Array int → index

LECTURE - 43

[DYNAMIC MEMORY ALLOCATION IN C]

Sensitive → [Memory]

Dynamic Memory Allocation

- Changable
- Storage
- Provide (allot)

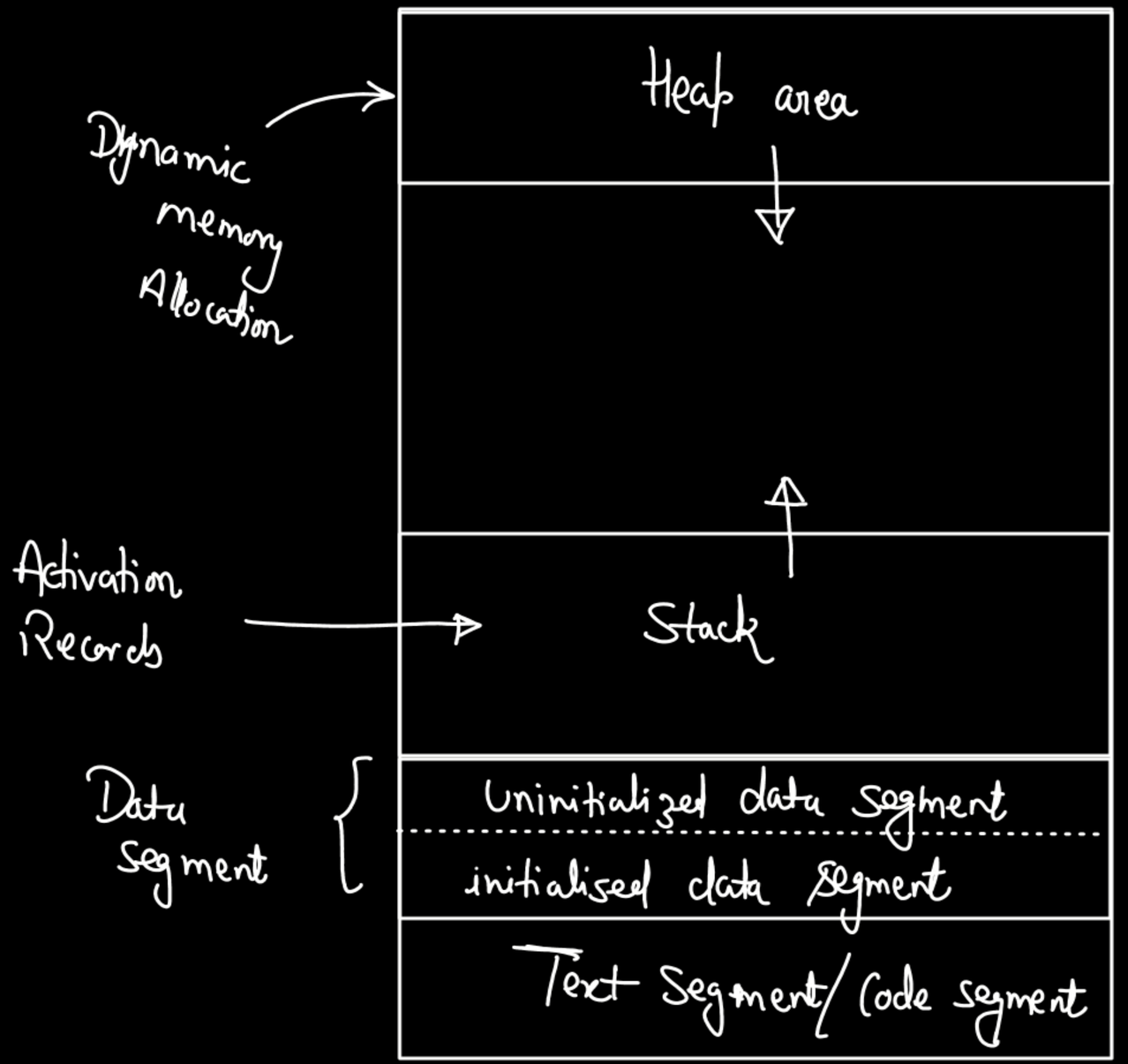
⇒ DMA refers to the process of allocating memory during program execution, as opposed to allocation it at Compile time.

