TAD < GRAFO >

< abstract object > G = (V, E)

- Note that “V” is a set of vertices.
- “E” is a set of edges.
- $E \subseteq (V \times V)$, which are ordered pairs of vertices.

{ inv: $< \forall (v_i, v_j) \in E, (v_j, v_i) \in E >$

Primitive operations :

Constructor:

createGraph		→	Graph

Modifiers:

addVertex	Graph X Element	→	Graph
addEdge	Graph X Element X Element	→	Graph
removeVertex	Graph X Element	→	Graph
removeEdge	Graph X Element X Element	→	Graph

Analyzers:

isEmpty:	Graph	→	Boolean
isAdjacent	Graph X Element X Element	→	Boolean
isContained	Graph X Element	→	Boolean

createGraph

creates an empty Graph

{ pre: TRUE }
{ post: Graph = { nill } }

addVertex

adds a vertex to the graph

{ pre: Graph (G) }

{ post: element ∈ Graph (G) }

addEdge

Given a graph, add a relationship between two elements (vertices) of that graph.

{ pre: Graph (G) $\neq \emptyset \ \wedge \ V_1 \ \in V \ (vertices) \wedge V_2 \ \in V \ (vertices)$}

{ post: $(V_{1,} \, , V_2)$ ∈ E(edges) ∨ $(V_{2,}, V_{1,})$ ∈ E(edges)}

removeVertex

Removes a given element from the graph

{ pre: v ∈ V ∧ V ≠ ∅ }

{ post: v ∉ V ∧ v ∉ G }

removeEdge

Eliminate a relationship between two elements (vertices) of the graph.

{ pre: $(V_{1,} , V_2)$ ∈ E(edges) ∨ $(V_{2,}, V_{1,})$ ∈ E(edges) }
{ post: $(V_{1,} , V_2)$ ∉ E(edges) ∧ $(V_{2,}, V_{1,})$ ∉ E(edges) }

isEmpty

checks if a graph is empty

{ pre:}
{ post: TRUE : if it's empty ∨ FALSE : If it isn't empty }

isAdjacent

Checks whether two nodes have an edge that relates them

{ pre: V_1 ∈ V ∧ V_2∈ V ∧ $V_1 \neq V_2$}
{ post:True : if $(V_{1,} , V_2)$ ∈ E ∨ False: if $(V_{1,} , V_2)$ ∉ E }

isContained

checks whether an element is found or not in the graph

{ pre: $G \neq \emptyset$}

{ post: True: if the element $\in G$ $\vee$ False: if the element $\notin$ G }