**TADs (tipo abstracto de dato) de las estructuras de datos**

1. **Tabla Hash**

|  |
| --- |
| TAD TablaHash  {Tamaño = <tamaño>, Elementos = <elementos>, FunciónHash = <función\_hash>}  { inv: TablaHash.Tamaño > 0, TablaHash.Elementos ≠ null } |
| Operaciones Primitivas: |
| ▪ CrearTablaHash: Entero → TablaHash  ▪ InsertarEnTabla: TablaHash x Clave x Valor → TablaHash  ▪ ObtenerValor: TablaHash x Clave → Valor  ▪ BorrarDeTabla: TablaHash x Clave → TablaHash ▪ ContieneClave: TablaHash x Clave → Booleano  ▪ isEmpty: HashTable → Booleano |

|  |
| --- |
| CrearTablaHash(tamaño) "Crea una nueva tabla hash con el tamaño especificado" |
| { pre: {tamaño > 0} } |
| { post: TablaHash.Tamaño = tamaño, TablaHash.Elementos está vacía } |

|  |
| --- |
| InsertarEnTabla(tabla, clave, valor) "Inserta un par clave-valor en la tabla hash" |
| { pre: {TablaHash = tabla, clave ≠ null, valor ≠ null} } |
| { post: TablaHash.Elementos contiene (clave, valor) } |

|  |
| --- |
| ObtenerValor(tabla, clave) "Obtiene el valor asociado a una clave en la tabla hash" |
| { pre: {TablaHash = tabla, clave ≠ null} } |
| { post: Si (clave, valor) está en TablaHash.Elementos, entonces se devuelve valor; de lo contrario, se devuelve un valor nulo } |

|  |
| --- |
| BorrarDeTabla(tabla, clave) "Borra un par clave-valor de la tabla hash" |
| { pre: {TablaHash = tabla, clave ≠ null} } |
| { post: Si (clave, valor) está en TablaHash.Elementos, entonces se elimina, y tabla.size = tabla.size - 1; de lo contrario, no ocurre ningún cambio en la tabla } |

|  |
| --- |
| ContieneClave(tabla, clave) "Verifica si la tabla hash contiene una clave" |
| { pre: {TablaHash = tabla, clave ≠ null} } |
| { post: Devuelve true si la tabla contiene la clave; de lo contrario, devuelve false } |

|  |
| --- |
| isEmpty(tabla) "Verifica si la tabla hash está vacía" |
| { pre: {HashTable = tabla} } |
| { post: Devuelve true si la tabla hash está vacía; de lo contrario, devuelve false } |

1. **Priority Queue**

|  |
| --- |
| TAD Priority Queue  {Elementos = <elementos>, Tamaño = <tamaño>, FunciónPrioridad = <función\_prioridad>}  { inv: PriorityQueue.Tamaño >= 0, PriorityQueue.Elementos ≠ null } |
| Operaciones Primitivas: |
| ▪ CrearPriorityQueue: → PriorityQueue  ▪ InsertarEnCola: PriorityQueue x Elemento → PriorityQueue  ▪ ObtenerSiguiente: PriorityQueue → Elemento  ▪ QuitarSiguiente: PriorityQueue → PriorityQueue  ▪ isEmpty: PriorityQueue → Booleano |

|  |
| --- |
| CrearPriorityQueue()"Crea una nueva cola de prioridad vacía" |
| { pre: NIL } |
| { post: PriorityQueue.Tamaño = 0, PriorityQueue.Elementos está vacía } |

|  |
| --- |
| InsertarEnCola(cola, elemento) "Inserta un elemento en la cola de prioridad" |
| { pre: {PriorityQueue = cola, elemento ≠ null} } |
| { post: PriorityQueue.Tamaño = cola.Tamaño + 1, elemento se encuentra en PriorityQueue.Elementos con prioridad adecuada } |

|  |
| --- |
| ObtenerElementoMax(cola) "Obtiene el elemento de mayor prioridad en la cola" |
| { pre: {PriorityQueue = cola} } |
| { post: element = cola.maxMaxElement. Se devuelve el elemento de mayor prioridad en la cola sin eliminarlo} |

|  |
| --- |
| DesencolarElementoMax(cola) "Quita/desencola el elemento de mayor prioridad de la cola" |
| { pre: {PriorityQueue = cola} } |
| { post: PriorityQueue.Tamaño = cola.Tamaño - 1, el elemento de mayor prioridad se elimina de PriorityQueue.Elementos} |

|  |
| --- |
| isEmpty(cola) "Verifica si la cola de prioridad está vacía" |
| { pre: {PriorityQueue = cola} } |
| { post: Devuelve true si la cola de prioridad está vacía; de lo contrario, devuelve false } |