Linked List

Like arrays, linked list is a type of linear data structure. But unlike arrays the elements are not stored in contiguous memory locations.

* Linked list could be used where the structure changes, unlike the arrays where the structure is fixed.
* Linked list is the beautiful application of classes.
* Linked list contains nodes. And Each node has 2 elements
* Data => Contains the information to be stored
* \*next=> Contains the address of the nest node. The nodes in a linked list are linked using this pointer. Each node points towards the next node.
* The first node is called Head.
* If the linked list is empty then the head points towards NULL.

NULL

112

108

104

112

108

104

100

**DATA**

Stores the data to be stored

**\*Next**

Stores the address of next nodes

NULL

Head Node

**Q)** Why linked list?

Since array is also a linear data structure, so it can be also used in place of linked list. But arrays have some limitations: -

* Fixed size: - size of the array is fixed, so we must know the upper limit, to avoid errors.
* Inserting a new element in an array of elements is expensive because the room has to be created for the new elements and to create room existing elements have to be shifted.

Advantages of linked list over arrays

* Dynamic size: - no fixed size issue, unlike arrays
* Easy insertion and deletion of elements
* Starting with Linked List

Simple Linked list in C++

///class for linked list

Class Node {

Public:

int data;

Node\*next;

///constructor

Node (int d) {

This->data=d;

next=NUL

}}

**Note: Follow the other document for better understanding**

Printing Linked List

Algo: - In linked list, if you know the address of the head node then you know the complete linked list. You don’t need address of each node, just the head node.

To print the list,

* Use a function with arguments as head node.
* Use while loop to check that the next of node! =Null
* Print the data

Code

Void print (Node\*head)

Node\*temp=head;///always try not to manipulate with head node. As if you lost the address of head node, ///you completely lost the linked list. So always copy the head and manipulate on them

While (temp! =Null) {

cout<<temp->data<<endl;///printing data

Temp=temp->next;///updating temp

}}

Taking input from the user and storing it in the Linked list dynamically

Algo: - We will create a function that takes input from the user continuously. And whenever the user enters -1 the function will terminate and come out of the loop. And we will return the head of the linked list

Node\* takeinput(void) {/// function

int data; cin>>data; ///taking the input from the user

node\*head=NULL; node\*tail=NULL;

while (data! =-1)///stopping criteria {

node\*n= new node(data) ///calling the parameterised constructor

if(head==NULL) {head=n; tail=n}

else{tail->next=n; tail=n;}cin>>data;}

Deleting ith node from the Linked list: -

Algo: - Again we will create function. Suppose we have to delete i=4th node. then we will iterate the linked list till i=3rd node and will change the pointer, which is pointing towards 4th node to 5th node. And finally return new head.

Node\*deleteithNode (node\*head, int i) {

Node\*curr=head; int len=length(head);

//we have used a function to count the total nodes in the linked list.

if(i==0) {head=head->next;

return head;}

if(i>0 && i<len) {

for (int j=0; j<i-1; j++) {//iterating and updating the curr node.

curr=curr->next;}

curr->next=curr->next->next;///breaking the old relation and establishing new.

Return head;}

Return head;///if i>len, it is invalid, so we will not do anything, just return it.