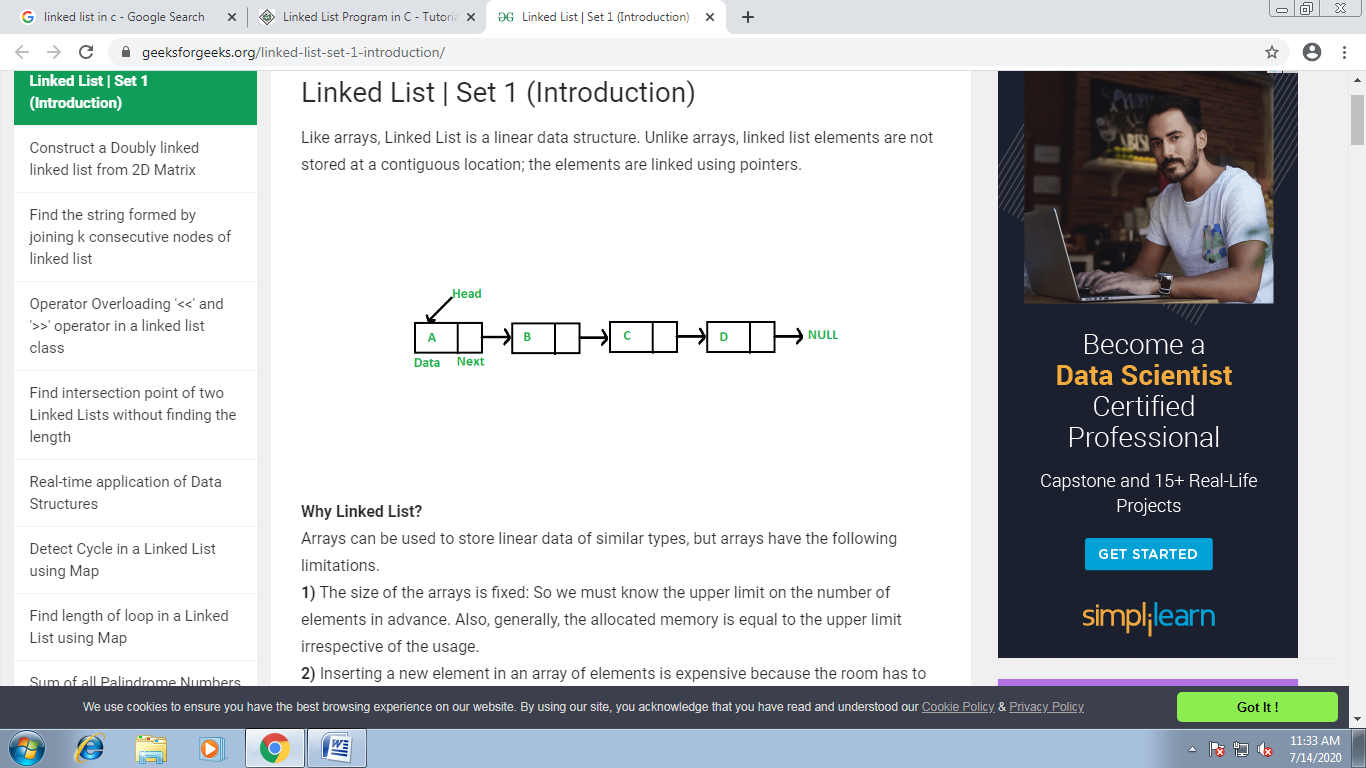
