PROLOG ACADEMY

DATA STRUCTURE

By- MASEERA ALI

Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

Split a Circular Linked List into two halves

Algorithm for this program is-

1) Store the mid and last pointers of the circular linked list using tortoise and hare algorithm.
2) Make the second half circular.
3) Make the first half circular.
4) Set head (or start) pointers of the two linked lists.

```
#include<stdio.h>
#include<stdlib.h>
struct link_list
{
        int info;
        struct link_list *link;
}*start=NULL,*NEW,*temp;
int enter()
{
        NEW=(struct link_list *)malloc(sizeof(struct link_list));
        printf("Enter the info");
        scanf("%d",&(NEW->info));
        if(start==NULL)
        {
                start=NEW;
                NEW->link=NEW;
        }
        else
        {
                temp=start;
                while(temp->link!=start)
                        temp=temp->link;
                NEW->link=start;
                temp->link=NEW;
        }
display(start);
return 0;
}
```

```
int display(struct link_list *start)
{       if(start==NULL)
         printf("No nodes to display");
         else
        {temp=start;
                do
                {
                        printf("%d\t",temp->info);
                        temp=temp->link;
                }
                while(temp!=start);
        }
        printf("\n");
        return 0;
}
```

```
int split()
{
	if(start==NULL)
		return 0;
	if(start->link==start)
	{
		printf("Only 1 node in linked list\n");
		return 0;
	}
	struct link_list *slowptr,*fastptr,*start2;
	fastptr=start;
	slowptr=start;
	while(fastptr->link!=start && fastptr->link->link!=start)
	{
		fastptr=fastptr->link->link;
		slowptr=slowptr->link;
	}
	if(slowptr==fastptr)
	{
		start2=fastptr->link;
		start->link=start;
		start2->link=start2;
	}
	else
	{
		start2=slowptr->link;
		slowptr->link=start;
		if(fastptr->link==start)
			fastptr->link=start2;
		else
			fastptr->link->link=start2;
	}
	printf("First -\n");
	display(start);
	printf("Second -\n");
	display(start2);
	return 0;
}
```

```
int main()
{       int n;
        while(1)
        {
                printf("\nWhat do you want to do\n1.enter\n2.split\n3.exit\n");
                fflush(stdin);
                scanf("%d",&n);

                switch(n)
                {
                        case 1:enter();
                                break;
                        case 2:split();
                                break;
                        case 3:exit(0);
                }
        }
return 0;
}
```

Double Linked list

Advantages over singly linked list
1) A DLL can be traversed in both forward and backward direction.
2) The delete operation in DLL is more efficient if pointer to the node to be deleted is given.
In singly linked list, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using left pointer.

