Short-term Hands-on Supplementary Course on C programming

Session 13:
Pointers and Structures

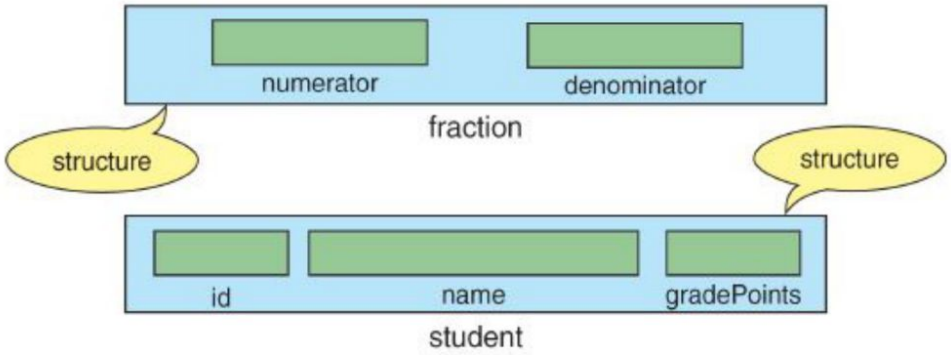
Nandakishor. V
Nitheesh Kumar. N

Agenda

1. Pointers to Structures
2. Dynamic Initialization of Structures
3. Self referential structures
4. Linked List using structures

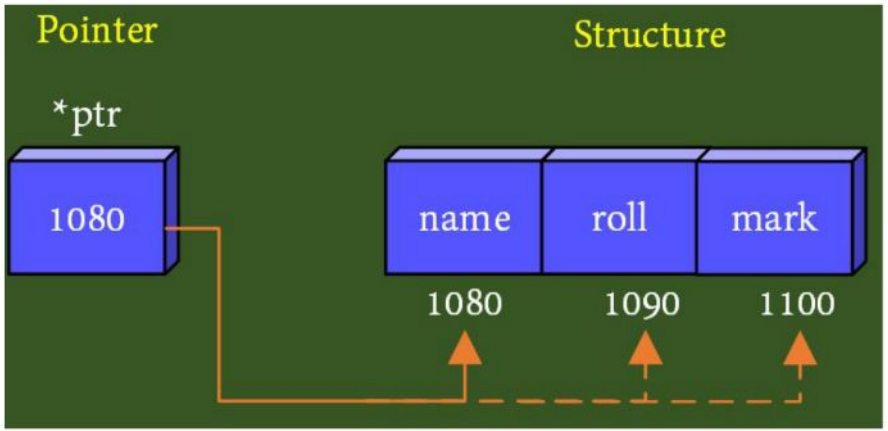
Recap - Structure

A **structure** in C is a user-defined data type. It is used to bind the two or more similar or different data types or data structures together into a single type. Structure is created using struct keyword and a structure variable is created using struct keyword and the structure tag name.



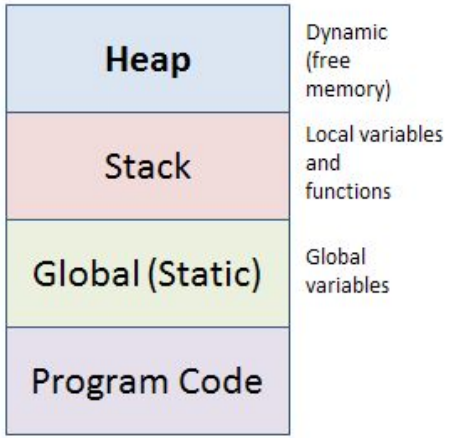
Pointer to structure

Declaration of structure pointer
Initialization of struct pointer
Accessing members through pointer
- (*) asterisk and (.) operators
- **(->) arrow operator** [recommended]



Dynamic Allocation of Memory

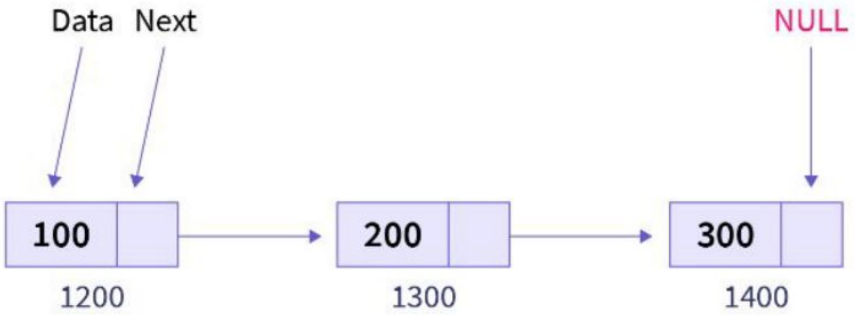
	malloc()	calloc()	free()	realloc()
Stands for	Memory allocation	Contiguous allocation	Freeing memory	Reallocation of memory
Used to	Allocate a single block of memory	Allocate multiple blocks of memory	Deallocate the dynamically allocated memory	Reallocate memory that malloc() or calloc() occupy