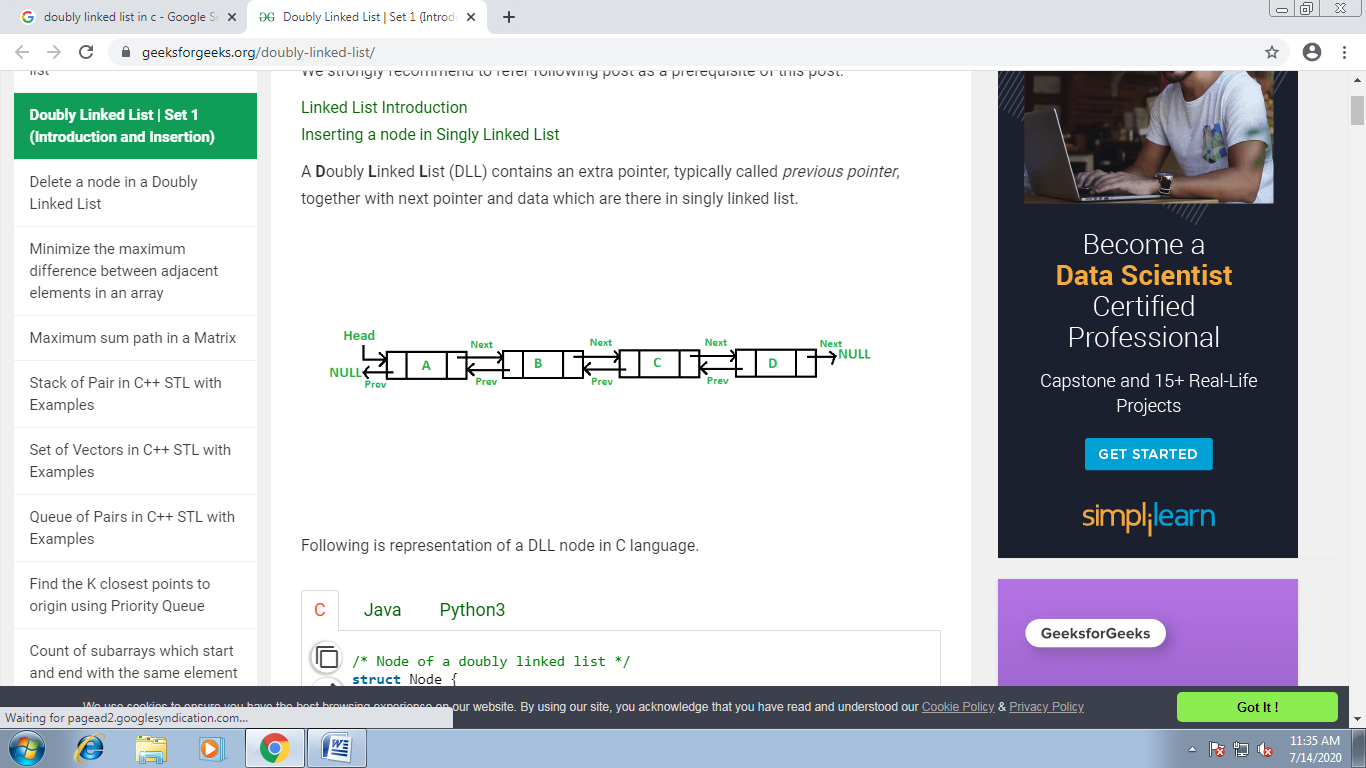
Linked List

