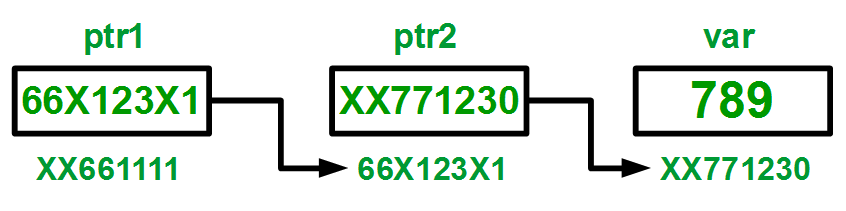
**Double Pointer**

We already know that a pointer points to a location in memory and thus used to store address of variables. So, when we define a pointer to pointer. The first pointer is used to store the address of second pointer. That is why they are also known as double pointers.

**How to declare a pointer to pointer in C?**  
Declaring Pointer to Pointer is similar to declaring pointer in C. The difference is we have to place an additional ‘\*’ before the name of pointer.  
**Syntax**:

int \*\*ptr; // declaring double pointers

Below diagram explains the concept of Double Pointers:  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/pointer2pointer.png)

The above diagram shows the memory representation of a pointer to pointer. The first pointer ptr1 stores the address of the second pointer ptr2 and the second pointer ptr2 stores the address of the variable.

// C program to demonstrate pointer to pointer

int main()

{

int var = 789;

// pointer for var

int \*ptr2;

// double pointer for ptr2

int \*\*ptr1;

// storing address of var in ptr2

ptr2 = &var;

// Storing address of ptr2 in ptr1

ptr1 = &ptr2;

// Displaying value of var using

// both single and double pointers

printf("Value of var = %d\n", var );

printf("Value of var using single pointer = %d\n", \*ptr2 );

printf("Value of var using double pointer = %d\n", \*\*ptr1);

return 0;

}

DLL

#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node \*next;

struct Node \*prev;

}node;

void insert(node \*\*head\_addr,node \*\*tail\_addr, int data)

{

// Use tail pointer to insert at the end

// Allocate memory for the new node

node \*newnode;

newnode=(node \*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

newnode->prev=NULL;

//if the list is empty

if(\*head\_addr==NULL)

{

\*head\_addr=newnode;

\*tail\_addr=newnode;

}

else

{

(\*tail\_addr)->next=newnode;

newnode->prev=\*tail\_addr;

\*tail\_addr=newnode;

}

}

void \*del\_last(node \*\*head\_addr,node\*\* tail\_addr)

{

node \*temp=\*tail\_addr;

if(temp!=NULL)

{

if(\*head\_addr==\*tail\_addr)

\*head\_addr=\*tail\_addr=NULL;

else

{

(\*tail\_addr)->prev->next=NULL;

\*tail\_addr=(\*tail\_addr)->prev;

}

free(temp);

}

else

printf("\n list is empty\n");

}

void printforward(node \*head)

{

if(head==NULL)

return;

printf("%d\n", head->data);

printforward(head->next);

return;

}

void printbackward(node \*tail)

{

if(tail==NULL)

return;

printf("%d\n", tail->data);

printbackward(tail->prev);

}

node \*find(node\* head, int value)

{

while(head!=NULL && head->data!=value)

head=head->next;

return head;

}

int main()

{

int data,choice, printoption;

// head always points to the first node of the linked list

// tail always points to the last node of the linked list

node \*head,\*tail,\*temp;

tail=head=NULL;

do

{

printf("1. Insert \n");

printf("2. Display the list\n");

printf("3.Find\n");

printf("4.Delete last\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the data to be inserted\n");

scanf("%d",&data);

insert(&head,&tail,data);

break;

case 2:

printf("1.forward?\n 2. backward?\n");

scanf("%d",&printoption);

if(printoption==1)

printforward(head);

else

printbackward(tail);

break;

case 3:

printf("Enter the data to be found\n");

scanf("%d",&data);

temp=find(head,data);

if(temp==NULL)

printf("data is not found\n");

else

printf("data %d is found\n", temp->data);

break;

case 4:

del\_last(&head,&tail);

}

}

while(choice<=4);

return 0;

}

* 1. Write a C program to find local maximum elements. An element is a local maximum, if it's greater than both its neighbors. In above list 34 and 78 are local maximum.
  2. Write a C program, which inserts an element between every pair of consecutive elements. The new element is the sum of neighbor. In above case 12 46 34 90 56 134 78 93 15 80 65 137 72.
  3. Read two locations. Write a C program to delete all elements between them. e.g. if linked list is 12 34 56 78 15 65 72 and locations are 2 and 5 then final linked list is 12 65 72.