**Write a program to implement operations on Singly Linked List such as node creation, traversing, searching, insertion, deletion.**

//Singly Linked List Operations

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

struct list {

int info;

struct list \*next;

};

struct list \*ptr, \*p, \*head;

void cre();

void bins();

void eins();

void mins();

void bdlt();

void edlt();

void mdlt();

void trv();

void srch();

void main()

{

int n;

clrscr();

do

{

printf(" \n\n LINK LIST OPERATIONS:\n");

printf(" 1. CREATION OF NODES:\n");

printf(" 2. TRAVERSING:\n");

printf(" 3. SEARCHING:\n");

printf(" 4. INSERTION OF NODE IN BEGINNING:\n");

printf(" 5. INSERTION OF NODE AT A LOCATION:\n");

printf(" 6. INSERTION OF NODE IN THE END:\n");

printf(" 7. DELETION OF NODE IN THE BEGINNING:\n");

printf(" 8. DELETION OF NODE AT A LOCATION:\n");

printf(" 9. DELETION OF NODE IN THE END:\n");

printf(" 0. EXIT:\n");

printf("\n Enter a choice: ");

scanf("%d",&n);

switch(n)

{

case 0: exit(0);

case 1: cre();

break;

case 2: trv();

break;

case 3: srch();

break;

case 4: bins();

break;

case 5: mins();

break;

case 6: eins();

break;

case 7: bdlt();

break;

case 8: mdlt();

break;

case 9: edlt();

break;

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

}

} while(1);

getch();

}

void cre()

{

char ch;

if(head == NULL)

{

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

head = ptr;

printf("\n Enter a number: ");

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

}

else

{

do

{

p = (struct list \*)malloc(sizeof(struct list \*));

printf("\n Enter a number: ");

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

ptr->next = p;

ptr = p;

printf("\n Do you want to create more nodes: ");

ch = getche();

} while (ch=='y' || ch=='Y');

}

ptr->next = NULL;

}

void trv()

{

ptr = head;

printf("\n");

while (ptr != NULL)

{

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

ptr = ptr->next;

}

}

void srch()

{

int n,temp=0,i=1;

if(head == NULL)

printf("\n List does not exist");

else

{

ptr=head;

printf("\n Enter the number to search in the list: ");

scanf("%d",&n);

while(ptr != NULL)

{

if(ptr->info == n)

{

printf("\n Number found at loc %d \n", i++);

ptr = ptr->next;

temp = 1;

}

i++;

ptr=ptr->next;

}

if(temp == 0)

printf("\n Number does not exist in the list: ");

}

}

void bins()

{

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

p = head;

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

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

head = ptr;

head->next = p;

}

void mins()

{

int loc, i;

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

scanf("%d", &loc);

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

p = head;

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

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

for (i=0; i<loc-1; i++)

{

p = p->next;

printf(" %d-->", p->info);

}

ptr->next = p->next;

p->next = ptr;

}

void eins()

{

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

p = head;

while (p->next != NULL)

{

p = p->next;

}

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

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

p->next = ptr;

ptr->next = NULL;

return;

}

void bdlt()

{

p = head;

head = p->next;

free(p);

printf("\n Element deleted!");

}

void mdlt()

{

int loc, i;

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

scanf("%d", &loc);

p = head;

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

{

ptr = p;

p = p->next;

if (p == NULL)

{

printf("\n There are elements less than %d!", loc);

return;

}

}

ptr->next = p->next;

printf("\n Element deleted!");

free(p);

}

void edlt()

{

if (head->next == NULL)

free(head);

else

{

p = head;

while (p->next != NULL)

{

ptr = p;

p = p->next;

}

ptr->next = NULL;

