3.5 Queues

This section will guide you to:

* Create a Windows Console project in Visual Studio to implement a queue
* Create a class, Queue, that will implement a queue
* Create a method, runApp(), that will show the use of a queue

**Development Environment**

* Visual Studio 2019 Community Version

This guide has six subsections, namely:

* + 1. Creating a Windows Console project in Visual Studio to implement a queue
    2. Creating a class, Queue, that will implement a queue
    3. Adding a method, runApp(), that will show the use of a queue
    4. Building the project
    5. Publishing and running the project
    6. Pushing the code to your GitHub repositories

**Step 3.5.1:** Creating a Windows Console project in Visual Studio to implement a queue

* Open Visual Studio.
* From the top menu, select **File->New->Project.**
* In **Create A New Project** screen, select **Console app (.NET Core)** from the list of available project types and click on **Next.**
* Enter **Project Name** as **Phase1Section4.11** and click on **Create.**
* This will create the files for a Windows Console project.

**Step** **3.5.2:** Creating a class, Queue, that will implement a queue

* In the **Solution Explorer** window, right click on **Phase1Section4.11** and choose **Add->Class.**
* Enter Class name as **Queue.cs** and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section4.\_11

{

**public** **class** Queue

{

**private** **int**[] ele;

**private** **int** front;

**private** **int** rear;

**private** **int** max;

**public** Queue(**int** size)

{

ele = **new** **int**[size];

front = 0;

rear = -1;

max = size;

}

**public** **void** insert(**int** item)

{

**if** (rear == max - 1)

{

**return**;

}

**else**

{

ele[++rear] = item;

}

}

**public** **int** delete()

{

**if** (front == rear + 1)

{

**return** -1;

}

**else**

{

**return** ele[front++];

}

}

**public** **string** printQueue()

{

**string** retVal = "";

**if** (front == rear + 1)

{

**return** retVal;

}

**else**

{

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

{

retVal += ele[i] + " " ;

}

}

**return** retVal;

}

}

}

**Step 3.5.3:** Adding a method, runApp(), that will show the use of a queue

* Select **Program.cs** as the current Code tab.
* Enter the following code:

**using** System;

**namespace** Phase1Section4.\_11

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

runApp();

}

**public** **static** **void** runApp()

{

Queue Q = **new** Queue(10);

Q.insert(10);

Q.insert(20);

Q.insert(30);

Q.insert(40);

Q.insert(50);

Q.insert(60);

Q.insert(70);

Q.insert(80);

Q.insert(90);

Q.insert(100);

Console.WriteLine(Q.printQueue());

Q.delete();

Q.delete();

Console.WriteLine(Q.printQueue());

}

}

}

**Step 3.5.4:** Building the project

* From the top menu, choose **Build->Build Solution.**
* If any compile errors are shown, fix them as required.

**Step** **3.5.5:** Publishing and running the project

* From the top menu, select **Debug->Start Without Debugging.**
* This will execute the program in a console window.

**Step** **3.5.6:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master