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a 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 the 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 the room has to be created for the new elements and to create room existing elements have to be shifted.

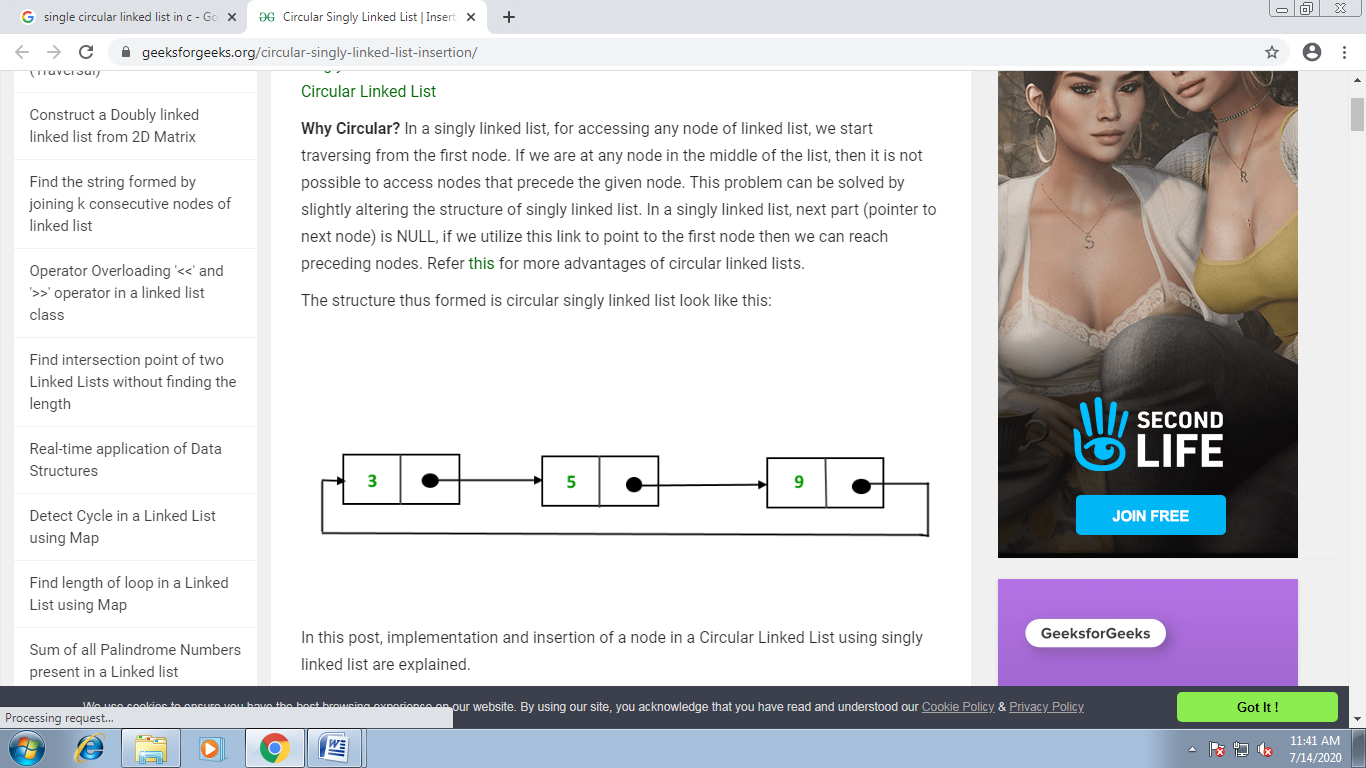
**Advantages over arrays**  
**1)** Dynamic size  
**2)** Ease of insertion/deletion

**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 efficiently with its default implementation.   
**2)** Extra memory space for a pointer is required with each element of the list.  
**3)** Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

**Representation:**  
A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head is NULL.  
Each node in a list consists of at least two parts:  
1) data  
2) Pointer (Or Reference) to the next node

**CIRCULAR LINKED LIST**

In a singly linked list, for accessing any node of linked list, we start traversing from the first node. If we are at any node in the middle of the list, then it is not possible to access nodes that precede the given node. This problem can be solved by slightly altering the structure of singly linked list. In a singly linked list, next part (pointer to next node) is NULL, if we utilize this link to point to the first node then we can reach preceding nodes.



**Program-**

#include<stdio.h>

#include<conio.h>

#include<alloc.h>

struct node

{

int info;

struct node \*next;

};

typedef struct node node;

node\* create();

node\* insbeg(node\*);

node\* insmid(node\*);

node\* insend(node\*);

node\* delbig(node\*);

node\* delmid(node\*);

node\* delend(node\*);

int count(node\*);

void sort(node\*);

void search(node\*);

void display(node\*);

void main()

{

int choice;

node \*head;

head=NULL;

while(1)

{

clrscr();

printf("\n\n\t\tThis is the program of CIRCULAR LINKED LIST");

printf("\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\n\n\n\t\tWhich operation you want to perform --");

printf("\n\n\t\t------------------------------------------");

printf(“\n\t\t 1. Create the list.”)

printf("\n\t\t 2. Insert element at the beginning OR");

printf("\n\t\t 3. Insert element at the middle OR");

printf("\n\t\t 4. Insert element at the end OR");

printf("\n\t\t 5. Delete the starting element OR");

printf("\n\t\t 6. Delete the element in between OR");

printf("\n\t\t 7. Delete the last element OR");

printf("\n\t\t 8. Count the number of elements OR");

printf("\n\t\t 9. Sort the elements of the list OR");

printf("\n\t\t 10. Search any particular element OR");

printf("\n\t\t 11. Display the list OR");

printf("\n\t\t 12. Wanna exit from the program.");

printf("\n\t\t------------------------------------------");

printf("\n\n\n\n\t\tEnter your choice = ");

scanf("%d",&choice);

switch(choice)