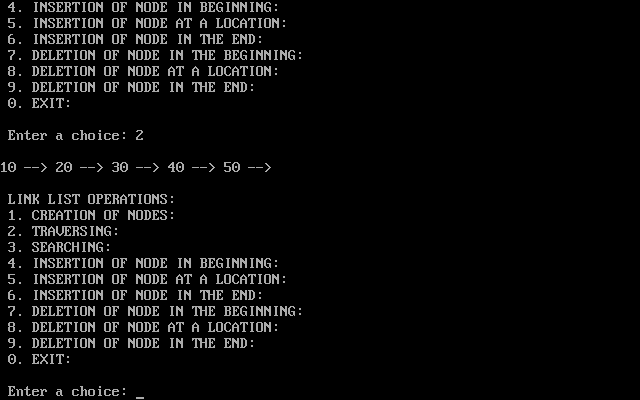
free(p);

}

printf("\n Element deleted!");

}

**OUTPUT:**



**Write a program to implement operations on Circular Linked List such as traversing, searching, insertion, deletion.**

//Circular Linked List Operations

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

};

struct node \*head = NULL, \*ptr, \*p, \*tail = NULL;

void srch();

void trv();

void insb();

void inse();

void delb();

void dele();

void main()

{

int n;

clrscr();

do

{

printf(" \n\n CIRCULAR LINK LIST OPERATIONS:\n");

printf(" 1. TRAVERSING:\n");

printf(" 2. SEARCHING:\n");

printf(" 3. INSERTION OF NODE IN BEGINNING:\n");

printf(" 4. INSERTION OF NODE IN THE END:\n");

printf(" 5. DELETION OF NODE IN THE BEGINNING:\n");

printf(" 6. DELETION OF NODE IN THE END:\n");

printf(" 0. EXIT:\n");

printf("\n Enter a choice: ");

scanf("%d",&n);

switch(n)

{

case 0: exit(0);

case 1: trv();

break;

case 2: srch();

break;

case 3: insb();

break;

case 4: inse();

break;

case 5: delb();

break;

case 6: dele();

break;

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

}

} while(1);

getch();

}

void insb()

{

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

if (head == NULL)

{

head = newn;

tail = newn;

newn->next = head;

printf("\n Enter a number: ");

scanf("%d", &newn->data);

}

else

{

p = head;

printf("\n Enter a number: ");

scanf("%d", &newn->data);

newn->next = p;

head = newn;

tail->next = head;

}

}

void inse()

{

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

if (head == NULL)

{

head = newn;

tail = newn;

newn->next = head;

printf("\n Enter a number: ");

scanf("%d", &newn->data);

}

else

{

printf("\n Enter a number: ");

scanf("%d", &newn->data);

tail->next = newn;

tail = newn;

tail->next = head;

}

}

void delb()

{

if (head == NULL)

{

return;

}

else

{

if (head != tail)

{

head = head->next;

tail->next = head;

}

else

head = tail = NULL;

}

printf("\n Element deleted!");

}

void dele()

{

if (head == NULL)

{

return;

}

else

{

if (head != tail)

{

struct node \*p = head;

while (p->next != tail)

{

p = p->next;

}

tail = p;

tail->next = head;

}

else

head = tail = NULL;

}

printf("\n Element deleted!");

}

void srch()

{

int ele;

struct node \*ptr = head;

printf("\n Enter the element to search: ");

scanf("%d", &ele);

int i = 1;

int flag = 0;

if (head == NULL)

{

printf("\n List is empty!");

}

else

{

do

{

if (ele == ptr->data)

{

flag++;

break;

}

ptr = ptr->next;

i++;

} while (ptr != head);

if (flag)

printf("\n Element found at position %d", i);

else

printf("\n Element not present in the list!");

}

}

void trv()

{

struct node \*ptr = head;

if (head == NULL)

{

printf("\n List is empty!");

}

else

{

do

{

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

ptr = ptr->next;

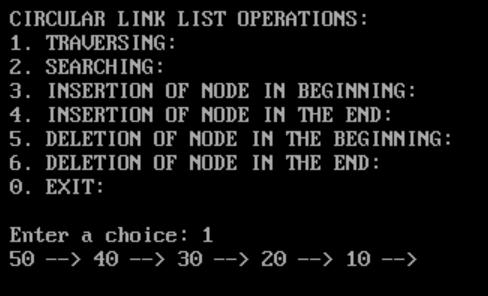
} while (ptr != head);

}

printf("\n");