Disadvantages over singly linked list
1) Every node of DLL Require extra space for an previous pointer. It is possible to implement DLL with single pointer though.
2) All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with next pointers.

```
#include<stdio.h>
#include<stdlib.h>
struct link_list
{
	int info;
	struct link_list *right,*left;
}*start=NULL,*end=NULL,*NEW,*temp;
int enter_beg()
{
	NEW=(struct link_list *)malloc(sizeof(struct 
link_list));
	printf("Enter the info");
	scanf("%d",&(NEW->info));
	if(start==NULL)
	{
		start=NEW;
		end=NEW;
		NEW->right=NULL;
		NEW->left=NULL;
	}
	else
	{
		NEW->right=start;
		start->left=NEW;
		NEW->left=NULL;
		start=NEW;
	}
display();
return 0;
}
```

```
int enter_last()
{
	NEW=(struct link_list *)malloc(sizeof(struct 
link_list));
	printf("Enter the info");
	scanf("%d",&(NEW->info));
	if(start==NULL)
	{
		start=NEW;
		end=NEW;
		NEW->right=NULL;
		NEW->left=NULL;
	}
	else
	{
		end->right=NEW;
		NEW->left=end;
		end=NEW;
		NEW->right=NULL;
	}
display();
return 0;
}
```

```
int enter_mid()
{       int n,c=1;
        temp=start;
        NEW=(struct link_list *)malloc(sizeof(struct
link_list));
        printf("Enter the node number ");
        scanf("%d",&n);
        while(c!=n)
        {  c++;
          temp=temp->right;
        }
        printf("Enter the info");
        scanf("%d",&(NEW->info));
        NEW->left=temp;
        NEW->right=temp->right;
        temp->right=NEW;
        NEW->right->left=NEW;
display();
return 0;
}
```

```
int display()
{       if(start==NULL)
         printf("No nodes to display");
         else
        {temp=start;
                while(temp!=NULL)
                {
                        printf("%d\t",temp->info);
                        temp=temp->right;
                }
        }
}
```

```c
int del()
{
        int n,flag;
        if(start==NULL)
        {
                printf("No nodes to delete\n");
                return 0;
        }
        printf("Enter the info you want to delete");
        scanf("%d",&n);
        temp=start;
        if(start==end)
        {
                start=NULL;
                end=NULL;
                free(temp);
                return 0;
        }
        if(temp->info==n)    //beginning
        {
                start=temp->right;
                start->left=NULL;
                free(temp);
                return 0;
        }
        while(temp->right!=NULL)  // middle
        {
                if(temp->info==n)
                {
                        flag=1;
                        break;
                }
                temp=temp->right;
        }
        if(flag==1)
        {
                temp->left->right=temp->right;
                temp->right->left=temp->left;
                free(temp);
        }
        else
        {
                if(temp->info==n)         //last
                {
                        temp->left->right=NULL;
                        end=temp->left;
                        free(temp);
                }

                else
                        printf("No such node exists\n");
        }
        return 0;
}
```

```
int len(){
        int c=0;
        temp=start;
        while(temp!=NULL)
        {
                temp=temp->right;
                c++;
        }
        printf("No of nodes - %d\n",c );
        return 0;
}
```

```
int find(){
        int c=1,n;
        if(start==NULL)
        {
                printf("Empty link_list\n");
                return 0;
        }
        temp=start;
        printf("Enter the info");
        scanf("%d",&n);
        printf("Location of %d -\n",n);
        while(temp!=NULL)
        {
                if(temp->info==n)
                {
                        printf("%d\t",c);
                }
                temp=temp->right;
                c++;
        }
        return 0;
}
```

```
int main()
{       int n;
        while(1)
        {
                printf("\nWhat do you want to do\n1.enter at beginning\n2.enter at mid\n3.enter at 
last\n4.display\n5.Delete\n6.length\n7.Find\n8.exit\n");
                fflush(stdin);
                scanf("%d",&n);

                switch(n)
                {
                        case 1:enter_beg();
                                break;
                        case 2:enter_mid();
                                break;
                        case 3:enter_last();
                                break;
                        case 4:display();
                                break;
                        case 5:del();
                                        display();
                                break;
                        case 6:len();
                                break;
                        case 7:find();
                                break;
                        case 8:exit(0);
                }
        }
return 0;
}
```

Reverse a Linked List

Steps to be followed-

1. Swap the left and right pointer of the node.
2. Swap the start and end pointer of the linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct link_list
{
        int info;
        struct link_list *right,*left;
}*start=NULL,*end=NULL,*NEW,*temp;
int enter()
{
        NEW=(struct link_list *)malloc(sizeof(struct
link_list));
        printf("Enter the info");
        scanf("%d",&(NEW->info));
        if(start==NULL)
        {
                start=NEW;
                end=NEW;
                NEW->right=NULL;
                NEW->left=NULL;
        }
        else
        {
                end->right=NEW;
                NEW->left=end;
                end=NEW;
                NEW->right=NULL;
        }
display();
return 0;
}
```

```
int display()
{       if(start==NULL)
        printf("No nodes to display");
        else
       {temp=start;
                while(temp!=NULL)
                {
                        printf("%d\t",temp->info);
                        temp=temp->right;
                }
       }
}
```

```
int reverse()
{
        struct link_list *value;
        temp=start;
        while(temp!=NULL)
        {
                value=temp->left;
                temp->left=temp->right;
                temp->right=value;
                temp=temp->left;
        }
        value=start;
        start=end;
        end=value;
        display();
}
```

```
int main()
{       int n;
        while(1)
        {
                printf("\nWhat do you want to 
do\n1.enter\n2.display\n3.reverse\n4.exit\n");
                fflush(stdin);
                scanf("%d",&n);

                switch(n)
                {
                        case 1:enter();
                                break;
                        case 2:display();
                                break;
                        case 3:reverse();
                                break;
                        case 4:exit(0);
                }
        }
return 0;
}
```