**VARUN KUMAR**

**2K19 / IT / 140**

Problem 21: Write a program to implement Stack Data Structure using Queue

#include <iostream>

#include <conio.h>

#include <stdlib.h>

using namespace std;

struct Node

{

int info;

Node \*link;

};

class Queue

{

private:

Node \*front = NULL;

Node \*rear = NULL;

public:

void enqueue(int item);

int dequeue();

int peek();

int isEmpty();

void display();

};

void Queue::enqueue(int item)

{

Node \*p;

p = (Node \*)malloc(sizeof(Node));

if(p == NULL)

{

cout<<"Press a key"<<endl;

}

else

{

p -> info = item;

p -> link = NULL;

if(front == NULL)

{

front = p;

rear = p;

}

else

{

rear -> link = p;

rear = p;

}

}

}

int Queue::dequeue()

{

int item;

if(front == NULL)

{

cout<<"Press a key"<<endl;

getch();

exit(1);

}

else

{

item = front -> info;

front = front -> link;

return item;

}

}

int Queue::peek()

{

int item;

if(front == NULL)

{

cout<<"Press a key"<<endl;

getch();

exit(1);

}

else

{

item = front -> info;

return item;

}

}

int Queue::isEmpty()

{

if(front == NULL)

{

return 1;

}

else

{

return 0;

}

}

void Queue::display()

{

Node \*p;

if(front == NULL)

{

cout<<"Stack is Empty"<<endl;

getch();

exit(1);

}

else

{

p = (Node \*)malloc(sizeof(Node));

p = front;

cout<<"Stack is : ";

while(p!=NULL)

{

cout<<p -> info<<" ";

p = p -> link;

}

cout<<endl;

}

}

class Stack

{

private:

Queue q1,q2;

public:

void push(int x);

int pop();

int top();

void display();

};

void Stack::push(int x)

{

q2.enqueue(x);

while(!q1.isEmpty())

{

q2.enqueue(q1.peek());

q1.dequeue();

}

Queue q;

q = q1;

q1 = q2;

q2 = q;

}

int Stack::pop()

{

if(q1.isEmpty())

{

cout<<"Stack is Empty"<<endl;

getch();

exit(1);

}

else

{

return q1.dequeue();

}

}

int Stack::top()

{

if(q1.isEmpty())

{

cout<<"Stack is Empty"<<endl;

getch();

exit(1);

}

else

{

return q1.peek();

}

}

void Stack::display()

{

q1.display();

}

int main()

{

int choice;

Stack s1;

while(1)

{

cout<<endl;

cout<<"1. PUSH"<<endl;

cout<<"2. POP"<<endl;

cout<<"3. TOP"<<endl;

cout<<"4. DISPLAY"<<endl;

cout<<"5. QUIT"<<endl;

cout<<"Enter the choice : ";

cin>>choice;

switch(choice)

{

case 1:

int d1;

cout<<"Enter the element to be inserted : ";

cin>>d1;

s1.push(d1);

break;

case 2:

int d2;

d2= s1.pop();

cout<<"Popped Element is : ";

cout<<d2<<endl;

break;

case 3:

int d3;

d3 = s1.top();

cout<<"Top Element is : ";

cout<<d3<<endl;

break;

case 4:

s1.display();

break;

case 5:

cout<<"Thanks!!!!!"<<endl;

getch();

exit(0);

default:

cout<<"Wrong Choice!!!!!"<<endl;

cout<<"Try Again....."<<endl;

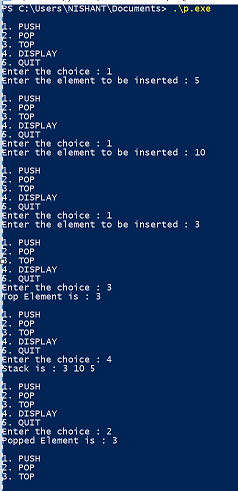
break;

}

}

return 0;

