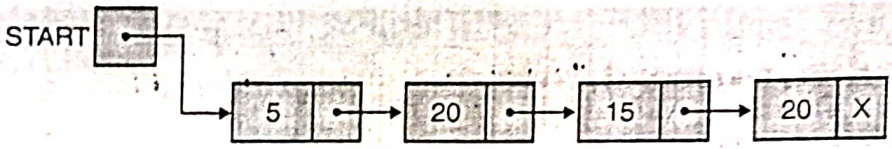
C++ Code - Linked List Implementation

```
#include <iostream.h>

//Declare Node
struct Node{
    int num;
    Node *next;
};

//Declare starting (Head) node
struct Node *head=NULL;

//Insert node at start
void insertNode(int n){
    struct Node *newNode=new Node;
    newNode->num=n;
    newNode->next=head;
    head=newNode;
}
 //Traverse/ display all nodes (print items)
void display(){
    if(head==NULL){
        cout<<"List is empty!"<<endl;
        return;
    }
    struct Node *temp=head;
    while(temp!=NULL){
        cout<<temp->num<<" ";
        temp=temp->next;
    }
    cout<<endl;
}

//delete node from start
void deleteItem(){
    if(head==NULL){
        cout<<"List is empty!"<<endl;
        return;
    }
    cout<<head->num<<" is removed."<<endl;
    head=head->next;
}
int main(){

    display();
    insertNode(10);
    insertNode(20);
    insertNode(30);
    insertNode(40);
    insertNode(50);
    display();
    deleteItem(); deleteItem(); deleteItem(); deleteItem();
deleteItem();
    deleteItem();
    display();
    }
```



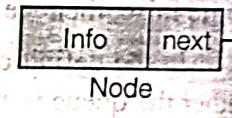
The left part of each node represents information and the other part represents a poin containing the address of the next node in the list.

Each linked list contains a pointer variable **start** which contains the address of the first no in the list. The last node of the list contains the null value in the pointer field which represents e of the linked list.

To implement a linked list which is a collection of nodes of same type, we use self refere structures which is a structure having a reference to itself. It can be defined as,

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

This declaration of structure **node** consists of two members, the first one is the information part and the other one is a self referential pointer member which points to the next node of the list. The above declaration of the structure node can be depicted as,



Prog 18.4 Program to create and display a linked list using object and classes ?

```
#include<conio.h>
#include <iomanip.h>
#include <iostream.h>
struct node
{
  int info;
  node* next;
};
class linked_list
  {
    node* start;
  public:
    linked_list()
    { start=NULL; }
    void create();
    void display();
    ~linked_list();
};
int main()
{
  clrscr();
  linked_list ll;//creating an object representing linked list
  ll.create();
  cout<<"linked list is :";
```

```
ll.display();
getch(); return 0;
}
oid linked_list::create()
{
node* ptr;
char ch='y';
int x;
while(ch=='y')
 {
   ptr=new node;  //creating a node dynamically
   cout<<"enter info. part of new node=";
   cin>>x;
   ptr->info=x;//assign value x to info part of new node
   ptr->next=start;
   start=ptr;
   cout.flush();
   cout<<"\nwant to input more: n to quit\n";
   cin>>ch;
  }
oid linked_list::display()
{
 node *ptr=start; //temporary pointer to first node
 while(ptr!=NULL)
 {
   cout<<ptr->info<<'\t';
   ptr=ptr->next;
 }
inked_list::~linked_list()
{
 node *ptr=start;
 cout<<"destroying... linked list\n";
 while(ptr!=NULL)
 {
   ptr=ptr->next; //pointing to next node
   delete start; // delete first node
   start=ptr;  //update start with address of second node
 }
```

put :

er info part of new node = 30↵
t to input more ? (y/n) = y
er info part of new node = 20↵
t to input more ? (y/n) = y
er info part of new node = 10↵
t to input more ? (y/n) = n
ed list is : 10 20 30
easing memory