By:
Juan Diego Lora
Santiago Prado
Jeison Lasprilla

TAD

TAD Grafo

Graph = {vertexList = <vertexList>}

{ invariant: Each vertex of the graph have the same type.
each vertex has a different name
each vertex has different arists}

- Operaciones Primitivas:
-
- CreateGraph: -> Graph
- InsertVertex: Vertex -> Graph
- GetVertex: number -> Vertex
- InsertArista : Vertex, Vertex -> Graph
- Order: -> Number
- sucesors: number -> Vertex
- show: -> String
- conexMethod: -> boolean

CreateGraph ()
"Create a new graph with an empty vertex list"
{ pre: TRUE }
{ post: Graph = { vertexList = ∅ }}

insertVertex (vertex)
"Add a vertex to the list"
{ pre: Graph = { vertexList = ∅ } }
{ post: vertexList = {x1 = vertex} }

getVertex (position)
"Return a vertex by its position on the list"
{ pre: Graph = { vertexList = ∅ } }
{ post:
if position on the list exist: Return {vertex on x-position
else return null}

insertArista (vstart, vfinal)
"Add an arist to a vertex of the vertext list"
{ pre: Graph = { vertexList = ∅ } }
{ post: {vstart.arist = new Arist (vfinal) } // vstart and vfinal has a common arist & vstart and vfinal are vertex }

order ()
"return the vertex list size"
{ pre: Graph = { vertexList = ∅ } }
{ post: return vertex.size}

sucesors (position)
"Return the sucesor of a vertex"
{ pre: Graph = { vertexList = ∅ } }
{ post: return vS / vS is a vertex & vS is the predecessor of the vertex in the given postion}

show ()
"Print the adjacent arista"
{ pre: Graph = { vertexList = ∅ } }
{ post: print each adjacent arista of each vertex}

conexMethod ()
"Return whether a vertex was visited"
{ pre: Graph = { vertexList = ∅ } }
{ post: return true or false}