A linked list is a series of data structures that are linked together.

Structure of linked list:

- Linked list looks like collection of connected nodes.
- Here the first node is called head.
- End of the linked list, the last node is marked as NULL.
- Node connection link is called next link.
- Each node contains two filed
    - First field called data field to store data
    - Second filed called next to store node connection link.

Representation using coding:

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

Here, variable *int data* contains the data of the node.

Variable *struct Node* next* stores the address of the next node.

Basic operations:

- Insertion : Add a node at the given position.
- Deletion : Delete a node.
- Updating : Update a node.

- Traversal : Traverse all the nodes one after another.
- Searching : Search element by value.
- Sorting : Arrange nodes in specific order.
- Merging : Merge linked list.

Traversal operation:

```
void traverseLL(Node head) {
    while(head != NULL)
    {
        print(head.data)
        head = head.next
    }
}
```

Insertion operation:

```
void insertionOperation(Node head, int val)
{
    newNode = new Node(val)
    if(head == NULL)
        return newNode
    else
    {
        newNode.next = head
        return newNode
    }
}
```

Deletion operation:

```
Node deleteOperation(Node head, Node dilit)
{
    if(head == dilit)
    {
        return head.next
    }
    Node temp = head
    while( temp.next != NULL )
    {
        if(temp.next == dilit)
        {
            temp.next = temp.next.next
            delete dilit
            return head
        }
        temp = temp.next
    }
    return head
}
```

Updation operation:

```
void updateOperation(Node head, int val, int newVal)
{
    Node temp = head
    while(temp != NULL)
    {
        if( temp.data == val)
        {
            temp.data = newVal
            return
```

```
        }
        temp = temp.next
    }
}
```

Linked list program using C:

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

// Creating a node
struct node {
  int value;
  struct node *next;
};

// print the linked list value
void printLinkedlist(struct node *p) {
  while (p != NULL) {
    printf("%d ", p->value);
    p = p->next;
  }
}

int main() {
  // Initialize nodes
  struct node *head;
  struct node *a = NULL;
  struct node *b = NULL;
  struct node *c = NULL;
```

```
    // Allocate memory
    a = malloc(sizeof(struct node));
    b = malloc(sizeof(struct node));
    c = malloc(sizeof(struct node));

    // Assign value values
    a->value = 9;
    b->value = 8;
    c->value = 7;

    // Connect nodes
    a->next = b;
    b->next = c;
    c->next = NULL;

    // printing node-value
    head = a;
    printLinkedlist(head);
}
```

Output:

```
9 8 7
```

Related Posts:

1. Net 42
2. Net 14
3. Net 13
4. Net 12
5. Net 35
6. Net 32

7. Net 29
8. Net 27
9. Net 52
10. Net 51
11. Net 45
12. Net 41
13. Net 38
14. Net 37
15. Net 36
16. GATE CS | Binary tree questions | Prof. Jayesh Umre
17. GATE | Binary Search Tree | Related Questions | Prof. Jayesh Umre
18. GATE 2004, Calculate height of Binary Tree | Prof. Jayesh Umre
19. GATE 2010 Binary tree descendent | Prof. Jayesh Umre
20. Array in Data Structure
21. Traversal operation on array
22. Insertion Operation on Array
23. Data Structures: Definition, Importance, Applications and types
24. Concepts of Data and Information