter name: ");
    scanf("%s", name);

    printf("Name is %s", name);

    return 0;
}
```

5.9 String Functions in C

```
#include <string.h>
```

Function	Purpose
strlen()	Length of string
strcpy()	Copy string
strcat()	Concatenate
strcmp()	Compare strings

Example

```
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[20] = "Hello";
    char str2[20] = "World";

    printf("Length = %lu\n", strlen(str1));
    strcpy(str2, str1);
    printf("Copied string = %s\n", str2);
    strcat(str1, " C");
    printf("Concatenated = %s\n", str1);
    printf("Compare = %d\n", strcmp(str1, str2));

    return 0;
}
```

5.10 Pointer and String

String can be accessed using pointer.

```
char str[] = "Hello";
char *p = str;
```

Example Program

```
#include <stdio.h>

int main()
{
    char str[] = "Hello";
    char *p = str;

    while(*p != '\0')
    {
        printf("%c ", *p);
        p++;
    }

    return 0;
}
```

Summary

Topic	Key Point
Array	Collection of same data type
2D Array	Matrix form
Pointer	Stores address
&	Address operator
*	Dereference operator
Pointer arithmetic	Moves by data size
String	Character array + \0
String functions	strlen, strcpy, strcat, strcmp
Array & Pointer	Array name = base address