Runtime

- `int x = 10;` → Compile → x ← 4 Byte reserve
- `int arr[10];` → Compile → arr ← 40 Byte reserve

fn Call ← Stack grow

DMA ← Heap grow



→ Compiler

→ Request (Memory)

↓

medium → function

↳ 32 Bytes of memory

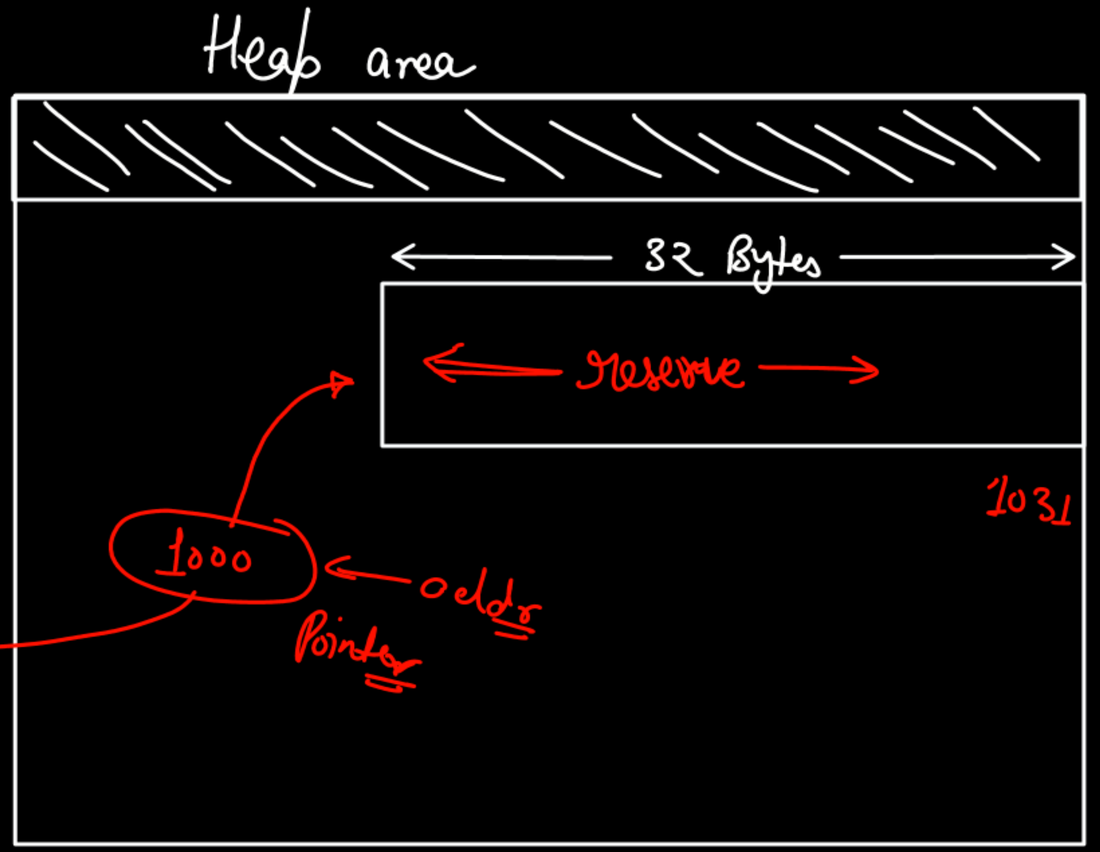
↓

Return ← address of first block

→ (Address 1000 के बाद के 32 blocks आयेंगे)

↓

(Cells)



Responsibility

जिम्मेदारी ← free up to reserved space after program execution / finishing the work.

↓

[User]

malloc ( ) :← memory allocate

↳ Allocates a block of memory of the specified size in bytes.

→ Returns a (void*) pointer to the beginning of the allocated block.

→ It returns NULL if their is an insufficient space.

