**Q. WAP to implement linear queue using array.**

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#define MAXSIZE 5

int queue[MAXSIZE];

int front = 0;

int rear = -1;

void cleanScreen()

{

system("cls");

}

int isFull()

{

return (rear == MAXSIZE - 1 ? 1 : 0);

}

int isEmpty()

{

return (rear < front ? 1 : 0);

}

void enqueue(int n)

{

if(isFull()){

cleanScreen();

printf("\nQueue Full!!");

getch();

return;

}

rear++;

queue[rear] = n;

return;

}

int dequeue()

{

if(isEmpty()){

cleanScreen();

printf("\nQueue Empty!!");

return 0;

}

int data = queue[front];

front++;

return data;

}

void display()

{

int i;

cleanScreen();

if(isEmpty()){

printf("\nQueue Empty!!");

return;

}

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

{

printf("\nqueue element %d -> %d", i+1, queue[i]);

}

return;

}

void peek()

{

cleanScreen();

if(isEmpty()){

printf("\nQueue Empty!!");

return;

}

printf("\nfront of queue -> %d", queue[front]);

return;

}

char prompt()

{

char choice;

cleanScreen();

printf("\nPress <1> to Enqueue. \nPress <2> to Dequeue. \nPress <3> to Peek. \nPress <4> to Display. \nPress <ESC> to Exit.");

choice = getch();

return choice;

}

int main()

{

char choice;

int loop = 1;

int n;

while(loop == 1)

{

choice = prompt();

switch (choice)

{

case 27:

loop = 0;

break;

case '1':

cleanScreen();

printf("\nEnter integer to enqueue : ");

scanf("%d", &n);

enqueue(n);

break;

case '2':

cleanScreen();

n = dequeue();

printf("\nInteger dequeued : %d", n);

getch();

break;

case '3':

peek();

getch();

break;

case '4':

display();

getch();

break;

default: loop = 1;

}

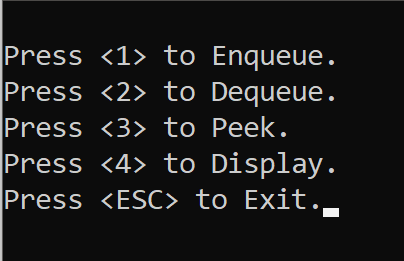
}

return 0;

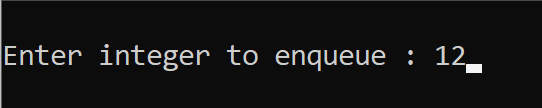
}

**Output:**

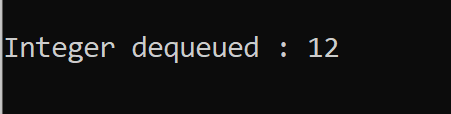
Main Prompt Screen.



Enqueue.

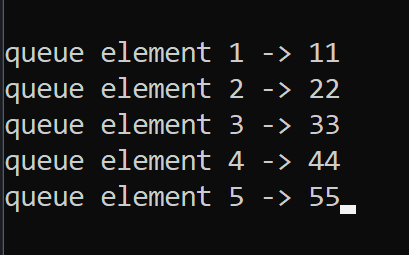


Dequeue.

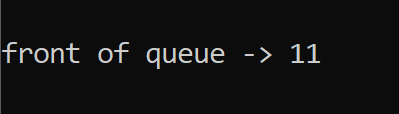


Now, in a new queue, enqueuing 11, 22, 33, 44, 55 in order.

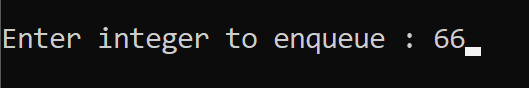
Display.



Peek.



Now, if we try to enqueue 66 in same queue.



It refuses as the queue is full.

