**Experement-3**

1) Design a data structure singly linked list with necessary members. Also implement operations on linked list (insert at first, insert at last, insert in between and delete from first, last and between, search, display).

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void insertAtBeginning(int);

void insertAtEnd(int);

void insertBetween(int,int,int);

void display();

void removeBeginning();

void removeEnd();

void removeSpecific(int);

struct Node

{

int data;

struct Node \*next;

}\*head = NULL;

void main()

{

int choice,value,choice1,loc1,loc2;

clrscr();

while(1){

mainMenu: printf("\n\n\*\*\*\*\*\* MENU \*\*\*\*\*\*\n1. Insert\n2. Display\n3. Delete\n4. Exit\nEnter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("Enter the value to be insert: ");

scanf("%d",&value);

while(1){

printf("Where you want to insert: \n1. At Beginning\n2. At End\n3. Between\nEnter your choice: ");

scanf("%d",&choice1);

switch(choice1)

{

case 1: insertAtBeginning(value);

break;

case 2: insertAtEnd(value);

break;

case 3: printf("Enter the two values where you wanto insert: ");

scanf("%d%d",&loc1,&loc2);

insertBetween(value,loc1,loc2);

break;

default: printf("\nWrong Input!! Try again!!!\n\n");

goto mainMenu;

}

goto subMenuEnd;

}

subMenuEnd:

break;

case 2: display();

break;

case 3: printf("How do you want to Delete: \n1. From Beginning\n2. From End\n3. Spesific\nEnter your choice: ");

scanf("%d",&choice1);

switch(choice1)

{

case 1: removeBeginning();

break;

case 2: removeEnd();

break;

case 3: printf("Enter the value which you wanto delete: ");

scanf("%d",&loc2);

removeSpecific(loc2);

break;

default: printf("\nWrong Input!! Try again!!!\n\n");

goto mainMenu;

}

break;

case 4: exit(0);

default: printf("\nWrong input!!! Try again!!\n\n");

}

}

}

void insertAtBeginning(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

if(head == NULL)

{

newNode->next = NULL;

head = newNode;

}

else

{

newNode->next = head;

head = newNode;

}

printf("\nOne node inserted!!!\n");

}

void insertAtEnd(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = NULL;

if(head == NULL)

head = newNode;

else

{

struct Node \*temp = head;

while(temp->next != NULL)

temp = temp->next;

temp->next = newNode;

}

printf("\nOne node inserted!!!\n");

}

void insertBetween(int value, int loc1, int loc2)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

if(head == NULL)

{

newNode->next = NULL;

head = newNode;

}

else

{

struct Node \*temp = head;

while(temp->data != loc1 && temp->data != loc2)

temp = temp->next;

newNode->next = temp->next;

temp->next = newNode;

}

printf("\nOne node inserted!!!\n");

}

void removeBeginning()

{

if(head == NULL)

printf("\n\nList is Empty!!!");

else

{

struct Node \*temp = head;

if(head->next == NULL)

{

head = NULL;

free(temp);

}

else

{

head = temp->next;

free(temp);

printf("\nOne node deleted!!!\n\n");

}

}

}

void removeEnd()

{

if(head == NULL)

{

printf("\nList is Empty!!!\n");

}

else

{

struct Node \*temp1 = head,\*temp2;

if(head->next == NULL)

head = NULL;

else

{

while(temp1->next != NULL)

{

temp2 = temp1;

temp1 = temp1->next;

}

temp2->next = NULL;

}

free(temp1);

printf("\nOne node deleted!!!\n\n");

}

}

void removeSpecific(int delValue)

{

struct Node \*temp1 = head, \*temp2;

while(temp1->data != delValue)

{

if(temp1 -> next == NULL){

printf("\nGiven node not found in the list!!!");

goto functionEnd;

}

temp2 = temp1;

temp1 = temp1 -> next;

}

temp2 -> next = temp1 -> next;

free(temp1);

printf("\nOne node deleted!!!\n\n");

functionEnd:

}

void display()

{

if(head == NULL)

{

printf("\nList is Empty\n");

}

else

{

struct Node \*temp = head;

printf("\n\nList elements are - \n");

while(temp->next != NULL)

{

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

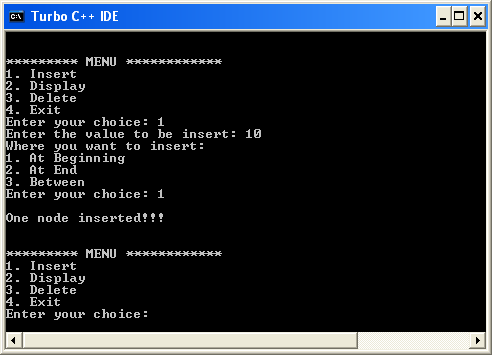
temp = temp->next;

}

printf("%d --->NULL",temp->data);

}

}



2) Develop a data structure circular singly linked list with necessary members. Also implement operations on circular linked list (insert at first, insert at last, insert in between and delete from first, last and between, search, display).

