

① Concept of linked list:

⇒ Visual Memory diagram,

struct Node {

int data;

struct Node *next;

};

Node 1 : [1000] → 5 | 1008

Node 2 : [1008] → 10 | 11008

Node 3 : [1006] → 15 | Null

⇒ each node holds data and a pointer to the next node

⇒ last node's (next) is (null).

⇒ Linked List: a chain of nodes, where each node points to the next. They are dynamic, not fixed like arrays.

↳ unlike arrays, nodes are scattered in memory, linked by pointers.

↳ Node: data + pointer to next node

Head: pointer to the first node

Tail: last node (next = NULL)

↳ ex- define a struct node with (int data) and a (next) pointer, then create one node with data 42.

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

```
struct Node {
```

```
    int data;
```

```
    struct Node *next;
```

```
};
```

```
int main () {
```

```
    struct Node n;
```

```
    n.data = 42;
```

```
    n.next = NULL;
```

```
    printf("Data: %d\n", n.data);
```

```
    return 0;
```

```
}
```

⇒ creating a linked list :-

S-1: define node structure: data & a pointer

```
struct Node {
```

```
    int data;
```

```
    struct Node *next;
```

```
}
```

S-2: Create list - allocate nodes dynamically & link them

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

```
#include <stdlib.h>
```

```
int main () {
```

```
    struct Node *head = malloc (sizeof (struct Node));
```

```
    head->data = 5;
```

```
    head->next = malloc (sizeof (struct Node));
```

```
    head->next->data = 10;
```

```
    head->next->next = NULL;
```

```
struct Node *current = head;
```

```
while (current != NULL) {
```

```
    printf ("%d -> ", current->data);
```

```
    current = current->next;
```

```
}  
printf ("NULL\n");
```

```
free (head->next);
```

```
free (head);
```

```
return 0;
```

```
}
```

Ex create a linked list with 3 nodes (1,2,3) and print it.

```
↳ #include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct Node {
```

```
    int data;
```

```
    struct Node *next;
```

```
};
```

```
int main() {
```

```
    struct Node *head = malloc (sizeof (struct Node));
```

```
    head->data = 1;
```

```
    head->next = malloc (sizeof (struct Node));
```

```
    head->next->data = 2;
```

```
    head->next->next = malloc (sizeof (struct Node));
```

```
    head->next->next->data = 3;
```

```
    head->next->next->next = NULL;
```

⇒ [2000] → [1] | 2008
[2008] → [2] | 2016
[2016] → [3] | NULL

⇒ struct Node *current = head;
↳ current starts at the first node (2000).

⇒ while (current != NULL)
↳ current = 2000, prints 1
current = 2008, prints 2
current = 2016, prints 3
Stops when current = NULL

⇒ freeing:
↳ free(head->next->next) : frees third node (2016)
↳ free(head->next) : frees second node (2008)
↳ free(head) : free's first node (2000)

↳ `struct Node *head = malloc (sizeof (struct Node));`
⇒ allocates node on the heap and makes head point to it.
⇒ (head) is a pointer (heap = dynamic mem.)
⇒ eg. address = 2000
⇒ `head → data = 1` (sets the data of 1st node to 1).

↳ `head → next = malloc (sizeof (struct Node))`
↳ allocates the 2nd node & links it to the first node's next.

↳ `head → next → data = 2;` (set the 2nd node's data = 2)

↳ `head → next → next = malloc (sizeof (struct node));`
↳ allocates a third node & links it to the 2nd node's next

↳ `head → next → next → data = 3;`
↳ set the third node's data to 3

↳ `head → next → next → next = null.`

ends off by 3rd node next to null.