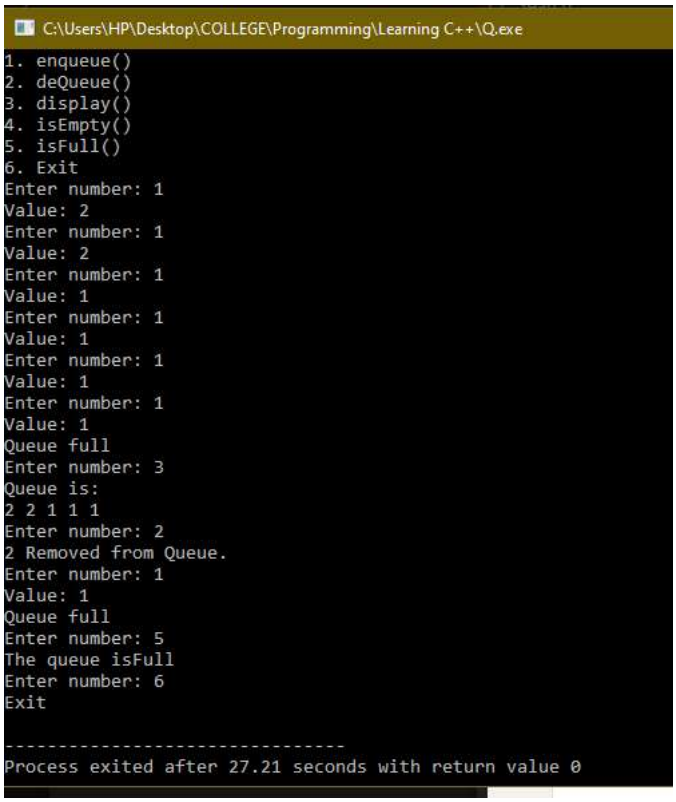
Week3 Batch3

Tuesday, December 07, 2021 10:48 PM

1. Implementation of Queue using arrays

```
#include<iostream>
using namespace std;
class Queue
{
private:
int Front ;
int Rear ;
int Size ;
int *Q ;
public:
Queue(){Front = Rear = -1; Size = 5; Q = new int[Size]; }
Queue(int Size){ Front = Rear = -1; this->Size = Size; Q = new int[this->Size]; }
void enqueue(int x);
int deQueue();
void display();
bool isEmpty();
bool isFull();
};
void Queue :: enqueue(int x)
{
if(Rear == Size-1)
cout<<"Queue full"<<endl;
else
{
Rear++;
Q[Rear] = x;
}
}
int Queue :: deQueue()
{
int x;
if(Front == Rear)
cout<<"Queue empty"<<endl;
else
{
x = Q[Front+1];
Front++;
}
return x;
}
void Queue :: display()
{
for(int i= Front+1; i<=Rear;i++)
cout<<Q[i]<<" ";
cout<<endl;
}
bool Queue :: isEmpty()
{
if(Front == Rear)
return true;
else
return false;
```



```
}
bool Queue :: isFull()
{
if(Rear == Size-1)
return true;
else
return false;
}
int main()
{
int n, val;
Queue q;
cout<<"1. enqueue()"<<endl;
cout<<"2. deQueue()"<<endl;
cout<<"3. display()"<<endl;
cout<<"4. isEmpty()"<<endl;
cout<<"5. isFull()"<<endl;
cout<<"6. Exit"<<endl;
do{
cout<<"Enter number: ";
cin>>n;
switch(n){
case 1:
{
cout<<"Value: ";
cin>>val;
q.enqueue(val);
break;
}
case 2:
{
cout<<q.deQueue()<<" Removed from Queue."<<endl;
break;
}
case 3:
{
cout<<"Queue is: "<<endl;
q.display();
break;
}
case 4:
{
if(q.isEmpty())
cout<<"The queue isEmpty"<<endl;
else
cout<<"The queue is not empty"<<endl;
break;
}
case 5:
{
if(q.isFull())
cout<<"The queue isFull"<<endl;
else
cout<<"The queue is not full"<<endl;
break;
}
case 6:
{
cout<<"Exit"<<endl;
```

```
    break;
    }
    default: cout<<"Enter Valid Value"<<endl;
    }
    }while(n!=6);
    return 0;
    }
```

2. Implement a circular queue of Strings with functions insert, delete and display.

```
#include<iostream>
#include<string.h>
using namespace std;
class Queue
{
char Q[5][25];
int Front,Rear;
public:
Queue(){Front = 0; Rear = -1;}
int is_Queue_Full();
int is_Queue_Empty();
int enQueue(char[]);
int deQueue(char[]);
};
  int Queue :: is_Queue_Full()
  {
    if(Rear!= -1 && Front == (Rear+1)%5)
   return 1;
    return 0;
  }
 int Queue :: is_Queue_Empty()
  {
    if(Rear == -1)
   return 1;
    return 0;
  }
 int Queue::enQueue(char Data[25])
  {
    if(is_Queue_Full())
      return false;
    Rear = (Rear + 1)%5;
    strcpy(Q[Rear], Data);
    return true;
   }
 int Queue :: deQueue(char Data[25])
  {
     if(is_Queue_Empty())
       return false;
     strcpy(Data,Q[Front]);
     Q[Front][0] = '\0';
     Front = (Front + 1)%5;
     if(Front == (Rear+1)%5)
       { Front = 0; Rear = -1; }
     return true;
   }

int main()
```

```
C:\Users\HP\Desktop\COLLEGE\Programming\Learning C++\CircularStringQ.exe
1:Entry          2:Deletion
Enter your Choice: 1
Enter String: This
Entered

Continue: y
Enter your Choice: 1
Enter String: is
Entered

Continue: y
Enter your Choice: 1
Enter String: Week
Entered

Continue: y
Enter your Choice:
1
Enter String: 3
Entered

Continue: y
Enter your Choice: 2
Deleted Element: This
Continue: y
Enter your Choice: 2
Deleted Element: is
Continue: y
Enter your Choice: 2
Deleted Element: Week
Continue: y
Enter your Choice: 2
Deleted Element: 3
Continue: y
Enter your Choice: 2
Queue is Empty can not Remove

Continue: y
Enter your Choice: 2
Queue is Empty can not Remove

Continue: n

--------------------------------
```

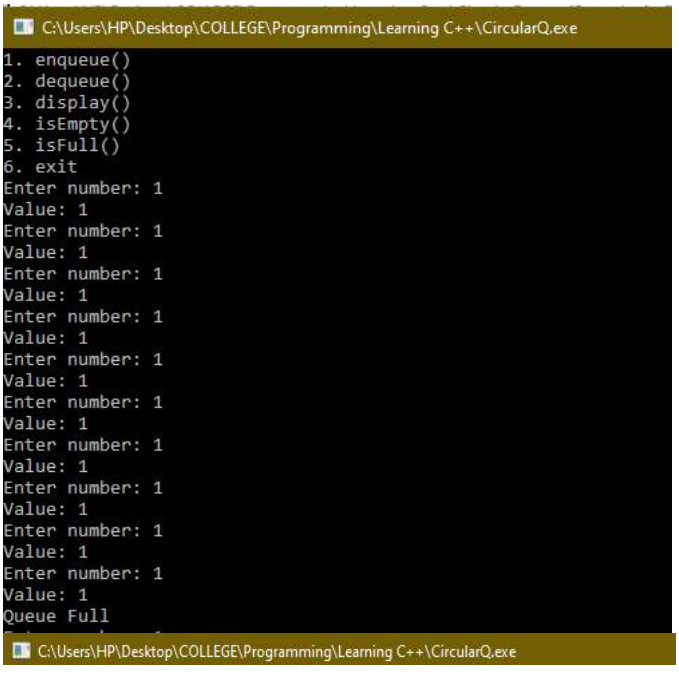
```
 {

Queue QU;
char Element[25];
   int Choice;
char Answer;
cout<<"1:Entry \t 2:Deletion "<<endl;
do
 {
   cout<< "Enter your Choice: ";
   cin>>Choice;
   switch(Choice)
       {
      case 1:
                 cout<<"Enter String: ";
         cin>>Element;
         if(QU.enQueue(Element))
                       cout<<"Entered\n";
         else
               cout<<"Queue Full Can not Enter\n";
                    break;
      case 2:
                    if(QU.deQueue(Element))
               cout<<"Deleted Element: "<<Element;
         else
                       cout<<"Queue is Empty can not Remove\n";
            break;
         }
   cout<<"\nContinue: ";
   cin>>Answer;
 }while(Answer=='y'||Answer=='Y');
return 1;
 }
```

