Tutorial 5 – Abstract Data Types (ADTs)

By Chris Dworczyk

# Carry out research on Linked List and Queue ADTs using relevant online and recommended resources to answer the following questions

## Brief description

### Linked List

A linked list is a sequence of data connected by a link. Each link connects to the next link which contains the next bit of data. This is the second most used data structure after arrays.

#### Operations

Link − each link of a linked list can store a data called an element.

Next − each link of a linked list contains a link to the next link called Next.

LinkedList − A Linked List contains the connection link to the first link called First.

#### Uses/Applications

Linked lists are a dynamic data structure, which can grow and be pruned, allocating and deallocating memory while the program is running. This means the list sized is not need to be known before implementation.

### Queue

Queue is similar to a stack however it’s open on both sides. One side to add things (enqueue) from and the other to remove things from (dequeue).

It operates on a LIFO (last in first out) methodology which means the last thing you insert will be the first things to come out.

#### Operations

enqueue() − add (store) an item to the queue.

dequeue() − remove (access) an item from the queue.

peek() − Gets the element at the front of the queue without removing it.

isfull() − Checks if the queue is full.

isempty() − Checks if the queue is empty.

#### Uses/Applications

If you have a web-site which serves files to thousands of users. You cannot service all requests; you can only handle 100 at once. A fair policy would be first-come-first serve: serve 100 at a time in order of arrival. A Queue would definitely be the most appropriate data structure.

# Carry out research using relevant online and recommended resources to provide 2 examples of algorithms executable within C # environment to implement Stack and Queue ADTs.

## Stack

using System;

using System.Collections;

public class SamplesStack

{

public static void Main()

{

// Creates and initializes a new Stack.

Stack myStack = new Stack();

myStack.Push("Hello");

myStack.Push("World");

myStack.Push("!");

myStack.Push("HELLO WORLD!!!");

// Displays the properties and values of the Stack.

Console.WriteLine("myStack");

Console.WriteLine("\tCount: {0}", myStack.Count);

Console.Write("\tValues:");

PrintValues(myStack);

}

public static void PrintValues(IEnumerable myCollection)

{

foreach (Object obj in myCollection)

Console.Write(" {0}", obj);

Console.WriteLine();

Console.Read();

}

}

## Queue

using System;

using System.Collections;

public class SamplesQueue

{

public static void Main()

{

// Creates and initializes a new Queue.

Queue myQ = new Queue();

myQ.Enqueue("Hello");

myQ.Enqueue("World");

myQ.Enqueue("!");

// Displays the properties and values of the Queue.

Console.WriteLine("myQ");

Console.WriteLine("\tCount: {0}", myQ.Count);

Console.Write("\tValues:");

PrintValues(myQ);

}

public static void PrintValues(IEnumerable myCollection)

{

foreach (Object obj in myCollection)

Console.Write(" {0}", obj);

Console.WriteLine();

Console.Read();

}

}