}

**OUTPUT:**

****

**Write a program to implement operations on Doubly Linked List such as traversing, searching, insertion, deletion.**

//Doubly Linked List Operations

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*prev;

struct node \*next;

};

struct node \*head, \*tail = NULL, \*ptr, \*p;

void bins();

void eins();

void mins();

void bdlt();

void edlt();

void mdlt();

void trv();

void srch();

void main()

{

int n;

clrscr();

do

{

printf(" \n\n DOUBLY LINK LIST OPERATIONS:\n");

printf(" 1. TRAVERSING:\n");

printf(" 2. SEARCHING:\n");

printf(" 3. INSERTION OF NODE IN BEGINNING:\n");

printf(" 4. INSERTION OF NODE AT A LOCATION:\n");

printf(" 5. INSERTION OF NODE IN THE END:\n");

printf(" 6. DELETION OF NODE IN THE BEGINNING:\n");

printf(" 7. DELETION OF NODE AT A LOCATION:\n");

printf(" 8. DELETION OF NODE IN THE END:\n");

printf(" 0. EXIT:\n");

printf("\n Enter a choice: ");

scanf("%d",&n);

switch(n)

{

case 0: exit(0);

case 1: trv();

break;

case 2: srch();

break;

case 3: bins();

break;

case 4: mins();

break;

case 5: eins();

break;

case 6: bdlt();

break;

case 7: mdlt();

break;

case 8: edlt();

break;

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

}

} while(1);

getch();

}

void bins()

{

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

printf("\n Enter a number: ");

scanf("%d", &ptr->data);

if (head == NULL)

{

head = tail = ptr;

head->prev = NULL;

tail->next = NULL;

}

else

{

head->prev = ptr;

ptr->next = head;

ptr->prev = NULL;

head = ptr;

}

}

void eins()

{

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

printf("\n Enter a number: ");

scanf("%d", &ptr->data);

if (head == NULL)

{

head = tail = ptr;

head->prev = NULL;

tail->next = NULL;

}

else

{

tail->next = ptr;

ptr->prev = tail;

tail = ptr;

tail->next = NULL;

}

}

void mins()

{

int loc,i;

printf("\n Enter the location to insert element: ");

scanf("%d",&loc);

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

p = head;

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

scanf("%d", &ptr->data);

for (i=1; i<loc-1; i++)

{

p = p->next;

if (p == NULL)

{

printf("\nThere are lesser elements than %d.",loc);

return;

}

}

ptr->next = p->next;

ptr->prev = p;

p->next = ptr;

p->prev = ptr;

}

void bdlt()

{

p = head;

head = p->next;

head->prev = NULL;

free(p);

printf("\n Element deleted!");

}

void edlt()

{

if (head->next == NULL)

free (head);

else

{

p = head;

while (p->next != NULL)

{

p = p->next;

}

p->prev->next = NULL;

free(p);

}

printf("\n Element deleted!");

}

void mdlt()

{

int loc,i;

printf("\n Enter the location to delete element: ");

scanf("%d",&loc);

if (head == NULL)

{

return;

}

else

{

struct node \*p = head;

for (i=1; i<loc-1; i++)

{

p = p->next;

}

if (p == head)

{

head = p->next;

}

else if (p == tail)

{

tail = tail->next;

}

else

{

p->prev->next = p->next;

p->next->prev = p->prev;

}

p = NULL;

}

printf("\n Element deleted!");

}

void trv()

{

struct node \*ptr = head;

if (head == NULL)

{

printf("\n List is empty!");

return;

}

while (ptr != NULL)

{

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

ptr = ptr->next;

}

printf("\n");

}

void srch()

{

int ele,flag=0;

printf("\n Enter element to be searched:");

scanf("%d",&ele);

ptr = head;

while (ptr != NULL)

{

if(ele == ptr->data)

{

flag = 1;

printf("\n Element is present in the List.");

break;

}

else

{}

ptr=ptr -> next;