3. Write a program to implement the circular queue using arrays

```
#include<iostream>
using namespace std;

class CircularQ
{
private:
int Front;
int Rear;
int Size;
int *Q;
public:
CircularQ(){Front=Rear=0; Size=10; Q= new int[Size]; }
CircularQ(int Size){ Front=Rear=0; this->Size=Size; Q= new int[this->Size]; }
void enqueue(int x);
int dequeue();
void display();
bool isEmpty();
bool isFull();
};
```



```
void CircularQ :: enqueue(int x)
{
if((Rear+1)%Size==Front)
cout<<"Queue Full"<<endl;
else
{
Rear=(Rear+1)%Size;
Q[Rear]=x;
}
}

int CircularQ :: dequeue()
{
int x;
if(Front==Rear)
cout<<"Queue Empty"<<endl;
else
{
Front=(Front+1)%Size;
x=Q[Front];
}
return x;
}

void CircularQ :: display()
{
int i=Front+1;
do{
cout<<Q[i]<<" ";
i=(i+1)%Size;
}while(i!=(Rear+1)%Size);
cout<<endl;
}

bool CircularQ :: isEmpty()
{
if(Front==Rear)
return true;
else
return false;
}

bool CircularQ :: isFull()
{
if((Rear+1)%Size==Front)
return true;
else
return false;
}

int main()
{
int n, val;
CircularQ q;
cout<<"1. enqueue()"<<endl;
cout<<"2. dequeue()"<<endl;
cout<<"3. display()"<<endl;
cout<<"4. isEmpty()"<<endl;
cout<<"5. isFull()"<<endl;
```

```
Enter number: 1
Value: 1
Queue Full
Enter number: 1
Value: 1
Queue Full
Enter number: 2
Removed: 1
Enter number: 1
Value: 1
Enter number: 2
Removed: 1
Enter number: 2
Removed: 1
Enter number: 2
Removed: 1
Enter number: 22
Enter Valid Value!!
Enter number: 2
Removed: 1
Enter number:
2
Removed: 1
Enter number:
2
Removed: 1
Enter number: 4
The queue is not empty
Enter number: 5
The queue is not full
Enter number: 6
Exit

--------------------------------
Process exited after 27.09 seconds with return value 0
Press any key to continue . . .
```

```
cout<<"6. exit"<<endl;
do{
cout<<"Enter number: ";
cin>>n;
switch(n){
case 1:
{
cout<<"Value: ";
cin>>val;
q.enqueue(val);
break;
}
case 2:
{
cout<<"Removed: "<<q.dequeue()<<endl;
break;
}
case 3:
{
cout<<"Queue is: "<<endl;
q.display();
break;
}
case 4:
{
if(q.isEmpty())
cout<<"The queue isEmpty"<<endl;
else
cout<<"The queue is not empty"<<endl;
break;
}
case 5:
{
if(q.isFull())
cout<<"The queue isFull"<<endl;
else
cout<<"The queue is not full"<<endl;
break;
}
case 6:
{
cout<<"Exit"<<endl;
break;
}
default: cout<<"Enter Valid Value!!"<<endl;
}
}while(n!=6);
return 0;
}
```