Self referential structures

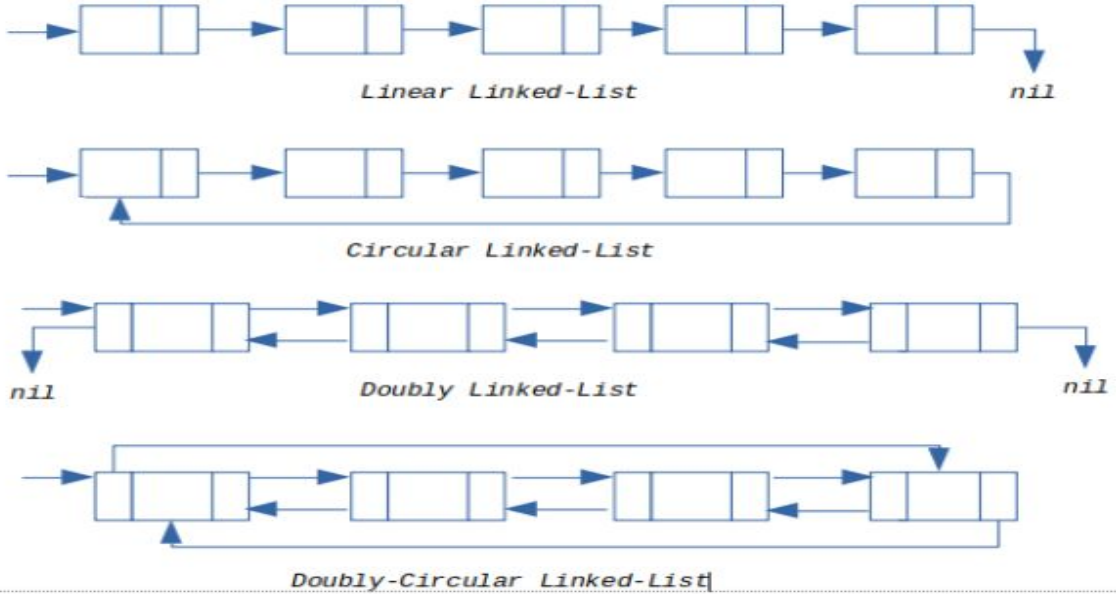
```
struct node{
    int data;
    struct node * next;
};
```

```
Struct node *head = (struct node *)malloc(sizeof(struct node));
```



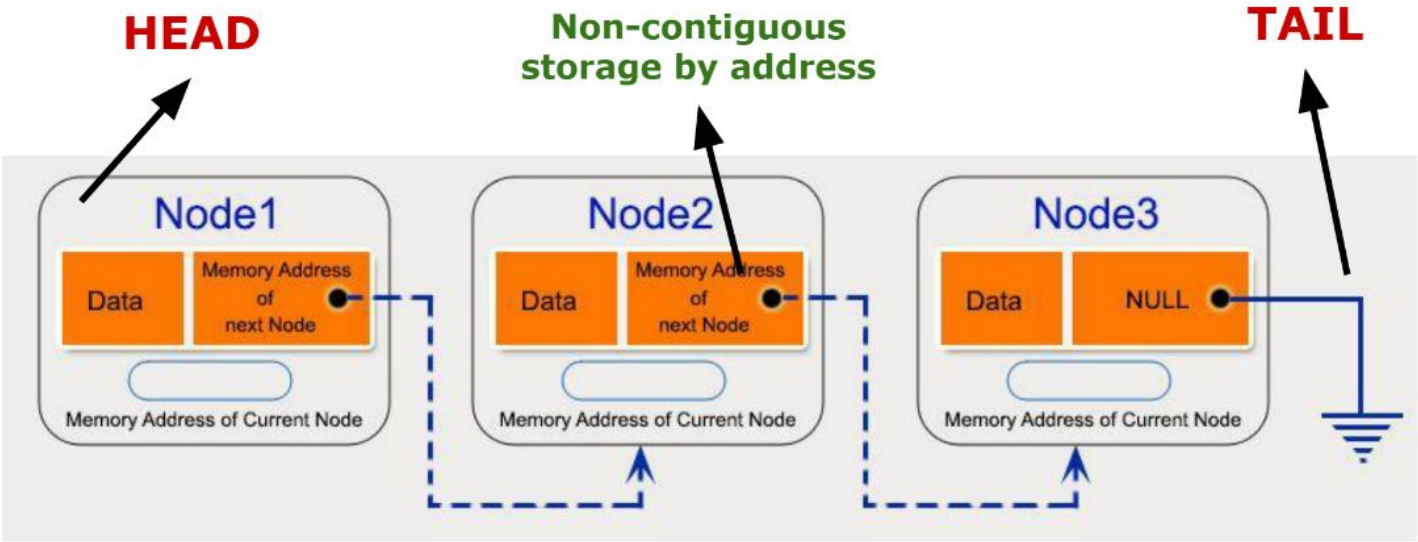
Linked List

1. A linear data structure
2. Unlike arrays, no contiguous memory is needed to store elements
3. Elements are usually regarded as nodes
4. Nodes can stores any complex custom data - defined by the underlying self referential structure
5. Classified as follows



Singly Linked List

- **Unidirectional linked list** — each node only points to its next node.
- Can be **traversed in one direction only**, i.e., from head node to tail node.



Demo Live Code: Creation of Linked List

1. Definition of underlying structure.

```
struct Node {
  int data;
  struct Node *next;
};
```

2. Development of function to insert a node in the front of linked list.

```
struct Node *insertFront(struct Node *head, int data) {
  struct Node *temp = (struct Node *)malloc(sizeof(struct Node));

  temp->data = data;
  temp->next = head;

  return temp;
}
```

3. Usage of Linked List from main.

Tutorial

Implement singly linked list in C using structures and pointers with the following functionalities :

a. Insert front
b. Insert end
c. Insert in middle
d. Delete a node
e. Display the list

Any Questions?

Thank You for attending!

Contact us regarding any questions through email

nandakishor2010608@ssn.edu.in
nitheesh2010343@ssn.edu.in