23-12-25

POINTERS & DYNAMIC MEMORY ALLOCATION

1. Pointers

A. Definition

- A pointer is a variable that stores the address of another variable.
- It does **not store the actual value**, it stores the **memory address** of the value.

B. Why pointers are used

- To access memory directly
- To modify variables using their address
- Required for dynamic memory allocation
- Used in arrays, functions, structures

C. Basic Terms

- Address-of operator (&) → gets address
- Dereference operator (*) → gets value from address

Example:

```
int a = 10;

int *ptr = &a;
```

- ptr → address of a
- *ptr → value of a (10)

Reference & Dereference

- Reference → holding address (ptr = &a)
- Dereference → accessing value (*ptr)

Important Point

- ptr = address
- *ptr = value

2. Types of Pointers

A. Typed Pointer

- Pointer data type **must match** variable data type
- **Example:**

```
int a = 10;
int *ptr = &a;   // correct
```

B. Void Pointer (Generic Pointer)

- Written as void *
- Can store **address of any data type**
- **Cannot be dereferenced directly**
- Must be **typecast before dereferencing**

Example

```
int a = 10;
void *ptr = &a;
printf("%d", *(int *)ptr);
```

3. Memory Types

A. Primary Memory

- Temporary
- RAM
- Fast access

B. Secondary Memory

- Permanent
- Hard disk, SSD
- Slower

4. Memory Allocation

Two Types

1. **Static Memory Allocation (SMA)**
2. **Dynamic Memory Allocation (DMA)**

5. Static Memory Allocation (SMA)

Features

- Memory size is fixed
- Allocated at compile time
- Cannot be resized
- Size depends on data type

Example:

```
int a;
float b;
```

6. Dynamic Memory Allocation (DMA)

A. Definition

Dynamic Memory Allocation allows us to **allocate memory at runtime** using predefined functions.

B. Advantages

- Memory is allocated **during runtime**
- Memory size **can be resized**
- Efficient memory usage

C. DMA Functions

Function	Purpose
malloc()	Allocate memory
calloc()	Allocate + initialize
realloc()	Resize memory
free()	Deallocate memory

7. malloc()

Syntax

ptr = (datatype *) malloc(size);

Features

- Allocates **single block**

- Memory is **uninitialized**
- Returns void *

Example:

int *ptr = (int *)malloc(sizeof(int));

8. calloc()

Syntax

ptr = (datatype *) calloc(n, size);

Features

- Allocates **multiple blocks**
- Memory is **initialized to zero**

Example:

int *ptr = (int *)calloc(5, sizeof(int));