Arrays in C

Declaration & Initialization

-
- **One-Dimensional Array**
-
- **Declaration**: int arr[5];
- **Initialization**: int arr[5] = {1, 2, 3, 4, 5};
-
- **Two-Dimensional Array (Matrix)**
-
- **Declaration**: int matrix[3][3];
- **Initialization**:
- c
- CopyEdit
- int matrix[2][2] = {
- {1, 2},
- {3, 4}
- };
-
-

Strings in C

- A **string** is an array of characters ending with a null (\0) character.
- **Declaration**: char str[10];
- **Initialization**: char str[] = "Hello";

String Operations (Using <string.h>)

- strcpy(dest, src); → Copy string
- strlen(str); → Find length of string
- strcat(str1, str2); → Concatenate strings
- strcmp(str1, str2); → Compare two strings
- strrev(str); → Reverse a string (not in standard library, but available in some compilers)

String Array (Array of Strings)

```
c
CopyEdit
char names[3][10] = {"Alice", "Bob", "Charlie"};
```

Sorting & Searching Algorithms

-
- **Sorting Algorithms** (Used to arrange elements in order):
- **Bubble Sort** → Repeatedly swaps adjacent elements if they are in the wrong order.
- **Selection Sort** → Finds the smallest element and places it at the correct position.
- **Insertion Sort** → Inserts elements in the correct order one by one.
-
- **Searching Algorithms**:
- **Linear Search** → Checks elements one by one.
- **Binary Search** → Efficient method (works on sorted arrays) by dividing the search space.

•

Matrix Operations

1st **Addition**
2nd**Subtraction**
3rd**Multiplication**
Example **Matrix Multiplication Logic**:

- Multiply rows of the first matrix with columns of the second matrix and sum up the values.

Functions in C

Definition & Declaration

- **Function Definition**:
- c
- CopyEdit
- int add(int a, int b) {
- return a + b;
- }
-
- **Function Declaration (Prototype)**:
- c
- CopyEdit
- int add(int, int);
-
- **Function Call**:
- c
- CopyEdit
- result = add(5, 10);
-

Passing Arguments

- **Pass by Value** → Copies values to function parameters (changes do not affect original variables).
- **Pass by Reference** → Uses pointers to modify the actual variables.

Example **Pass by Value**:

```
c
CopyEdit
void modify(int x) {
    x = 10;
}
```

Example **Pass by Reference** (Using Pointers):

```
c
CopyEdit
void modify(int *x) {
    *x = 10;
}
```

Recursion in C

- **Recursion** → A function calling itself until a base condition is met.
- Example: **Factorial Calculation**
- c

```
CopyEdit
int factorial(int n) {
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}
```

This **C Programming Cheat Sheet** covers **arrays, strings, sorting/searching, matrix operations, functions, and recursion**. Let me know if you need more details!