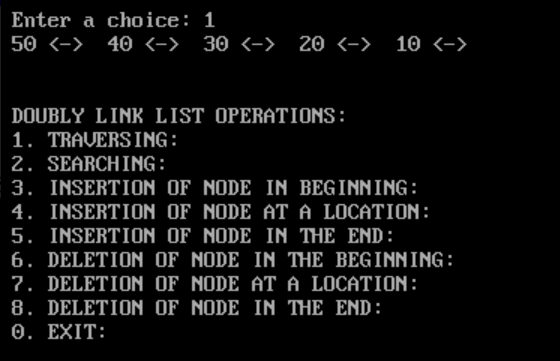
}

if (flag == 0)

printf("\n Element is not present in the List.");

}

**OUTPUT:**

****

**Write a program to implement operations on Circular Doubly Linked List such as insertion and deletion in beginning and end.**

//Circular Doubly Linked List Operations

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void iend();

void istart();

void delbeg();

void delend();

void traverse();

void search();

struct node

{

int info;

struct node \* next;

struct node \* prev;

};

struct node \*ptr;

struct node \*p;

struct node \*head = NULL;

void iend()

{

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

clrscr();

p = head;

while (p->next != head)

{

p = p->next;

}

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

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

p->next = ptr;

ptr->prev = p;

head->prev = ptr;

ptr->next = head;

return;

}

void istart()

{

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

p = head;

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

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

while (p->next != head)

p = p->next;

ptr->next = head;

head->prev = ptr;

ptr->prev = p;

p->next = ptr;

head = ptr;

}

void delbeg()

{

p = head;

while (p->next != head)

p = p->next;

p->next = head->next;

head->next->prev = p;

free(head);

head = p->next;

printf("\n Element deleted!");

}

void delend()

{

if (head->next == head)

free(head);

else

{

p = head;

while(p->next != head)

{

p = p->next;

}

p->prev->next = head;

head->prev = p->prev;

free(p);

}

printf("\n Element deleted!");

}

void traverse()

{

ptr = head;

printf("\n Elements in list are:\n");

while (ptr->next != head)

{

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

ptr = ptr->next;

}

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

}

void search()

{

int ele, flag = 0;

printf("\n Enter element to search: ");

scanf("%d", &ele);

ptr = head;

while (ptr->next != head)

{

if( ele == ptr->info)

{

flag = 1;

printf("\n Element is present in the List.");

break;

}

else

{}

ptr = ptr->next;

}

if (flag == 0)

{

if(ptr->info == ele)

printf("\n Element is present in the List.");

else

printf("\n Element is not present in the List.");

}

}

void main()

{

int c;

clrscr();

do

{

if (head == NULL)

{

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

printf("\n Enter 1st Element of the Linked List: ");

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

head = ptr;

ptr->next = head;

head->prev = head;

}

else

{

printf("\n 1.Insert Element at the end of the list: ");

printf("\n 2.Insert Element at the beginning of the list: ");

printf("\n 3.Traverse the list: ");

printf("\n 4.Delete Element from the Beginning: ");

printf("\n 5.Delete Element from End: ");

printf("\n 6.Search for an element: ");

printf("\n 0.EXIT: ");

printf("\n Enter Your Choice:");

scanf("%d", &c);

switch(c)

{

case 0: exit(0);

case 1: iend();

break;

case 2: istart();

break;

case 3: traverse();

break;

case 4: traverse();

delbeg();

printf("\n After deleting 1st Element");

traverse();

break;

case 5: traverse();

delend();

printf("\n After deleting Last Element");

traverse();

break;

case 6: traverse();

search();

break;

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

}

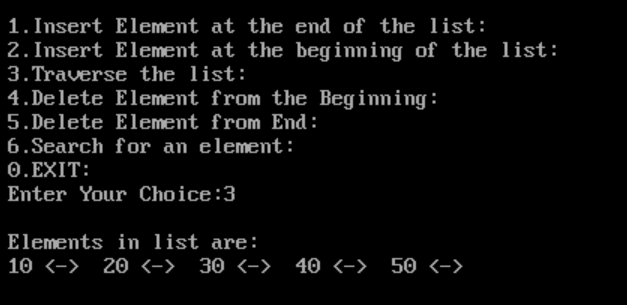
}

} while (1);

getch();

}

**OUTPUT:**

****