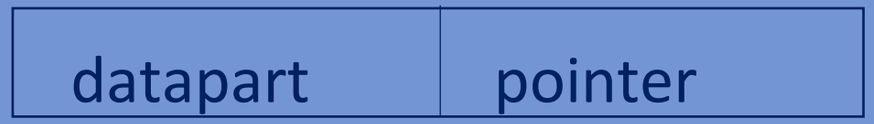
Data Structures

Creating and traversing the linked list

Creating a node:

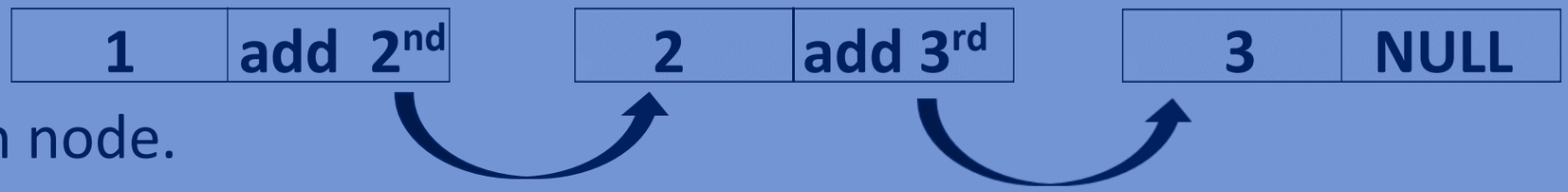


- Create a node for a single linked list.
- Data part contains an integer.
- Pointer contains the address of the next node.
- typedef is used for easy accessing of structure variables.

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

//cresting a node.
typedef struct lin_list{
    int data;
    struct lin_list *next;
}lin_list;
```

Creating a linked list:



- Declaring each node.
- Allocate memory for all the nodes.
- Store the data to in respective nodes and pointing to the next node.
- Printing the linked list.
- We traverse the linked list using the head node.

```
int main(){
    lin_list *head=NULL;
    lin_list *second=NULL;
    lin_list *third=NULL;

    head=(lin_list*)malloc(sizeof(lin_list));
    head->data=1;
    second=(lin_list*)malloc(sizeof(lin_list));
    head->next=second;
    second->data=2;
    third=(lin_list*)malloc(sizeof(lin_list));
```

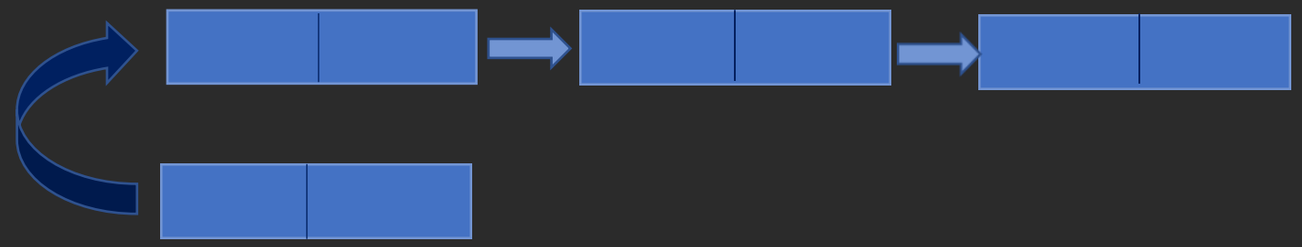
```
    second->next=third;
    third->data=3;
    third->next=NULL;
    printf("first node-%d \n",head->data);
    printf("second node from first node-%d \n",head->next->data);
    printf("second node-%d \n",second->data);
    printf("third node from second node-%d \n",second->next->data);
    printf("third node-%d \n",third->data);
    while(head!=NULL){
        printf("%d ",head->data);
        head=head->next;
    }
    return 0;
}
```

```
first node-1
second node from first node-2
second node-2
third node from second node-3
third node-3
1 2 3
```

Inserting a node into a linked list:

- Write a function to insert element into linked list.
- Return type is pointer to structure.
- Allocate memory for newly added node.
- Store the data and add the new node to the linked list.

```
lin_list  *insertnode(lin_list *head,int data){
    lin_list *temp=(lin_list*)malloc(sizeof(lin_list));
    temp->data=data;
    temp->next=NULL;
    if(head==NULL){
        head=temp;
    }
    else{
        temp->next=head;
        head=temp;
    }
    return head;
}
```



Traversing the linked list:

- Traverse the linked list using the head node.
- Head contains the starting address of the linked list.
- head->next contains the address of the next node of the linked list.
- head=head->next means we are entering into next node,now head represent next node of the previous head node.

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

Whole program:

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

//cresting a node.
typedef struct lin_list{
    int data;
    struct lin_list *next;
}lin_list;

lin_list  *insertnode(lin_list *head,int data){
    lin_list *temp=(lin_list*)malloc(sizeof(lin_list));
    temp->data=data;
    temp->next=NULL;
    if(head==NULL){
        head=temp;
    }
    else{
        temp->next=head;
        head=temp;
```

```
    }
    return head;
}

int main(){
    lin_list *head=NULL;
    head=insertnode(head,5);
    head=insertnode(head,6);
    head=insertnode(head,7);
    while(head!=NULL){
        printf("%d ",head->data);
        head=head->next;
    }
    return 0;
}
```

Output:

```
7 6 5
```