#include<stdio.h>

#include<conio.h>

void insertAtBeginning(int);

void insertAtEnd(int);

void insertAtAfter(int,int);

void deleteBeginning();

void deleteEnd();

void deleteSpecific(int);

void display();

struct Node

{

int data;

struct Node \*next;

}\*head = NULL;

void main()

{

int choice1, choice2, value, location;

clrscr();

while(1)

{

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

printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");

scanf("%d",&choice1);

switch()

{

case 1: printf("Enter the value to be inserted: ");

scanf("%d",&value);

while(1)

{

printf("\nSelect from the following Inserting options\n");

printf("1. At Beginning\n2. At End\n3. After a Node\n4. Cancel\nEnter your choice: ");

scanf("%d",&choice2);

switch(choice2)

{

case 1: insertAtBeginning(value);

break;

case 2: insertAtEnd(value);

break;

case 3: printf("Enter the location after which you want to insert: ");

scanf("%d",&location);

insertAfter(value,location);

break;

case 4: goto EndSwitch;

default: printf("\nPlease select correct Inserting option!!!\n");

}

}

case 2: while(1)

{

printf("\nSelect from the following Deleting options\n");

printf("1. At Beginning\n2. At End\n3. Specific Node\n4. Cancel\nEnter your choice: ");

scanf("%d",&choice2);

switch(choice2)

{

case 1: deleteBeginning();

break;

case 2: deleteEnd();

break;

case 3: printf("Enter the Node value to be deleted: ");

scanf("%d",&location);

deleteSpecic(location);

break;

case 4: goto EndSwitch;

default: printf("\nPlease select correct Deleting option!!!\n");

}

}

EndSwitch: break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nPlease select correct option!!!");

}

}

}

void insertAtBeginning(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode -> data = value;

if(head == NULL)

{

head = newNode;

newNode -> next = head;

}

else

{

struct Node \*temp = head;

while(temp -> next != head)

temp = temp -> next;

newNode -> next = head;

head = newNode;

temp -> next = head;

}

printf("\nInsertion success!!!");

}

void insertAtEnd(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode -> data = value;

if(head == NULL)

{

head = newNode;

newNode -> next = head;

}

else

{

struct Node \*temp = head;

while(temp -> next != head)

temp = temp -> next;

temp -> next = newNode;

newNode -> next = head;

}

printf("\nInsertion success!!!");

}

void insertAfter(int value, int location)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode -> data = value;

if(head == NULL)

{

head = newNode;

newNode -> next = head;

}

else

{

struct Node \*temp = head;

while(temp -> data != location)

{

if(temp -> next == head)

{

printf("Given node is not found in the list!!!");

goto EndFunction;

}

else

{

temp = temp -> next;

}

}

newNode -> next = temp -> next;

temp -> next = newNode;

printf("\nInsertion success!!!");

}

EndFunction:

}

void deleteBeginning()

{

if(head == NULL)

printf("List is Empty!!! Deletion not possible!!!");

else

{

struct Node \*temp = head;

if(temp -> next == head)

{

head = NULL;

free(temp);

}

else{

head = head -> next;

free(temp);

}

printf("\nDeletion success!!!");

}

}

void deleteEnd()

{

if(head == NULL)

printf("List is Empty!!! Deletion not possible!!!");

else

{

struct Node \*temp1 = head, temp2;

if(temp1 -> next == head)

{

head = NULL;

free(temp1);

}

else{

while(temp1 -> next != head){

temp2 = temp1;

temp1 = temp1 -> next;

}

temp2 -> next = head;

free(temp1);

}

printf("\nDeletion success!!!");

}

}

void deleteSpecific(int delValue)

{

if(head == NULL)

printf("List is Empty!!! Deletion not possible!!!");

else

{

struct Node \*temp1 = head, temp2;

while(temp1 -> data != delValue)

{

if(temp1 -> next == head)

{

printf("\nGiven node is not found in the list!!!");

goto FuctionEnd;

}

else

{

temp2 = temp1;

temp1 = temp1 -> next;

}

}

if(temp1 -> next == head){

head = NULL;

free(temp1);

}

else{

if(temp1 == head)

{

temp2 = head;

while(temp2 -> next != head)

temp2 = temp2 -> next;

head = head -> next;

temp2 -> next = head;

free(temp1);

}

else

{

if(temp1 -> next == head)

{

temp2 -> next = head;

}

else

{

temp2 -> next = temp1 -> next;

}

free(temp1);

}

}

printf("\nDeletion success!!!");

}

FuctionEnd:

}

void display()

{

if(head == NULL)

printf("\nList is Empty!!!");

else

{

struct Node \*temp = head;

printf("\nList elements are: \n");

while(temp -> next != head)

{

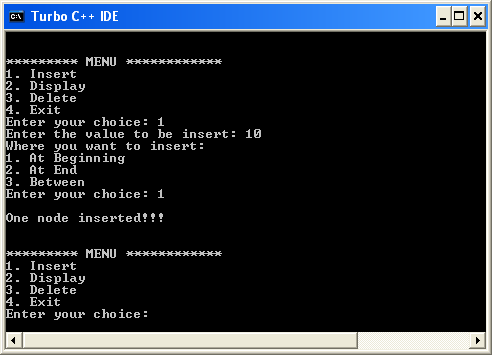
printf("%d ---> ",temp -> data);

}

printf("%d ---> %d", temp -> data, head -> data);

}

}



3) Demonstrate a data structure doubly linked list with necessary members. Also implement operations on doubly linked list (insert at first, insert at last, insert in between and delete from first, last and between, search, display).

