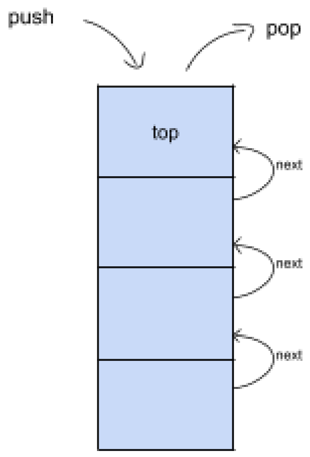
TAD Stack

{ inv: size (Stack) ≥ 0 Λ top = s_n }
Operaciones primitivas: ● Stack B ● top Stack → Node at the top ● isEmpty Stack → boolean ● pop Stack → Node at the top ● push Stack x Element → Stack

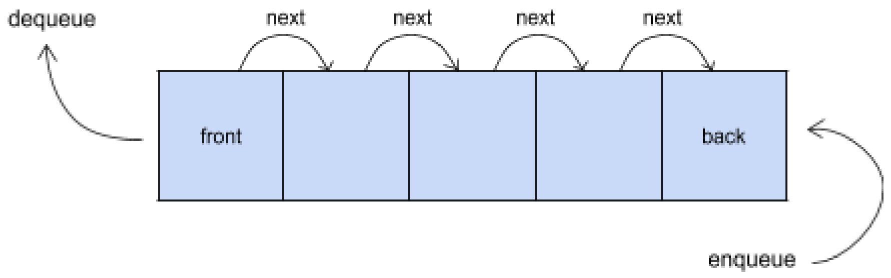
Stack()
“Creates a new stack.”
{ pre: true }
{ post: Stack ≠ Ø}
Operation type: constructor

top()
“Gets the value of the node at the top of the Stack.”
{ pre: Stack ≠ null, Stack = $\{s_0, s_1, s_2 ... s_n\}$ }
{ post: Node s_0}
Operation type: analyzer

isEmpty()
“Determines if the stack is empty.”
{ pre: Stack ≠ null}
{ post: true if Stack = Ø, false if Stack ≠ Ø}
Operation type: analyzer

pop()
“Extracts the element at the top of the stack.”
{ pre: Stack ≠ null, Stack = $\{s_0, s_1, s_2 ... s_n\}$ }
{ post: Stack = $\{s_1, s_2, s_3 ... s_n\}$} }
Operation type: modifier

push(s_k)
“Adds a new node to the top of the Stack.”
{ pre: Stack ≠ null, Stack = $\{s_0, s_1, s_2 ... s_n\}$ }
{ post: Stack = $\{s_k, s_0, s_1, s_2 ... s_n\}$ v Stack = $\{s_k\}$ }
Operation type: modifier

TAD Queue
 dequeue next next next next front back enqueue
{ inv: size(Queue) ≥ 0 Λ front = q_1 Λ back = q_n }
Operaciones primitivas: ● Queue T ● front Queue → Node at the front ● isEmpty Queue → boolean ● dequeue Queue → Node at the top ● enqueue Queue x Element → Queue

Queue	T	
front	Queue	→ Node at the front
isEmpty	Queue	→ boolean
dequeue	Queue	→ Node at the top
enqueue	Queue x Element	→ Queue

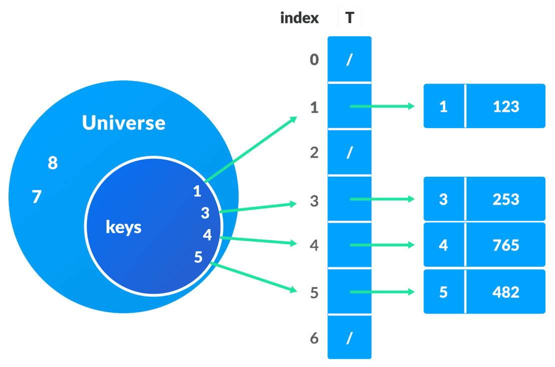
Queue()
"Creates a new queue."
{ pre: true }
{ post: Queue ≠ Ø}
Operation type: constructor

front()
"Gets the value of the node at the front of the Queue."
{ pre: Queue ≠ null, Queue = {q_0, q_1, q_2...q_n} }
{ post: Node q_0 }
Operation type: analyzer