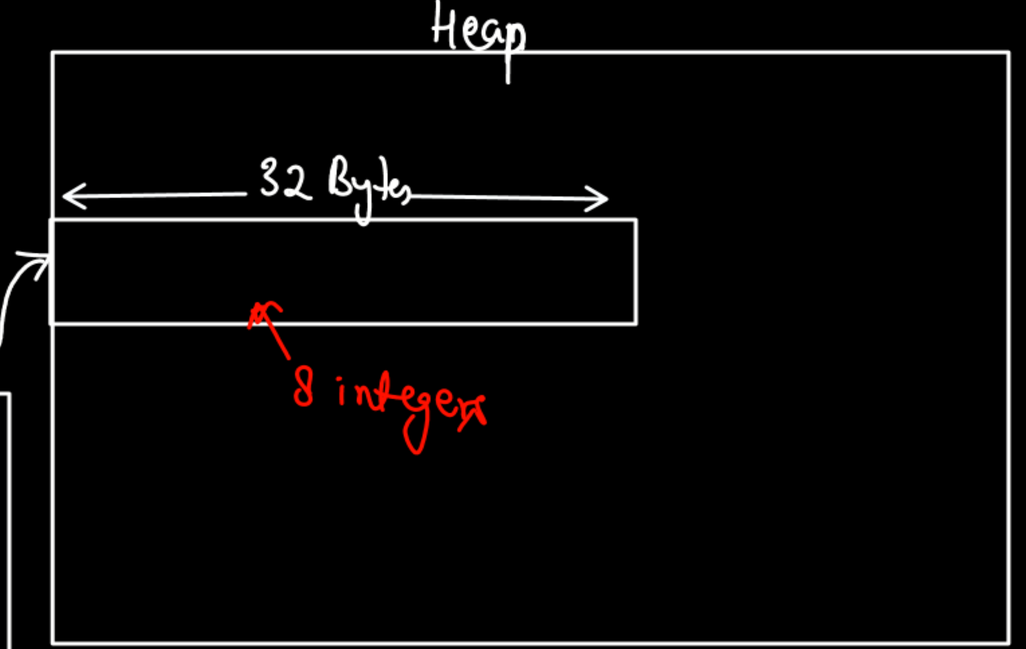
ptr = malloc (size_in_bytes) ; ← गलत

ptr = malloc (32) ; // return the address of first block

```
int *ptr = malloc (32) ;
```

↳ GCC ← int = 4 Bytes

Turbo C++ (ANSI) ← int = 2 bytes ← 16 integers



```
int *ptr = malloc(16);   <- ANSI -> int = 2 Bytes
                 |
                 -> 8 integer ✓
```

Copy

```
int *ptr = malloc(16)   <- GNU GCC 14.1 -> int = 4 Bytes
                |
                -> 4 integers
```

