# Data structures ADTs

**Stack:**

|  |
| --- |
| **ADT Stack** |
| Stack = {Elements= <>, Top= <top>, Size = <size>} |
| {inv: } |
| Primitive operations:   * Stack: 🡪 Stack * push: 🡪 Stack * pop: Stack 🡪 Stack * isEmpty: Stack 🡪 Boolean * top: Stack 🡪 Element |

**Operation descriptions:**

1. **Stack:**

Builds an empty stack.

1. **push:**

Adds a specified element to the stack, always on its top.

1. **pop:**

Returns, and removes, the element placed on the top of the stack.

1. **isEmpty:**

Says if the stack is empty or not.

1. **top:**

Returns, but not removes, the element placed on the top of the stack.

**Queue:**

|  |
| --- |
| **Queue ADT** |
| Queue = {Elements = <>, Front = <front>, Back = <back>, Size = <size>} |
| {inv: } |
| Primitive operations:   * Queue: 🡪 Queue * enqueue: 🡪 Queue * dequeue: Queue 🡪 Element * front: Queue 🡪 Element * isEmpty: Queue 🡪 boolean |

**Operation descriptions:**

1. **Queue:**

Builds an empty queue.

1. **enqueue:**

Adds a specified element to the queue, always on the back.

1. **dequeue:**

Returns, and removes, the element placed on the front of the queue.

1. **isEmpty:**

Says if the queue is empty or not.

1. **front:**

Returns, but not removes, the element placed on the front of the queue.

**Hashtable:**

|  |
| --- |
| **Hashtable ADT** |
| Hashtable = {Elements = <>, Size = <size>} |
| {inv: } |
| Primitive operations:   * Hashtable: 🡪 Hashtable * insert: 🡪 Hashtable * search: 🡪 Element * delete: 🡪 Hashtable |

**Operation descriptions:**

1. **Hashtable:**

Builds an empty hashtable.

1. **insert:**

Receives a value and a key and adds the couple to the hashtable, always on its head.

1. **search:**

Receives a key and searches the element in the table with that key.

1. **delete:**

Receives an element and removes it from the hashtable.

**Priority Queue:**

|  |
| --- |
| **MaxPriorityQueue ADT** |
| PriorityQueue = {Elements = <>, Front = <front>, Back = <back>, Size = <size>} |
| {inv: ) )} |
| Primitive operations:   * MaxPriorityQueue: 🡪 MaxPriorityQueue * insert: 🡪 MaxPriorityQueue * maximum: MaxPriorityQueue 🡪 Element * extractMax: MaxPriorityQueue 🡪 Element * increaseKey: 🡪 MaxPriorityQueue |

**Operation descriptions:**

1. **MaxPriorityQueue:**

Builds an empty max priority queue.

1. **insert:**

Receives a value and a key and inserts the couple to the max priority queue, always on its back.

1. **maximum:**

Returns, but not removes, the element with the maximum priority of the max priority queue.

1. **extractMax:**

Returns, and removes, the element with the maximum priority of the max priority queue.

1. **increaseKey:**

Receives a priority (key) and increases the priority of a specified element to the priority that it received previously.