isEmpty()
“Determines if the queue is empty.”
{ pre: Queue ≠ null}
{ post: true if Queue = Ø, false if Queue ≠ Ø}
Operation type: analyzer

dequeue()
“Extracts the element at the front of the queue.”
{ pre: Queue ≠ null, Queue = $\{q_0, q_1, q_2...q_n\}$ }
{ post: Queue = $\{q_1, q_2, q_3...q_{n-1}\}$} }
Operation type: modifier

enqueue(q_k)
“Adds a new node to the back of the Queue.”
{ pre: Queue ≠ null, Queue = $\{q_0, q_1, q_2...q_n\}$ }
{ post: Queue = {Queue = $\{q_0, q_1, q_2...q_n, q_k\}$ v Queue = $\{q_k\}$ }
Operation type: modifier

TAD HASH TABLE
 index T 0 1 2 3 4 5 6 / / / Universe 8 7 keys 1 3 4 5 1 123 3 253 4 765 5 482
{ inv: Max > 0 ^ # conoc < max ^ conoc = dom tabla }
Operaciones primitivas • create m → d • add c: T0 ^ v: T1 → d • remove c: T0 → d • search c: T0 ^ v: T1 → d • exists d ^ c: T0 → boolean

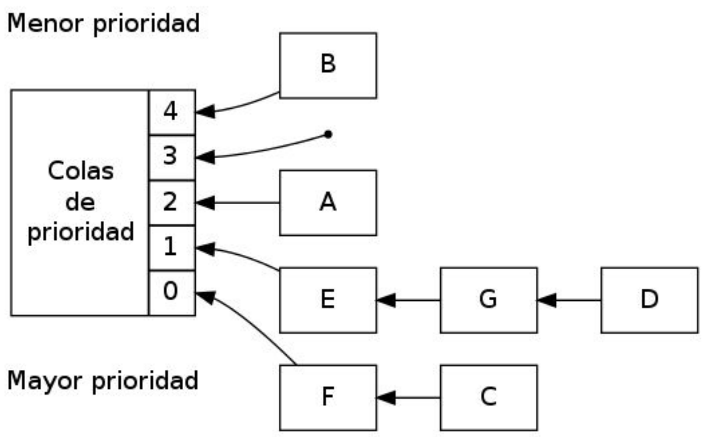
create()
“Create a new object in the system”
{ pre: m > 0 }
{ pos: d.MAX = m ^ d.conoc = 0 ^ d.tabla = 0 }
Operation type: modifier

add()
“Add a object into the hash table”
{ pre: c E d.conoc ^ # d.conoc < d.MAX }
{ pos: d.conoc = d0.conoc U { c } ^ d.tabla = d0.tabla U { (c, v) } }
Operation type: modifier

remove()
“Eliminated a object in the hash table”
{ pre: c E d.conoc }
{ pos: d.conoc = d0.conoc { c } ^ d.tabla = d0.tabla { (c, d0.tabla c) } }
Operation type: modifier

search()
“Look for any object into the hash table”
{ pre: c E d.conoc }
{ pos: v = d.tabla c }
Operation type: analyzer

exists()
“Search if a object is live”
{ pre: true }
{ pos: e = (c E d.conoc) }
Operation type: analyzer

TAD PRIORITY QUEUE
 Menor prioridad Colas de prioridad 4 3 2 1 0 B A E G D F C Mayor prioridad
{ pre: true}
Operaciones primitivas: ● empty → C prioridad a ● insert C prioridad a → C prioridad a ● first C prioridad a → a ● rest C prioridad a → C prioridad a ● valid C prioridad a → boolean

empty()
It is the empty priority queue.
{ pre: true }
{ pos: true if the priority is queue }
Operation type: analyzer

insert()
Add element x to priority queue
{ pre: true }
{ pos: Priority queue with the new item added to the place you It corresponds to you according to your priority. }
Operation type: modifier

first()
It is the first item in the priority queue
{ pre: The priority queue on which the query is made must not be empty. }
{ pos: Returns the element with the highest priority of those stored, without removing it from the queue.}
Operation type: analyzer

rest()
It's the rest of the priority queue
{ pre: true }
{ pos: Returns the element with the highest priority of those stored, without remove it from the queue. }
Operation type: analyzer

valid()
Check if c is a valid priority queue
{ pre: true }
{ pos: Returns true in case the priority queue is valid. }
Operation type: analyzer