Linked List | Set 1 (Introduction)

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

Why Linked List?

Arrays can be used to store linear data of similar types, but arrays have following limitations.

1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.

2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

For example, in a system if we maintain a sorted list of IDs in an array id[].

id[] = [1000, 1010, 1050, 2000, 2040].

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

Advantages over arrays

1) Dynamic size

2) Ease of insertion/deletion

Drawbacks:

1) Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists.

2) Extra memory space for a pointer is required with each element of the list.

Representation in C:

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.

Each node in a list consists of at least two parts:

1) data

2) pointer to the next node

In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

```
struct node
```

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

Run on IDE

First Simple Linked List in C Let us create a simple linked list with 3 nodes.

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

struct node
{
  int data;
  struct node *next;
};

// Program to create a simple linked list with 3 nodes
int main()
{
  struct node* head = NULL;
  struct node* second = NULL;
  struct node* third = NULL;

  // allocate 3 nodes in the heap
  head = (struct node*)malloc(sizeof(struct node));
  second = (struct node*)malloc(sizeof(struct node));
  third = (struct node*)malloc(sizeof(struct node));

  /* Three blocks have been allocated  dynamically.
     We have pointers to these three blocks as first, second and third
       head           second           third
        |               |               |
        |               |               |
    +---+-----+     +----+----+     +----+----+
    | #  | #  |     | #  | #  |     |  # |  # |
    +---+-----+     +----+----+     +----+----+

   # represents any random value.
   Data is random because we haven't assigned anything yet  */

  head->data = 1; //assign data in first node
  head->next = second; // Link first node with the second node

  /* data has been assigned to data part of first block (block
    pointed by head).  And next pointer of first block points to
    second.  So they both are linked.

       head          second         third
        |              |              |
        |              |              |
    +---+---+     +----+----+     +-----+----+
    | 1  | o----->| #  | #  |     |  #  | #  |
    +---+---+     +----+----+     +-----+----+
  */

  second->data = 2; //assign data to second node
  second->next = third;

  /* data has been assigned to data part of second block (block pointed by
     second). And next pointer of the second block points to third block.
    So all three blocks are linked.
```

```
    head         second         third
     |             |              |
     |             |              |
  +---+---+     +---+---+     +----+----+
  | 1  | o----->| 2 | o-----> |  # |  # |
  +---+---+     +---+---+     +----+----+      */

  third->data = 3; //assign data to third node
  third->next = NULL;

  /* data has been assigned to data part of third block (block pointed
    by third). And next pointer of the third block is made NULL to indicate
    that the linked list is terminated here.

     We have the linked list ready.

           head
             |
             |
        +---+---+     +---+---+       +----+------+
        | 1  | o----->|  2  | o-----> |  3 | NULL |
        +---+---+     +---+---+       +----+------+


    Note that only head is sufficient to represent the whole list.  We can
    traverse the complete list by following next pointers.    */

  getchar();
  return 0;
}
```

Run on IDE

Linked List Traversal

In the previous program, we have created a simple linked list with three nodes. Let us traverse the created list and print the data of each node. For traversal, let us write a general purpose function printList() that prints any given list.

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

struct node
{
  int data;
  struct node *next;
};

// This function prints contents of linked list starting from the given node
void printList(struct node *n)
{
  while (n != NULL)
  {
     printf(" %d ", n->data);
     n = n->next;
  }
}

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

```
  // allocate 3 nodes in the heap
  head  = (struct node*)malloc(sizeof(struct node));
  second = (struct node*)malloc(sizeof(struct node));
  third  = (struct node*)malloc(sizeof(struct node));

  head->data = 1; //assign data in first node
  head->next = second; // Link first node with the second node

  second->data = 2; //assign data to second node
  second->next = third;

  third->data = 3; //assign data to third node
  third->next = NULL;

  printList(head);

  getchar();
  return 0;
}
```

Run on IDE

Output:

```
1  2  3
```

You may like to try **Practice MCQ Questions on Linked List**

We will soon be publishing more posts on Linked Lists.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Category: Data Structures

What does struct node*head=NULL mean? and what if it were like struct node*head; only and we passed it as a pointer. what is the difference b/w these two pointers?
plz answer it anybody with explanation

28 ▲ | ▼ • Reply • Share ›

Ankit Singh ➚ sonu431 • a year ago

We must ensure that before allocating memory addresses to the nodes,the node pointers must contain null values so that they cannot point to any random memory location which is surely undesirable.That is why the statement struct node*head=NULL.

16 ▲ | ▼ • Reply • Share ›

sonu431 ➚ Ankit Singh • a year ago

take this as an example.. here