} insufficient space

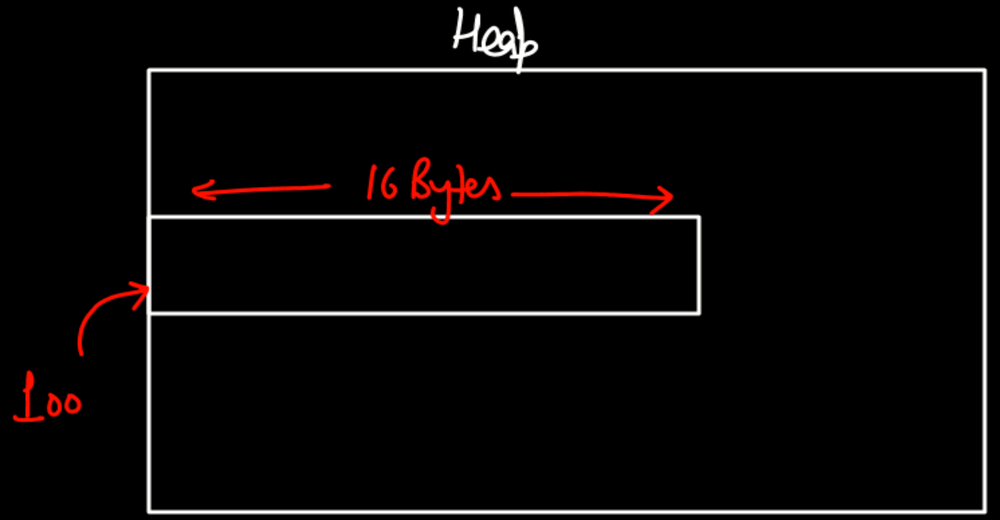
↓

मेमोरी

Rectify (सुधार करना)

```
int *ptr = (int*) malloc ( 8 * sizeof (int));
```

- `int *ptr` — ptr variable
- `(int*)` — integer pointer
- `malloc` — Return address
- `8` — no. of integers
- `sizeof (int)` — Size of integer



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

GCC → int = 4

```
int *ptr = (int *) malloc ( 4 )
```

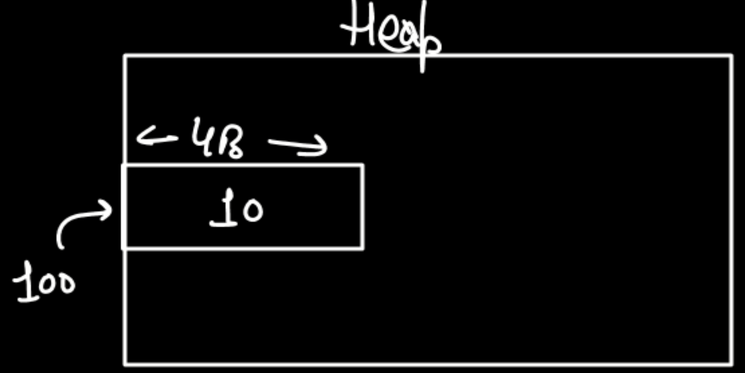
↓

4 Bytes Allocated ← first Block address from Heap is assigned to ptr

```
*ptr = 10
```

Value at (ptr) = 10

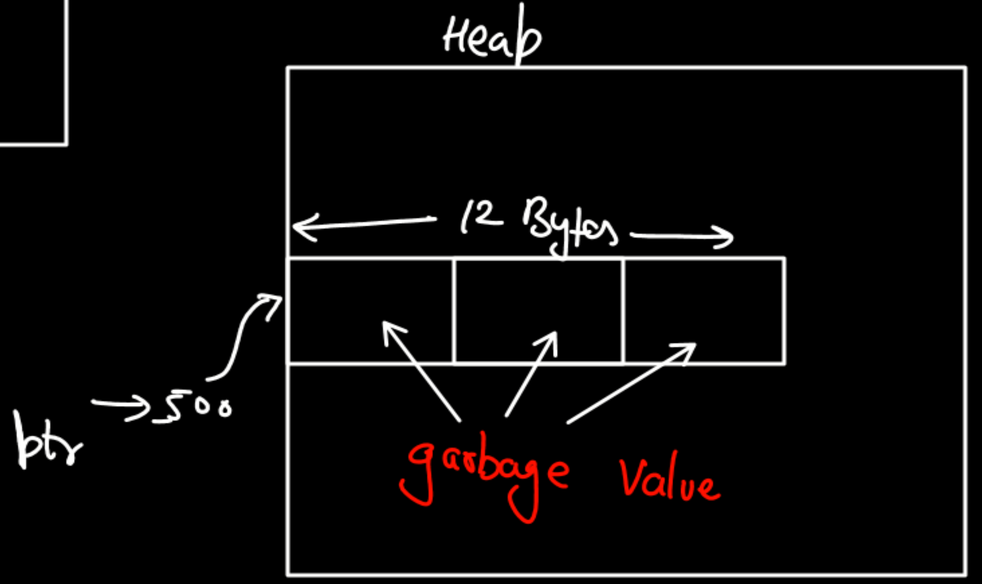
Value at (100) = 10



→ `int *ptr = (void *) malloc( 3 * sizeof (int))`

(int *)

3 * 4 = 12



free(address of allocated block)

free(ptr) ; ← ~~जिम्मेदारी~~

malloc( )

```
datatype *ptr = (void *) malloc ( no. of elements * Size of datatype);
```

⇒ Calloc ( ) ← Allocated space → [initialized to zero]