|  |  |  |
| --- | --- | --- |
| HEAP | | |
| Heap= {HeapData= <T1, T2,…,Tn>, SizeOfHeap=<Integer>,HeapMaxSize=n,FRONT=1} | | |
| {Inv: (sizeOfHeap ≥0) ⋀ (heapMaxSize>0) ⋀ (sizeOfHeap ⋀ heapMaxSize) ∈ Z} | | |
| Heap:  GetParentPosition:  GetLeftChildPosition:  GetRightChildPosition:  CheckLeaf:  Swap:  maxHeapify:  DesignMaxHeap: | HEAPDATA  HEAP x INTEGER  HEAP x INTEGER  HEAP x INTEGER  HEAP x INTEGER  HEAP x INTEGER x INTEGER  HEAP x INTEGER  HEAP | 🡪 HEAP  → INTEGER  🡪 INTEGER  🡪 INTEGER  → BOOLEAN  → HEAP  🡪 HEAP  🡪 HEAP |

**Constructor Operations:**

|  |
| --- |
| **Heap()**  “Creates a new Heap”  {pre: datos=<T1, T2,…,Tn>}  {post: Heap= {HeapData= <T1, T2,…,Tn>, SizeOfHeap = 0, HeapMaxSize=n} } |

**Modifying Operations:**

|  |
| --- |
| **Swap()**  “Changes two elements of place according to an index given”  {pre: Heap={HeapData= <T1, T2,Ti,Tj,…,Tn>, SizeOfHeap=<Integer>,HeapMaxSize=n},FirstNode=i, SecondNode=j}  {post: Heap={HeapData= <T1, T2,Tj,Ti,…,Tn>, SizeOfHeap=<Integer>,HeapMaxSize=n} } |

|  |
| --- |
| **MaxHeapify()**  “Heapifies the element given of the Heap in order to maintain its property”  {pre: Heap={HeapData= < T1, T2,Ti,Tj,…,Tn >, SizeOfHeap=<Integer>,HeapMaxSize=n} , Position =i}  {post: Heap={HeapData= < T1, T2,Ti,Tj,…,Tn >, SizeOfHeap=<Integer>,HeapMaxSize=n} ; i,j Ti>Tj } |

|  |
| --- |
| **DesignMaxHeap()**  “Heapifies all of the elements of the array in order to assure the properties of the structure”  {pre: TRUE }  {post: Heap={HeapData= < T1, T2,…,Tn >, SizeOfHeap=<Integer>,HeapMaxSize=n}; ∀i,j Ti>Tj} |

**Analyzing Operation:**

|  |
| --- |
| **GetParentPosition()**  “Returns ‘Parent’ element of the one in said position”  {pre: Position=n, n ∈ Z}  {post: (n-1)/2} |

|  |
| --- |
| **GetRightChildPosition()**  Returns ‘Right Child’ element of the one in said position”  {pre: Position=n, n ∈ Z }  {post: (2\*n)+1 } |

|  |
| --- |
| **GetLeftChildPosition()**  Returns ‘Left Child’ element of the one in said position”  {pre: Position=n, n ∈ Z }  {post: (2\*n) } |

|  |
| --- |
| **CheckLeaf()**  “Returns whether the element in said position has the possibility of owning any children”  {pre: Position=n, n ∈ Z }  {post: TRUE if [n> (SizeOfHeap/2)] ⋀ (n ≤ sizeOfHeap), else FALSE} |