AIM:- Doubly linked list

Algorithm:-

Step 1: if ptr =NULL

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

Program: -

#include<stdio.h>//standard input output header file//

#include<stdlib.h>//standard library header file//

struct node//initializing the struct data type//

{

struct node \*prev;//declaring the prev//

struct node \*next;//declaring the next//

int data;// //declaring data//

};

struct node \*head;

void insertion\_beginning();//insert at beginning function//

void insertion\_last();//insert at last function//

void insertion\_specified();//insert at random function//

void deletion\_beginning();//deletion at beginning function//

void deletion\_last();//deletion at last function//

void deletion\_specified();//deletion at random function//

void display();//display function//

void search();//search function//

void main ()//main function//

{

int choice =0;//declaring the choice//

while(choice != 9)//while loop//

{

printf("\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");//prints the statement//

printf("\nChoose one option from the following list ...\n");//prints the statement//

printf("\n===============================================\n");//prints the statement//

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n5.Delete from last\n6.Delete the node after the given data\n7.Search\n8.Show\n9.Exit\n");//prints the statement//

printf("\nEnter your choice?\n");//prints the statement//

scanf("\n%d",&choice);//memory slot will be initialized//

switch(choice)//switch loop//

{

case 1:

insertion\_beginning();//insert function//

break;//terminates the switch loop//

case 2:

insertion\_last();//insert function//

break;//terminates the switch loop//

case 3:

insertion\_specified();//insert function//

break;//terminates the switch loop//

case 4:

deletion\_beginning();//delete function//

break; //terminates the switch loop//

case 5:

deletion\_last();//delete function//

break; //terminates the switch loop//

case 6:

deletion\_specified();//delete function//

break; //terminates the switch loop//

case 7:

search();//search function//

break; //terminates the switch loop//

case 8:

display();//display function//

break; //terminates the switch loop//

case 9:

exit(0);

break; //terminates the switch loop//

default:

printf("Please enter valid choice..");//prints the statement//

}

}

}

void insertion\_beginning()//insert at start function//

{

struct node \*ptr;

int item;

ptr = (struct node \*)malloc(sizeof(struct node));//dynamic memory allocation//

if(ptr == NULL)//if statement//

{

printf("\nOVERFLOW");//prints the statement//

}

else

{

printf("\nEnter Item value");//prints the statement//

scanf("%d",&item);//memory slot will be initialized//

if(head==NULL)//if statement//

{

ptr->next = NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;

ptr->prev=NULL;

ptr->next = head;

head->prev=ptr;

head=ptr;

}

printf("\nNode inserted\n");//prints the statement//

}

}

void insertion\_last()

{

struct node \*ptr,\*temp;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value");

scanf("%d",&item);

ptr->data=item;

if(head == NULL)

{

ptr->next = NULL;

ptr->prev = NULL;

head = ptr;

}

else

{

temp = head;

while(temp->next!=NULL)

{

temp = temp->next;

}

temp->next = ptr;

ptr ->prev=temp;

ptr->next = NULL;

}

}

printf("\nnode inserted\n");//prints the statement//

}

void insertion\_specified()

{

struct node \*ptr,\*temp;

int item,loc,i;

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

if(ptr == NULL)

{

printf("\n OVERFLOW");//prints the statement//

}

else

{

temp=head;

printf("Enter the location");//prints the statement//

scanf("%d",&loc);

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

{

temp = temp->next;

if(temp == NULL)

{

printf("\n There are less than %d elements", loc); //prints the statement//

return;

}

}

printf("Enter value");//prints the statement//

scanf("%d",&item);

ptr->data = item;

ptr->next = temp->next;

ptr -> prev = temp;

temp->next = ptr;

temp->next->prev=ptr;

printf("\nnode inserted\n");//prints the statement//

}

}

void deletion\_beginning()

{

struct node \*ptr;

if(head == NULL)

{

printf("\n UNDERFLOW");//prints the statement//

}

else if(head->next == NULL)

{

head = NULL;

free(head);

printf("\nnode deleted\n");//prints the statement//

}

else

{

ptr = head;

head = head -> next;

head -> prev = NULL;

free(ptr);

printf("\nnode deleted\n");//prints the statement//

}

}

void deletion\_last()

{

struct node \*ptr;

if(head == NULL)

{

printf("\n UNDERFLOW");//prints the statement//

}

else if(head->next == NULL)

{

head = NULL;

free(head);

printf("\nnode deleted\n");//prints the statement//

}

else

{

ptr = head;

while(ptr->next != NULL)

{

ptr = ptr -> next;

}

ptr -> prev -> next = NULL;

free(ptr);

printf("\nnode deleted\n");//prints the statement//

}

}

void deletion\_specified()

{

struct node \*ptr, \*temp;

int val;

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

scanf("%d", &val);

ptr = head;

while(ptr -> data != val)

ptr = ptr -> next;

if(ptr -> next == NULL)

{

printf("\nCan't delete\n");//prints the statement//

}

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

{

ptr ->next = NULL;

}

else

{

temp = ptr -> next;

ptr -> next = temp -> next;

temp -> next -> prev = ptr;

free(temp);//deallocates the memory//

printf("\nnode deleted\n");//prints the statement//

}

}

void display()

{

struct node \*ptr;

printf("\n printing values...\n");//prints the statement//

ptr = head;

while(ptr != NULL)

{

printf("%d\n",ptr->data);//prints the statement//

ptr=ptr->next;

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");//prints the statement//

}

else

{

printf("\nEnter item which you want to search?\n");//prints the statement//

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("\nitem found at location %d ",i+1); //prints the statement//

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

printf("\nItem not found\n");//prints the statement//

}

}

}

Screenshot of the output:-





















