STRUCTURES, FUNCTIONS AND POINTERS

Presented by Haaniya Asiya

STRUCTURES, FUNCTIONS & POINTERS (IN DSA)

1. Structures

A **structure** is a *user-defined data type* in C/C++ that groups variables of different data types under one name.

Properties:

- Heterogeneous (different types of data in one unit).
- Accessed using `.` (dot operator).
- Useful for real-world modeling (e.g., Student, Employee).

2. Functions

A **function** is a block of reusable code that performs a specific task.

Types:

- **Library Functions** (e.g., `printf`, `scanf`).
- **User-defined Functions** (e.g., `int add(int a, int b)`).

Properties:

- Improves modularity & reusability.
- Can return values or be void.
- Helps reduce code redundancy.

3. Pointers

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

Types:

- Null Pointer
- Void Pointer
- Wild Pointer
- Dangling Pointer
- Function Pointer

Properties:

- Enable dynamic memory allocation (`malloc`, `free`).
- Essential for linked lists, trees, and graphs in DSA.
- Operators: `&` (address-of), `*` (dereference).

STRUCTURES

- **Definition:** A *structure* in C is a user-defined data type that groups different data items together.
- **Why use it?** Helps organize complex data under one name.
- **How it works:** Combines multiple variables (of different types) into a single unit.

```c
struct Student {
    char name[50];
    int age;
    float marks;
};
```

FUNCTIONS

- **Definition:** A *function* is a block of code designed to perform a specific task.
- **Why use it?** Reusability, easier debugging, modular programming.
- **How it works:** Call the function whenever the task is needed → reduces repetition.

```c
int add(int a, int b) {
    return a + b;
}
```

POINTERS

- **Definition:** A *pointer* is a variable that stores the memory address of another variable.
- **Why use it?** Efficient memory handling, dynamic memory allocation, helps in linked lists & trees.
- **How it works:** `ptr` points to the address of `x`, so `*ptr` gives the value.

```c
int x = 10;
int *ptr = &x;
```