}



# Problem-22 Write a program to implement Queue Data Structure using Stack

**#include <iostream>**

**#include <conio.h>**

**#include <stdlib.h>**

**using namespace std;**

**struct Node**

**{**

**int info;**

**Node \*link;**

**};**

**class Stack**

**{**

**private:**

**Node \*top = NULL;**

**public:**

**void push(int item);**

**int pop();**

**int peek();**

**void display();**

**int isEmpty();**

**};**

**void Stack::push(int item)**

**{**

**Node \*p;**

**p = (Node\*)malloc(sizeof(Node));**

**if(p == NULL)**

**{**

**cout<<"Press a Key"<<endl;**

**}**

**else**

**{**

**p -> info = item;**

**p -> link = top;**

**top = p;**

**}**

**}**

**int Stack::pop()**

**{**

**int item;**

**Node \*p;**

**p = (Node \*)malloc(sizeof(Node));**

**if(top==NULL)**

**{**

**cout<<"Press a Key"<<endl;**

**getch();**

**exit(1);**

**}**

**else**

**{**

**p = top;**

**item = p -> info;**

**top = p -> link;**

**return item;**

**}**

**}**

**int Stack::peek()**

**{**

**int item;**

**if(top == NULL)**

**{**

**cout<<"Press a Key";**

**getch();**

**exit(1);**

**}**

**else**

**{**

**item = top -> info;**

**return item;**

**}**

**}**

**void Stack::display()**

**{**

**Node \*p;**

**if(top == NULL)**

**{**

**cout<<"Queue is Empty"<<endl;**

**getch();**

**exit(1);**

**}**

**else**

**{**

**p = (Node \*)malloc(sizeof(Node));**

**p = top;**

**cout<<"Queue is : ";**

**while(p != NULL)**

**{**

**cout<< p -> info<<" ";**

**p = p -> link;**

**}**

**cout<<endl;**

**}**

**}**

**int Stack::isEmpty()**

**{**

**if(top==NULL)**

**{**

**return 1;**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**class Queue**

**{**

**private:**

**Stack s1,s2;**

**public:**

**void enqueue(int item);**

**int dequeue();**

**int front();**

**void display();**

**};**

**void Queue::enqueue(int item)**

**{**

**while(!s1.isEmpty())**

**{**

**s2.push(s1.peek());**

**s1.pop();**

**}**

**s1.push(item);**

**while(!s2.isEmpty())**

**{**

**s1.push(s2.peek());**

**s2.pop();**

**}**

**}**

**int Queue::dequeue()**

**{**

**int x;**

**if(s1.isEmpty())**

**{**

**cout<<"Queue is Empty"<<endl;**

**getch();**

**exit(1);**

**}**

**else**

**{**

**x = s1.pop();**

**}**

**return x;**

**}**

**int Queue::front()**

**{**

**int x;**

**if(s1.isEmpty())**

**{**

**cout<<"Queue is Empty"<<endl;**

**getch();**

**exit(1);**

**}**

**else**

**{**

**x = s1.peek();**

**}**

**return x;**

**}**

**void Queue::display()**

**{**

**s1.display();**

**}**

**int main()**

**{**

**Queue q1;**

**int choice;**

**while(1)**

**{**

**cout<<endl;**

**cout<<"1. ENQUEUE"<<endl;**

**cout<<"2. DEQUEUE"<<endl;**

**cout<<"3. FRONT"<<endl;**

**cout<<"4. DISPLAY"<<endl;**

**cout<<"5. QUIT"<<endl;**

**cout<<"Enter your choice : ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**int d1;**

**cout<<"Enter the element to be inserted : ";**

**cin>>d1;**

**q1.enqueue(d1);**

**break;**

**case 2:**

**int d2;**

**d2 = q1.dequeue();**

**cout<<"Dequeued Item is : ";**

**cout<<d2<<endl;**

**break;**

**case 3:**

**int d3;**

**d3 = q1.front();**

**cout<<"Front Element is : ";**

**cout<<d3<<endl;**

**break;**

**case 4:**

**q1.display();**

**break;**

**case 5:**

**cout<<"Thanks!!!!!"<<endl;**

**getch();**

**exit(0);**

**default:**

**cout<<"Wrong Choice!!!!!"<<endl;**

**cout<<"Try Again....."<<endl;**

**break;**

**}**

**}**

**return 0;**

**}**