```
void reverse(struct node** head_ref)
{
struct node* prev = NULL;
struct node* current = *head_ref;
struct node* next;
while (current != NULL)
{
next = current->next;
current->next = prev;
prev = current;
current = next;
}
*head_ref = prev;
}
```

in above code *prev=NULL,
why not *next=NULL, but if we make it point to
NULL it doesnt make any difference..
but in *prev=NULL instead of it if we take as only
*prev it is giving not proper o/p.. plz explain..

1 ▲ | ▼ • Reply • Share ›

unlimited700 ➚ sonu431 • a year ago

If you will not initialize prev it will have random value right ? Now when you execute first time above while loop the instruction current->next = prev ; will assign next of current any random value rather than NULL and this current will work as a last node after execution of whole function . In this case when we try to traverse the whole list we can are not sure to terminate because this last node will point to any random value rather than NULL (As you

know we usually put a condition in which we check traverse the list while we not reach the NULL) .

7 ▲ | ▼ • Reply • Share ›

AbhinavB09 ➦ Ankit Singh • a month ago

Ankit it is not compulsory though. It works almost fine by sonu431's method. The need of struct node*head=NULL is still not clear.
Please help someone.

▲ | ▼ • Reply • Share ›

BeautifulCode ➦ Ankit Singh • 2 months ago

It is usually not feasible to not assign some memory location to this pointer.

Even we don't do :

struct node*head=NULL; we just declare a variable pointer struct node * head;

we will be assigning some memory location reference to it right ?

head =(struct node*)malloc(struct node);

It never sounds a great thing to me.

Why would someone pass such pointer to any method ? To do what with that variable ?

▲ | ▼ • Reply • Share ›

vimal • 10 months ago

why there is no previous and next button on this site..

6 ▲ | ▼ • Reply • Share ›

Manas Sharma • a year ago

What does -> operator do? And why can't we use dot operator for the same purpose i.e (head.data=1;) instead of (head->data=1;)

6 ▲ | ▼ • Reply • Share ›

Gaurav Suman ➦ Manas Sharma • a year ago

First, head.data is wrong , it should be (*head).data because u need to dereference the head pointer first.

-> provides short hand for this.
As an example consider changing next element from head.
using dot operator: (*((*head).next)).data = 1
using -> operator: head->next->data = 1

17 ▲ | ▼ • Reply • Share ›

Manas Sharma ➚ Gaurav Suman • a year ago

thanks!

• Reply • Share ›

Pradeep • a year ago

Why getcher() is used at end of the program?

6 • Reply • Share ›

Alex ➚ Pradeep • a year ago

to stop output screen...until you enter a character.

8 • Reply • Share ›

kaps • 10 months ago

In the declaration:
struct node
{
int data;
struct node *next;
};

how are we able to declare a pointer to a structure, which is
not yet declared fully ?

3 • Reply • Share ›

Rajat Kumar Seth ➚ kaps • 10 months ago

struct node is used to create a user defined data type to store data in Primary Memory (RAM)..
Since we are making a user defined datatype, so it can include all the predefined datatypes which includes pointers also..

• Reply • Share ›

Manas Sharma • a year ago

In the statement

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

What is the significance of type-casting (struct node*)? And how will be my result be affected if I don't do this type-casting.

And what is the significance of * in this type-cast?

3 • Reply • Share ›

anonymous ➚ Manas Sharma • a year ago

Because malloc returns a pointer to void, i.e., it is simply allocating chunks of memory with no regard as to the data that will be stored there.

2 • Reply • Share ›

Rahul K Kaushik ➚ Manas Sharma • 7 months ago

as malloc returns a pointer we need to typecast it into a pointer which we do through (node *)

1 • Reply • Share ›

sumit rawat ➚ Manas Sharma • 8 days ago

No you dont need to type cast malloc result. malloc function returns void pointer which automatically converts to the type of pointer is to allocated to. See this http://stackoverflow.com/quest...
Also its a bad practice to typecast malloc results as it will make difficult to maintain the code.

• Reply • Share ›

peru • 8 months ago

please..cut down some advertisements...they distract a lot..

2 • Reply • Share ›

Mahesh G V P ➚ peru • 3 months ago

Use Ad Block Plus

3 • Reply • Share ›

Satyam • a year ago

what is difference between 'struct node *temp' and 'struct node* temp' please comment fast

2 • Reply • Share ›

Adauta Garcia Ariel ➚ Satyam • 7 months ago

Hello Satyam. It's something about "style" I've found people who likes write:

int* iptr;

Other people:

int *iptr;

Just we have to be careful when we declare more than one variable in one line.

int *iptr, var;

iptr is a pointer to int, "var" is an int.

If we want to declare "var" as a pointer we do it as follows

If we want to declare "var" as a pointer we do it as follows:

int *iptr, *var;

There are people who read that declarations as follows.

"var" is variable pointer wich point to an int.

iptr is a variable pointer wich point to an int.

But it's a little weird if you do it as follows.

int* iptr, *var;

In summary : It's just about readability

5 ^ | v • Reply • Share ›

Shaurabh ➚ Satyam • a year ago

No difference!

2 ^ | v • Reply • Share ›

RK ➚ Satyam • a year ago

There is no difference, but it would be better for understanding if you use 'struct node *temp'

1 ^ | v • Reply • Share ›

Himanshu Dagar • 2 years ago

very very helpful discussion

2 ^ | v • Reply • Share ›

Ashish Patil • a year ago

very helpful post,

1 ^ | v • Reply • Share ›

sumit rawat • 8 days ago

@GeeksforGeeks You should not encourage type casting malloc function's return pointer to the type of the pointer is to allocated to. There is no need to do that. Simply head=malloc (sizeof(struct node)); will do fine.

^ | v • Reply • Share ›

PETERU Satyanarayana • 8 days ago

Why head is in printlist not another second and third

^ | v • Reply • Share ›

qwerty • 14 days ago

How can we use struct node *next in struct node?I mean struct node is not completely

defined right?

• Reply • Share ›

Ankit Singh ➚ qwerty • 13 days ago

Rajat Kumar Seth has given answer to this question , you can see it above in the comments , When sort the comments by Best.

• Reply • Share ›

anonymus • 16 days ago

http://linkedlistmadeeasy.blog...

• Reply • Share ›

Anshul katta • a month ago

is c still working ... the world is moving to object oriented please...

• Reply • Share ›

shreya • a month ago

what is the meaning of fun2(start->next->next) ?

• Reply • Share ›

Anil ➚ shreya • a month ago

Here this statement means, we are passing the third node of the list to the function "fun2" (If "start" points to the first node).

start ********** First Node
stat->next ********** Second Node
start->next->next ********** Third Node

1 • Reply • Share ›

shreya • a month ago

why we use to write struct node *next?what is the literal meaning of node?
Is it a way of initiallizing?

• Reply • Share ›

Ankit Singh ➚ shreya • 13 days ago

There can be more than one user defined structure in your program so you need to point out which structure you want to point out using (*next) pointer.

• Reply • Share ›

erol yeniaras • 2 months ago

FYI: Practice MCQ Questions on Linked List link is broken.

• Reply • Share ›

GeeksforGeeks Mod ↗ erol yeniaras • 2 months ago

Thanks for pointing this out. We have updated the link.

^ | v • Reply • Share ›

Vivek Kalola • 2 months ago

To create Heap here they used "head, second, third".. instead of using these can't we use a simple array "a[3]" (a[0]=head, a[1]=second, a[2]=third).? It can be easy to make long heap. i tried this but can't do it properly. any suggestion about this idea ???

^ | v • Reply • Share ›

Holden • 2 months ago

Heap should change to linkedlist:
// allocate 3 nodes in the heap

^ | v • Reply • Share ›

Junaid • 3 months ago

What is the meaning of " malloc " here in above link list example...???

^ | v • Reply • Share ›

chris ↗ Junaid • 3 months ago

malloc is the function in C to allocate the size it takes the int value and allocates that many memory space. and when it returns the pointer we need to type cast as per requirement.

^ | v • Reply • Share ›

BeautifulCode ↗ chris • 2 months ago

It will allocate memory for both int and the pointer to the next node :)

^ | v • Reply • Share ›

radek • 3 months ago

what difference does it make if we declare a normal pointer say int *ptr rather than creating a structure pointer ..?

```
struct sll {
int data;
int *ptr; };

vs

struct sll {
int data;
struct sll *ptr; };
```

^ | v • Reply • Share ›

BeautifulCode ↗ radek • 2 months ago

let's create the linkedlist with the given structure :
int * x ; //pointer to say value 3
struct node * head = NULL; // one with int node * ptr

struct node * second = NULL; //one with struct node * ptr

struct node * third = NULL;

head->data = 1;

head->ptr = ?? is expecting pointer to an integer only

so head->ptr = x;

How will you proceed further creating the linkedlist ?

You can't form a linkedlist with this way

• Reply • Share ›

BeautifulCode ➚ BeautifulCode • a month ago
I meant int * ptr
• Reply • Share ›

BeautifulCode ➚ radek • 2 months ago
Your next pointer should point to a node which has space for data and pointer to another node
not an integer pointer
• Reply • Share ›

PHILISTINE • 5 months ago
ABOVE CODE IS GIVING ERROR FOR VISUAL STUDIO
• Reply • Share ›

Rajeev ➚ PHILISTINE • 5 months ago
Could you post the error you got?
• Reply • Share ›

sneha • 5 months ago
in the declaration section,
why we have used struct node *next??
i mean why struct node?
• Reply • Share ›

Load more comments