|  |  |
| --- | --- |
|  | SVKM’s NMIMS  School of Technology Management & Engineering Navi Mumbai Campus |
| Department of Computer Engineering |

|  |  |
| --- | --- |
| Name:  YASH PATIL | SAP ID:  704722000140 |
| Semester: III | Year: II |
| Subject: DSA | Roll No.: A176 |
| Practical: 3 | Date: 11/08/2023 |
| Batch: 1 |  |

**Aim:–** Implementation Of Linear Queue And Circular Queue Data Structure

**Theory:–**

We have used static array to implement given problem statement.

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To create an array, define the data type (like int) and specify the name of the array followed by **square brackets []**.

To insert values to it, use a comma-separated list, inside curly braces:

We can loop through the array elements with the for loop.

The following example outputs all elements in the myNumbers array:

int myNumbers[] = {25, 50, 75, 100};   
int i;   
   
for (i = 0; i < 4; i++) {   
  printf("%d\n", myNumbers[i]);   
}

**Code/Implementation –**

**LINEAR QUEUE CODE:**

##include<iostream>

using namespace std;

#define size 5

int arr[size],front=-1,rear=-1;

void enqueue(int a)

{

if(rear==size==1)

cout<<"\nOverflow\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

else

{

if(rear==-1)

rear=0,front=0;

else

rear++;

arr[rear]=a;

}

}

void dequeue()

{

if(front==-1)

cout<<"Underflow\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

else

{

cout<<"Deleted",arr[front];

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

if(front==rear)

front=-1,rear=-1;

else

front++;

}

}

void display()

{

int i;

cout<<"Elements are: ";

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

for(i=front;i<=rear;i++)

cout<<arr[i];

}

int main()

{

int ch,a;

do

{

cout<<"\n1.Enqueue"<<endl;

cout<<"2.Dequeue"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Exit"<<endl;

cout<<"Choose your option: ";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter the element: ";

cin>>a;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

enqueue(a);

break;

case 2:

dequeue();

break;

case 3:

display();

break;

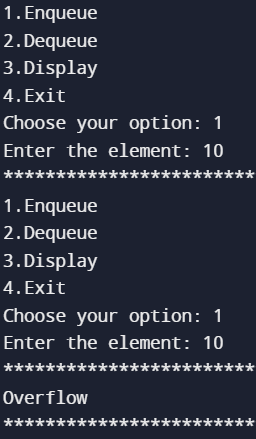
}

}while(ch<=3);

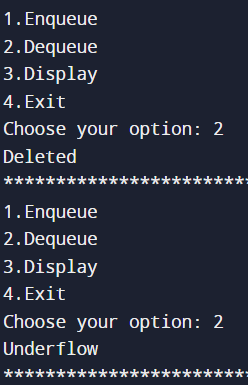
}

**OUTPUT :**

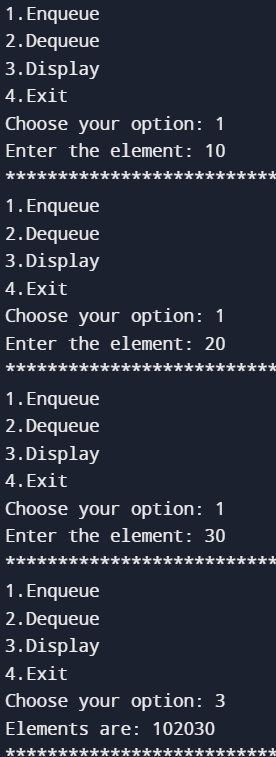
**OVERFLOW**

****

**DEQUEUE AND UNDERFLOW**

****

**ENQUEUE**



**CIRCULAR QUEUE :**

**CODE:**

#include<iostream>

using namespace std;

#define size 5

int arr[size],front=-1,rear=-1;

void enqueue(int a)

{

if(front==(rear+1)%size)

cout<<"\nOverflow\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

else

{

if(rear==-1)

rear=front=0;

else

rear=(rear+1)%size;

arr[rear]=a;

}

cout<<"Front and rear : "<<front<<rear;

}

void dequeue()

{

if(front==-1)

cout<<"Underflow\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

else

{

cout<<"Deleted",arr[front];

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

if(front==rear)

front=-1,rear=-1;

else

front==(front+1)%size;

}

}

void display()

{

if(front<=rear)

{

for(int i=front;i<=rear;i++)

cout<<arr[i];

}

else

{

for(int i=front;i<=size-1;i++)

cout<<arr[i];

for(int i=0;i<=rear;i++)

cout<<arr[i];

}

}

int main()

{

int ch,a;

do

{

cout<<"\n1.Enqueue"<<endl;

cout<<"2.Dequeue"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Exit"<<endl;

cout<<"Choose your option: ";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter the element:";

cin>>a;

enqueue(a);

break;

case 2:

dequeue();

break;

case 3:

display();

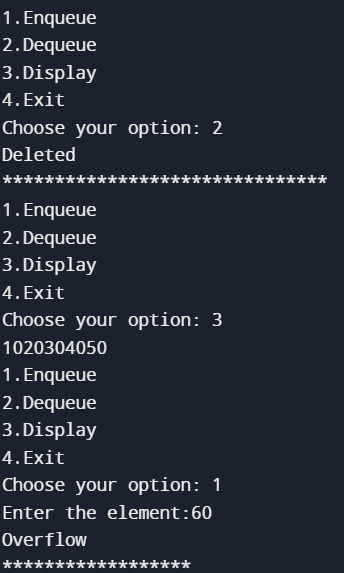
break;

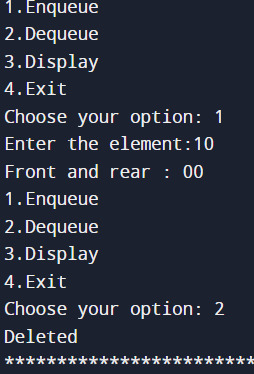
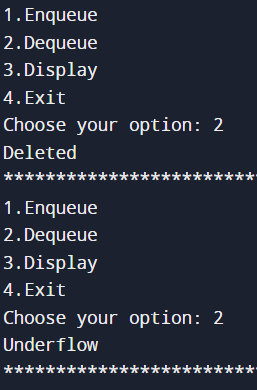
}

}while(ch<=3);

}

**Output:-**



**Conclusion:-**

**From The Given Experiment I Was Successfully Able To Execute The Linear & Circular Queue Operations For Data Structure.**

**Outcome: - Identified And Applied Appropriate Linear Data Structure For The Given Problem**