{

Case 1: head=create();

Continue;

case 2: head=insbeg(head);

continue;

case 3: head=insmid(head);

continue;

case 4: head=insend(head);

continue;

case 5: head=delbig(head);

continue;

case 6: head=delmid(head);

continue;

case 7: head=delend(head);

continue;

case 8: clrscr();

printf("\n\n\t\tThe list has %d elements.",count(head));

printf("\n\n\t\t\*\*Press any key to continue.\*\*");

getch();

continue;

case 9: sort(head);

continue;

case 10: search(head);

continue;

case 11:display(head);

continue;

case 12:printf("\n\n\n\n\t\t\*\*You have successfully terminated from the program.\*\*");

getch();

exit(0);

default:printf("\n\n\t\tYou entered a invalid choice!!!");

printf("\n\n\t\tPress any key to choose again.......");

getch();

}

}

}

node \* create()

{

node \*p, \*temp,\*head;

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

head->next=NULL;

printf(“\n\n\t Enter data to insert or -999 to stop”);

scanf(“%d”, &head->info);

p=head;

while(p->info!= -999)

{

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

temp->next=NULL;

scanf(“%d”,temp->info);

p->next=temp;

p=temp;

}

p->next=head;

return(head);

}

node\* insbeg(node \*head)

{

node \*ins,\*tmp;

clrscr();

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

printf("\n\n\t\tEnter the number to insert at the beginning = ");

fflush(stdin);

scanf("%d",&ins->info);

if(head==NULL)

{

head=ins;

head->next=head;

}

else

{

ins->next=head;

for(tmp=head;tmp->next!=head;tmp=tmp->next);

tmp->next=ins;

head=ins;

}

printf("\n\n\t\t\*\*Your number has been inserted successfully at beginning.");

getch();

return head;

}

node\* insmid(node \*head)

{

node \*tmp,\*ins;

int pos,cout,i;

cout=count(head);

while(1)

{

clrscr();

printf("\n\n\t\tAt which position you want to insert the element = ");

fflush(stdin);

scanf("%d",&pos);

if(pos>cout+1 || pos<1)

{

printf("\n\n\t\tYou entered a wrong position.");

printf("\n\n\t\tPress any key to choose any other position.");

getch();

continue;

}

else

{

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

printf("\n\n\n\n\t\tEnter the number to insert = ");

fflush(stdin);

scanf("%d",&ins->info);

if(pos==1)

{

ins->next=head;

for(tmp=head;tmp->next!=head;tmp=tmp->next);

tmp->next=ins;

head=ins;

}

else

{

tmp=head;

for(i=2;i<pos;i++,tmp=tmp->next);

ins->next=tmp->next;

tmp->next=ins;

}

return head;

}

}

}

node\* insend(node \*head)

{

node \*ins,\*tmp;

int num;

clrscr();

printf("\n\n\t\tEnter any number to insert at the end = ");

fflush(stdin);

scanf("%d",&num);

if(head==NULL)

{

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

head->info=num;

head->next=head;

}

else

{

for(tmp=head;tmp->next!=head;tmp=tmp->next);

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

ins->next=head;

tmp->next=ins;

ins->info=num;

}

printf("\n\n\t\t\*\*Your element has been successfully inserted at end.\*\*");

getch();

return head;

}

node\* delbig(node \*head)

{

node \*tmp;

clrscr();

if(head==NULL)

{

printf("\n\n\t\tUnderflow occur!!!");

printf("\n\n\t\tThere is no element in the list for deletion.");

getch();

return head;

}

else

if(head->next==head)

{

head=NULL;

printf("\n\n\t\t\*\*Starting element has been successfully deleted.\*\*");

getch();

return head;

}

else

{

for(tmp=head;tmp->next!=head;tmp=tmp->next);

tmp->next=head->next;

printf("\n\n\t\t\*\*Starting element has been successfully deleted.\*\*");

getch();

return head->next;

