Program for implementation of linked list

ALGORITHM:

Step 1: IF ptr = NULL

Write OVERFLOW

Go to Step 9

[END OF IF]

Step 2: SET NEW\_NODE = ptr

Step 3: SET ptr = ptr -> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET NEW\_NODE -> PREV = NULL

Step 6: SET NEW\_NODE -> NEXT = START

Step 7: SET head -> PREV = NEW\_NODE

Step 8: SET head = NEW\_NODE

Step 9: EXIT

PROGRAM:

//Implementation of doubly linked list

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*prev;

struct node \*next;

int data;

};

struct node \*head;

void insert\_begin()

{

struct node \*ptr;

int item;

ptr=(struct node \*)malloc(sizeof(struct node));

if (ptr==NULL)

{

printf("Overflow\n");

}

else

{

printf("Enter item value:\n");

scanf("%d",&item);

if(head==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;

ptr->next=head;

head->prev=ptr;

head=ptr;

}

printf("Node Inserted\n");

}

}

void insert\_last()

{

struct node \*ptr,\*temp;

int item;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow\n");

}

else

{

printf("Enter item value:\n");

scanf("%d",&item);

if(head==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=ptr;

ptr->prev=temp;

ptr->next=NULL;

}

} \

}

void insert\_specified()

{

struct node \*ptr,\*temp;

int item,loc,i;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow\n");

}

else

{

temp=head;

printf("Enter the location:\n");

scanf("%d",&loc);

for(i=0;i<loc;i++)

{

temp=temp->next;

if(temp==NULL)

{

printf("Cant find Loc\n");

return;

}

}

printf("Enter value:");

scanf("%d",&item);

ptr->data=item;

ptr->next=temp->next;

ptr->prev=temp;

temp->next=ptr;

temp->next->prev=ptr;

printf("Node inserted\n");

}

}

void delete\_begin()

{

struct node \*ptr;

if(head==NULL)

{

printf("Underflow");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("Node Deleted");

}

else

{

ptr=head;

head=head->next;

head->prev=NULL;

free(ptr);

printf("Node deleted\n");

}

}

void delete\_last()

{

struct node \*ptr;

if(head=NULL)

{

printf("Underflow\n");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("Node deleted\n");

}

else

{

ptr=head;

if(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->prev->next=NULL;

free(ptr);

printf("Node deleted\n");

}

}

void delete\_specified()

{

struct node \*ptr,\*temp;

int val;

printf("Enter the data after which the node is to be deleted\n");

scanf("%d",&val);

ptr=head;

while(ptr->data!=val)

ptr=ptr->next;

if(ptr->next==NULL)

{

printf("Cant delete");

}

else if(ptr->next->next==NULL)

{

ptr->next=NULL;

}

else

{

temp=ptr->next;

ptr->next=temp->next;

temp->next->prev=ptr;

free(temp);

printf("Node deleted");

}

}

void display()

{

if(head==NULL)

{

printf("Linked List is Empty\n");

return;

}

printf("LinkedList: ");

struct node\* ptr = head;

while(ptr!=NULL) // start from first node

{

printf("%d ",ptr->data);

ptr = ptr->next;

}

printf("\n");

}

int main()

{

int choice;

while(1)

{

printf("Operation performed by Linked Lists:\n");

printf("1.Insert at Begin\n2.Insert at Last\n3.Insert at Random\n4.Delete at Begin\n5.Delete at Last\n6.Delete at random\n7.Display\n8.Exit\n");

printf("Enter Your Choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1: insert\_begin();

break;

case 2: insert\_last();

break;

case 3: insert\_specified();

break;

case 4: delete\_begin();

break;

case 5: delete\_last();

break;

case 6: delete\_specified();

break;

case 7: display();

break;

case 8: exit(0);

default: printf("Invalid Choice\n");

}

}

}

OUTPUT:







