|  |  |  |  |
| --- | --- | --- | --- |
| **TAD** Grafo [K] | | | |
| **Conjunto de elementos:**  (V, A) -> V = {v1, ... ,vn}, A = { ( v, w, c ) } | | | |
| {**inv**: ∀ (v, w, c) ∈ A: (v, w, c) ∈ A ⇒ (v ∈ V ∧ w ∈ V ∧ v → w ∧ c ≥ 0)  ∀ (v, w) ∈ A: ((v, w, c1) ∈ A ∧ (v, w, c2) ∈ A) ⇒ c1 = c2  } | | | |
| **Nombre** | **Entrada** | **Salida** | **Tipo de método** |
| Grafo(): void |  | * Grafo | Constructor |
| Insertar(): void | Grafo x K | * Grafo | Modificadora |
| Eliminar(): void | Grafo x int x int | * Grafo | Modificadora |
| Costo(): int | Grafo x int x int | * int | Analizadora |
| Sucesores(): Lista[ int] | Grafo x int | * Lista[int] | Analizadora |
| Orden(): int | Grafo | * int | Analizadora |

|  |
| --- |
| Grafo( ) : void  /\* Crea y retorna un grafo vacío \*/  { post: newGrafo = ( ) } |
| Insertar(Grafo g, TipoG elem ) : void  /\* Agrega un vértice al grafo con la información elem asociada \*/  { pre: g = ( { v1, ... , vn }, A ) }  { post: g = ( { v1, ... , vn, vn+1 }, A ), info( vn+1 ) = elem } |
| Eliminar( Grafo g, int x1, int x2 ): void  /\* Elimina del grafo el arco x1 x2 \*/  { pre: ( x1, x2, c ) A }  { post: g = ( V, A - ( x1, x2, c ) ) } |
| Costo( Grafo g, int x1, int x2 ): int  /\* Retorna el costo del arco x1 x2 si éste existe. En caso contrario retorna -1 \*/  { post: ( ( x1, x2, c ) A , costoArco = c ) ( ¬$c ½ ( x1, x2, c ) Î A , costoArco = -1 ) } |
| sucesores( Grafo g, int v ): Lista[int]  /\* Retorna una lista con los identificadores de los vértices sucesores de v \*/  { pre: v V }  { post: sucesores = < x1, ... , xk > | v xi } |
| Orden( Grafo g ) : int  /\* Retorna el número de vértices del grafo \*/  { post: ordenGrafo = n } |