}

}

node\* delmid(node \*head)

{

node \*tmp;

int pos,cout;

char ch;

while(1)

{

clrscr();

if(head==NULL)

{

printf("\n\n\t\tUnderflow occur!!!");

printf("\n\n\t\tThere is no element in the list for deletion.");

getch();

}

else

{

printf("\n\n\tEnter the position of the element you wanna to delete =");

scanf("%d",&pos);

cout=count(head);

if(pos<1 || pos>cout)

{

printf("\n\n\t\tYou entered a wrong position.");

printf("\n\n\t\tPress any key to choose any other position");

printf("\n\n\t\tOR press 'N' to go back on the main menu.");

if((ch=getch())=='n' || ch=='N')

return head;

continue;

}

else

if(pos==1)

{

if(head->next==head)

head=NULL;

else

{

for(tmp=head;tmp->next!=head;tmp=tmp->next);

head=head->next;

tmp->next=head;

}

}

else

{

tmp=head;

for(cout=2;cout<pos;cout++,tmp=tmp->next);

tmp->next=tmp->next->next;

}

printf("\n\n\t\t\*\*Your element has been successfully deleted.\*\*");

getch();

}

return head;

}

}

node\* delend(node \*head)

{

node \*tmp;

clrscr();

tmp=head;

if(head==NULL)

{

printf("\n\n\t\tUnderflow occur!!!");

printf("\n\n\t\tThere is no element in the list for deletion.");

getch();

}

else

{

if(head->next==head)

head=NULL;

else

while(tmp->next!=head)

{

if(tmp->next->next==head)

{

tmp->next=head;

break;

}

tmp=tmp->next;

}

printf("\n\n\t\t\*\*Last element has been successfully deleted.\*\*");

}

getch();

return head;

}

void sort(node \*head)

{

node \*tmp1,\*tmp2,\*swap;

clrscr();

if(head==NULL)

{

printf("\n\n\t\t\*\*Lallu, sort karne k liye elements hone chahiye!!!\*\*");

getch();

return;

}

for(tmp1=head;tmp1->next!=head;tmp1=tmp1->next)

for(tmp2=tmp1->next;tmp2!=head;tmp2=tmp2->next)

if(tmp1->info > tmp2->info)

{

swap->info=tmp1->info;

tmp1->info=tmp2->info;

tmp2->info=swap->info;

}

printf("\n\n\t\t\*\*Your list has been sorted.\*\*");

getch();

}

void search(node \*head)

{

node \*tmp;

int item,flag=0;

tmp=head;

clrscr();

if(head==NULL)

{

printf("\n\n\t\tEmpty list ko search kar raha hai, akal nahi hai.");

printf("\n\n\t\tPlease inseart some elements first then try again.");

}

else

{

printf("\n\n\t\tEnter the item to search in the list = ");

scanf("%d",&item);

printf("\n\n\t\tYour result is as follows --\n");

do

{

if(tmp->info==item)

{

printf("\n\n\t\t[%d] -->found",tmp->info);

flag=1;

}

else

printf("\n\n\t\t%d",tmp->info);

tmp=tmp->next;

}

while(tmp!=head);

if(flag==0)

printf("\n\n\n\n\t\t\*\*Your item is not present in the whole list!!!\*\*");

else

printf("\n\n\n\n\t\t\*\*Your item found at the above locations.\*\*");

}

getch();

}

int count(node \*head)

{

int count;

node \*tmp;

tmp=head;

if(head==NULL)

count=0;

else

for(count=1;tmp->next!=head;count++,tmp=tmp->next);

return count;

}

void display(node \*head)

{

node \*tmp;

tmp=head;

clrscr();

if(head==NULL)

{

printf("\n\n\t\tThere is no element in the linked list.");

printf("\n\n\t\tPlease inseart some elements first then try again.");

}

else

{

printf("\n\n\t\tYour list is as follows --\n");

printf("\n\t\t%d",tmp->info);

while(tmp->next!=head)

{

tmp=tmp->next;

printf("\n\t\t%d",tmp->info);

}

}

getch();

}