Date:06-10-2022

Program: Perform all operations on Circular Linked List.

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void beginsert ();

void lastinsert ();

void begin\_delete();

void last\_delete();

void display();

int main ()

{

int choice =0;

while(choice != 6)

{

printf("\nMain Menu\n");

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

printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n4.Delete from last\n5.Search for an element\n6.Show\n7.Exit\n");

printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

beginsert();

break;

case 2:

lastinsert();

break;

case 3:

begin\_delete();

break;

case 4:

last\_delete();

break;

case 5:

display();

case 6:

exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

void beginsert()

{

struct node \*ptr,\*temp;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter the node data?");

scanf("%d",&item);

ptr -> data = item;

if(head == NULL)

{

head = ptr;

ptr -> next = head;

}

else

{

temp = head;

while(temp->next != head)

temp = temp->next;

ptr->next = head;

temp -> next = ptr;

head = ptr;

}

printf("\nnode inserted\n");

}

}

void lastinsert()

{

struct node \*ptr,\*temp;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

printf("\nEnter Data?");

scanf("%d",&item);

ptr->data = item;

if(head == NULL)

{

head = ptr;

ptr -> next = head;

}

else

{

temp = head;

while(temp -> next != head)

{

temp = temp -> next;

}

temp -> next = ptr;

ptr -> next = head;

}

printf("\nnode inserted\n");

}

}

void begin\_delete()

{

struct node \*ptr;

if(head == NULL)

{

printf("\nUNDERFLOW");

}

else if(head->next == head)

{

head = NULL;

free(head);

printf("\nnode deleted\n");

}

else

{ ptr = head;

while(ptr -> next != head)

ptr = ptr -> next;

ptr->next = head->next;

free(head);

head = ptr->next;

printf("\nnode deleted\n");

}

}

void last\_delete()

{

struct node \*ptr, \*preptr;

if(head==NULL)

{

printf("\nUNDERFLOW");

}

else if (head ->next == head)

{

head = NULL;

free(head);

printf("\nnode deleted\n");

}

else

{

ptr = head;

while(ptr ->next != head)

{

preptr=ptr;

ptr = ptr->next;

}

preptr->next = ptr -> next;

free(ptr);

printf("\nnode deleted\n");

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag=1;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item which you want to search?\n");

scanf("%d",&item);

if(head ->data == item)

{

printf("item found at location %d",i+1);

flag=0;

}

else

{

while (ptr->next != head)

{

if(ptr->data == item)

{

printf("item found at location %d ",i+1);

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

}

if(flag != 0)

{

printf("Item not found\n");

}

}

}

void display()

{

struct node \*ptr;

ptr=head;

if(head == NULL)

{

printf("\nnothing to print");

}

else

{

printf("\n printing values ... \n");

while(ptr -> next != head)

{

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

ptr = ptr -> next;

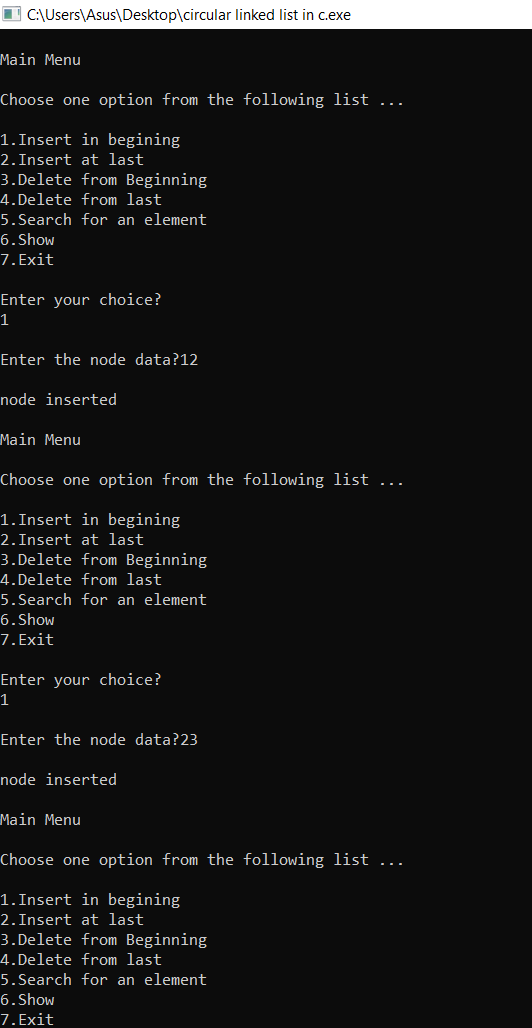
}

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

}

}

Output:



Program: Linked List representation on Stack

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct Node

{

int Data;

struct Node\* next;

}\*rear, \*front;

void delQueue()

{

struct Node \*temp, \*var=rear;

if(var==rear)

{

rear = rear->next;

free(var);

}

else

printf("\nQueue Empty");

}

void insert(int value)

{

struct Node \*temp;

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

temp->Data=value;

if (front == NULL)

{

front=temp;

front->next=NULL;

rear=front;

}

else

{

front->next=temp;

front=temp;

front->next=NULL;

}

}

void display()

{

struct Node \*var=rear;

if(var!=NULL)

{

printf("\nElements are as: ");

while(var!=NULL)

{

printf("\t%d",var->Data);

var=var->next;

}

printf("\n");

}

else

printf("\nQueue is Empty");

}

int main()

{

int i=0;

front=NULL;

printf("\n LINKED QUEUE");

printf(" \n1. Insert");

printf(" \n2. Delete");

printf(" \n3. Display");

printf(" \n4. Exit\n");

while(1)

{

printf(" \nChoose Option: ");

scanf("%d",&i);

switch(i)

{

case 1:

{

int value;

printf("\nEnter an element to insert into Queue: ");

scanf("%d",&value);

insert(value);

display();

break;

}

case 2:

{

delQueue();

display();

break;

}

case 3:{

display();

break;

}

case 4:exit(0);

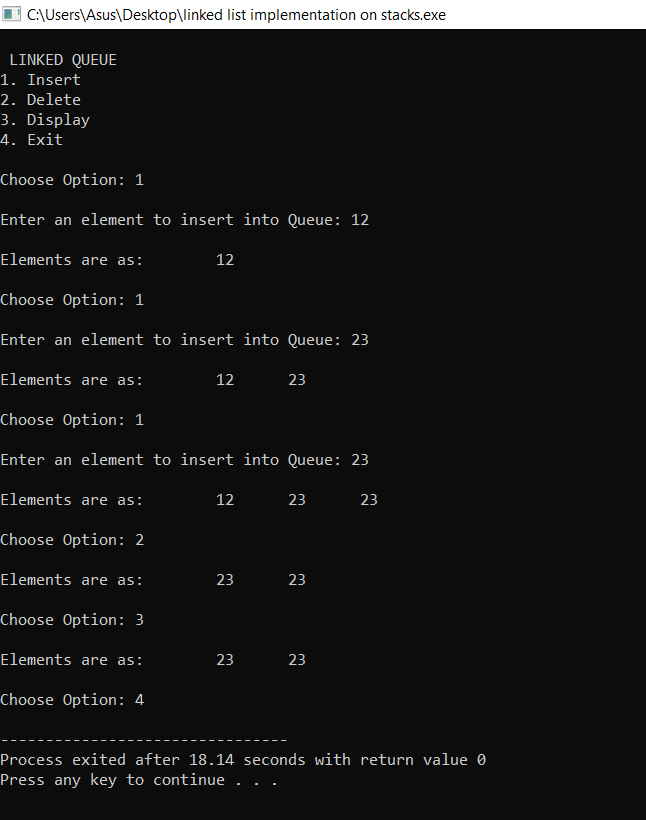
default:printf("\nwrong choice entered");

}

}

}

Output:



Program: Linked List implementation on Queues.

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int Data;

struct Node\* next;

}\*rear, \*front;

void delQueue()

{

struct Node \*temp, \*var=rear;

if(var==rear)

{

rear = rear->next;

free(var);

}

else

printf("\nQueue Empty");

}

void insert(int value)

{

struct Node \*temp;

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

temp->Data=value;

if (front == NULL)

{

front=temp;

front->next=NULL;

rear=front;

}

else

{

front->next=temp;

front=temp;

front->next=NULL;

}

}

void display()

{

struct Node \*var=rear;

if(var!=NULL)

{

printf("\nElements are as: ");

while(var!=NULL)

{

printf("\t%d",var->Data);

var=var->next;

}

printf("\n");

}

else

printf("\nQueue is Empty");

}

int main()

{

int i=0;

front=NULL;

printf("\n LINKED QUEUE");

printf(" \n1. Insert");

printf(" \n2. Delete");

printf(" \n3. Display");

printf(" \n4. Exit\n");

while(1)

{

printf(" \nChoose Option: ");

scanf("%d",&i);

switch(i)

{

case 1:

{

int value;

printf("\nEnter an element to insert into Queue: ");

scanf("%d",&value);

insert(value);

display();

break;

}

case 2:

{

delQueue();

display();

break;

}

case 3:

{

display();

break;

}

case 4:

exit(0);

default:

printf("\nwrong choice entered");

}

}

}

Output:

