**The Queue**

The Queue is a First In, First Out (FIFO) data structure. This is because the first item you put into the queue will be the first item you will take out. This is useful for when you want to go through data in time order.

* New elements may only be added to the end of a queue. This is called ‘enQueue‘
* Elements may only be retrieved from the front of a queue. This is called deQueue’

**Operations On a Queue:**

* enQueue(item) – Adds a new item to the end of the queue
* deQueue() – Removes the front item of the queue and returns it
* isEmpty() – Checks if the queue is empty
* isFull() – Checks if the queue is full

**Visualising the Queue:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| “A” |  |  |  |  |  |  |

We will now enQueue “B”, “C” & “D”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| “A” | “B” | “C” | “D” |  |  |  |

We will now deQueue

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | “B” | “C” | “D” |  |  |  |

**Advantages:**

* Data is accessed by two memory locations. The front and the end of the queue.
* It can be used for tasks for when the Stack is not a suitable data structure.
* It makes good use of memory space as there are no gaps.
* It makes good use of processer time as adding or deleting date is just a single operation.

**Disadvantages:**

* It is an inflexible data structure.
* The locations of the front and end of the Queue are always changing.

**How data is stored in the Queue:**

The computer uses pointers to store the start and the end of the queue. This is called the **head** and the **tail**.

All the data items stay in the same place. It is only the pointers that move.

For example:

Here are some values stored within a Queue data structure:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| “A” | “B” | “C” | “D” | “E” | “F” | “G” |  |  |

We will deQueue

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | “B” | “C” | “D” | “E” | “F” | “G” |  |  |

The position where the value “B” is located, is now the start of the queue and “G” is the end of the queue.