#include<stdio.h>

#include<conio.h>

void insertAtBeginning(int);

void insertAtEnd(int);

void insertAtAfter(int,int);

void deleteBeginning();

void deleteEnd();

void deleteSpecific(int);

void display();

struct Node

{

int data;

struct Node \*previous, \*next;

}\*head = NULL;

void main()

{

int choice1, choice2, value, location;

clrscr();

while(1)

{

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

printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");

scanf("%d",&choice1);

switch()

{

case 1: printf("Enter the value to be inserted: ");

scanf("%d",&value);

while(1)

{

printf("\nSelect from the following Inserting options\n");

printf("1. At Beginning\n2. At End\n3. After a Node\n4. Cancel\nEnter your choice: ");

scanf("%d",&choice2);

switch(choice2)

{

case 1: insertAtBeginning(value);

break;

case 2: insertAtEnd(value);

break;

case 3: printf("Enter the location after which you want to insert: ");

scanf("%d",&location);

insertAfter(value,location);

break;

case 4: goto EndSwitch;

default: printf("\nPlease select correct Inserting option!!!\n");

}

}

case 2: while(1)

{

printf("\nSelect from the following Deleting options\n");

printf("1. At Beginning\n2. At End\n3. Specific Node\n4. Cancel\nEnter your choice: ");

scanf("%d",&choice2);

switch(choice2)

{

case 1: deleteBeginning();

break;

case 2: deleteEnd();

break;

case 3: printf("Enter the Node value to be deleted: ");

scanf("%d",&location);

deleteSpecic(location);

break;

case 4: goto EndSwitch;

default: printf("\nPlease select correct Deleting option!!!\n");

}

}

EndSwitch: break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nPlease select correct option!!!");

}

}

}

void insertAtBeginning(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode -> data = value;

newNode -> previous = NULL;

if(head == NULL)

{

newNode -> next = NULL;

head = newNode;

}

else

{

newNode -> next = head;

head = newNode;

}

printf("\nInsertion success!!!");

}

void insertAtEnd(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode -> data = value;

newNode -> next = NULL;

if(head == NULL)

{

newNode -> previous = NULL;

head = newNode;

}

else

{

struct Node \*temp = head;

while(temp -> next != NULL)

temp = temp -> next;

temp -> next = newNode;

newNode -> previous = temp;

}

printf("\nInsertion success!!!");

}

void insertAfter(int value, int location)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode -> data = value;

if(head == NULL)

{

newNode -> previous = newNode -> next = NULL;

head = newNode;

}

else

{

struct Node \*temp1 = head, temp2;

while(temp1 -> data != location)

{

if(temp1 -> next == NULL)

{

printf("Given node is not found in the list!!!");

goto EndFunction;

}

else

{

temp1 = temp1 -> next;

}

}

temp2 = temp1 -> next;

temp1 -> next = newNode;

newNode -> previous = temp1;

newNode -> next = temp2;

temp2 -> previous = newNode;

printf("\nInsertion success!!!");

}

EndFunction:

}

void deleteBeginning()

{

if(head == NULL)

printf("List is Empty!!! Deletion not possible!!!");

else

{

struct Node \*temp = head;

if(temp -> previous == temp -> next)

{

head = NULL;

free(temp);

}

else{

head = temp -> next;

head -> previous = NULL;

free(temp);

}

printf("\nDeletion success!!!");

}

}

void deleteEnd()

{

if(head == NULL)

printf("List is Empty!!! Deletion not possible!!!");

else

{

struct Node \*temp = head;

if(temp -> previous == temp -> next)

{

head = NULL;

free(temp);

}

else{

while(temp -> next != NULL)

temp = temp -> next;

temp -> previous -> next = NULL;

free(temp);

}

printf("\nDeletion success!!!");

}

}

void deleteSpecific(int delValue)

{

if(head == NULL)

printf("List is Empty!!! Deletion not possible!!!");

else

{

struct Node \*temp = head;

while(temp -> data != delValue)

{

if(temp -> next == NULL)

{

printf("\nGiven node is not found in the list!!!");

goto FuctionEnd;

}

else

{

temp = temp -> next;

}

}

if(temp == head)

{

head = NULL;

free(temp);

}

else

{

temp -> previous -> next = temp -> next;

free(temp);

}

printf("\nDeletion success!!!");

}

FuctionEnd:

}

void display()

{

if(head == NULL)

printf("\nList is Empty!!!");

else

{

struct Node \*temp = head;

printf("\nList elements are: \n");

printf("NULL <--- ");

while(temp -> next != NULL)

{

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

}

printf("%d ---> NULL", temp -> data);

}

}

