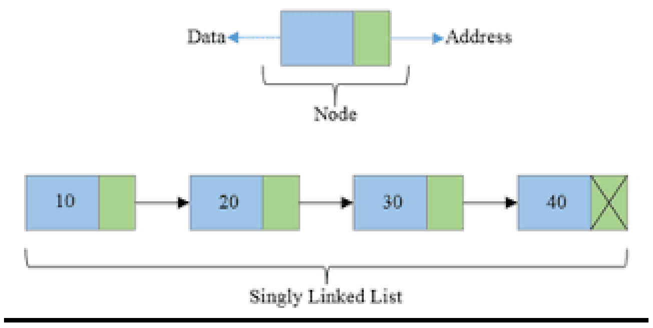
Singly Linked List

1. **Singly Linked List:** A Singly-linked list is a collection of nodes linked together in a sequential way where each node of the singly linked list contains a data field and an address field that contains the reference of the next node.



Basic Structure of Singly Linked List

```
Struct Node
{
int data;
struct Node *Next;
};
```

Advantages of Singly Linked List

- Singly linked list is probably the most easiest data structure to implement.
- Insertion and deletion of element can be done easily.
- Insertion and deletion of elements doesn't requires movement of all elements when compared to an array.
- Requires less memory when compared to doubly, circular or doubly circular linked list.
- Can allocate or deallocate memory easily when required during its execution.
- It is one of most efficient data structure to implement when traversing in one direction is required.

Disadvantages of Singly Linked List

- It uses more memory when compared to an array.
- Since elements are not stored sequentially hence requires more time to access each elements of list.
- Traversing in reverse is not possible in case of Singly linked list when compared to Doubly linked list.
- Requires O(n) time on appending a new node to end. Which is relatively very high when compared to array or other linked list.

1. C program to create and Traverse Singly Linked List

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
}*head;
void createList(int n);
void traverseList();
void main()
{
    int n;

    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);


    printf("\nData in the list \n");
    traverseList();


    getch();
}
void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
    head = (struct node *)malloc(sizeof(struct node));
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
        exit(0);
    }
    printf("Enter the data of node 1: ");
    scanf("%d", &data);
    head->data = data;
    head->next = NULL;
    temp = head;
    for(i=2; i<=n; i++)
    {
        newNode = (struct node *)malloc(sizeof(struct node));
```

```
            if(newNode == NULL)
        {
           printf("Unable to allocate memory.");
           break;
        }

        printf("Enter the data of node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;
        temp->next = newNode;
        temp = temp->next;        }
}
void traverseList()
{
    struct node *temp;
if(head == NULL)
    {
        printf("List is empty.");
        return;
    }

    temp = head;
    while(temp != NULL)
    {
        printf("Data = %d\n", temp->data);
        temp = temp->next;
    }
}
```

2 C program to insert new node at beginning in singly linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
}*head;
void createList(int n);
void insertNodeAtBeginning(int data);
void displayList();
void main()
{
    int n, data;
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();
    printf("\nEnter data to insert at beginning of the list: ");
    scanf("%d", &data);
    insertNodeAtBeginning(data);

    printf("\nData in the list \n");
    displayList();

    getch();
}

void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;

    head = (struct node *)malloc(sizeof(struct node));
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);
```

```
head->data = data;
head->next = NULL;
temp = head;
for(i=2; i<=n; i++)
{
    newNode = (struct node *)malloc(sizeof(struct node));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
        break;
    }
    else
    {
        printf("Enter the data of node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;
        temp->next = newNode;
        temp = temp->next;
    }
}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
```

```
    }
}
void insertNodeAtBeginning(int data)
{
    struct node *newNode;

    newNode = (struct node*)malloc(sizeof(struct node));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        newNode->data = data;
        newNode->next = head;

        head = newNode;

        printf("DATA INSERTED SUCCESSFULLY\n");
    }
}

void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);
            temp = temp->next;
        }
    }
}
```

3 C program to insert new node at the End of singly linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
}*head;
void createList(int n);
void insertNodeAtEnd(int data);
void displayList();
void main()
{
    int n, data;
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();
    printf("\nEnter data to insert at end of the list: ");
    scanf("%d", &data);
    insertNodeAtEnd(data);

    printf("\nData in the list \n");
    displayList();

    getch();
}

void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
    head = (struct node *)malloc(sizeof(struct node));



    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data;
head->next = NULL;
        temp = head;
        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data;
                newNode->next = NULL;
                temp->next = newNode;
              temp = temp->next;
            }
        }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
void insertNodeAtEnd(int data)
{
    struct node *newNode, *temp;

    newNode = (struct node*)malloc(sizeof(struct node));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        newNode->data = data; // Link the data part
        newNode->next = NULL;

        temp = head;
        while(temp->next != NULL)
            temp = temp->next;

        temp->next = newNode;
        printf("DATA INSERTED SUCCESSFULLY\n");
    }
}
void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);
            temp = temp->next;
        }
    }
```

```
}
```

4. **C program to insert new node at the Middle of singly linked list.**

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
}*head;
void createList(int n);
void insertNodeAtMiddle(int data, int position);
void displayList();
void main()
{
    int n, data, position;
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();
    printf("nEnter data to insert at middle of the list: ");
    scanf("%d", &data);
    printf("Enter the position to insert new node: " );
    scanf("%d", &position);
    insertNodeAtMiddle(data, position);
    printf("\nData in the list \n");
    displayList();


    getch();
}
void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;


    head = (struct node *)malloc(sizeof(struct node));
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);


        head->data = data;
        head->next = NULL;


        temp = head;


        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data;
                newNode->next = NULL;

                temp->next = newNode;
                temp = temp->next;
            }
        }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
```

```
void insertNodeAtMiddle(int data, int position)
{
    int i;
    struct node *newNode, *temp;

    newNode = (struct node*)malloc(sizeof(struct node));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        newNode->data = data;
        newNode->next = NULL;

        temp = head;
 for(i=2; i<=position-1; i++)
        {
            temp = temp->next;

            if(temp == NULL)
                break;
        }
        if(temp != NULL)
        {
                newNode->next = temp->next;

            temp->next = newNode;

            printf("DATA INSERTED SUCCESSFULLY\n");
        }
        else
        {
            printf("UNABLE TO INSERT DATA AT THE GIVEN POSITION\n");
        }
    }
}

void displayList()
{
    struct node *temp;
if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
           printf("Data = %d\n", temp->data);
           temp = temp->next;
        }
    }
}
```