Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

CI-2692 – Lab. Algoritmos y Estructuras II

Enero-Marzo 2014

PROYECTO #1

**Nombre:** Arleyn Goncalves , 10-10290

Francisco Sucre, 10-17107

**MODELO ABSTRACTO**

Especificación A del TAD Matriz (T)

Modelos de Representación

**const** num\_filas, num\_columnas : int

**var** Matrix = Seq (Seq T)

Invariante de Representacion

num\_filas >= 0 ˄ num\_columnas >= 0 ˄

# Matriz [ i ] <= num\_columnas ˄ # Matriz <= num\_filas

Operaciones

**proc** crear(**in** p, q: int ; **out** m: Matriz)

{Pre p >= 0 ˄ q >= 0 }

{Post m.num\_filas = p ˄ m.num\_columnas = q ˄ m.Matrix = [ ] }

**proc** obtener(**in-out** m: Matriz ; **in** fil, col: int ; **out** v: T)

{Pre 0 <= fil < m.num\_filas ˄ 0 <= col < m.num\_columnas}

{Post v = m.Matrix[ fil ] [ col ]}

**proc** colocar(**in-out** m: Matriz ; **in** fil, col, v: int )

{Pre 0 <= fil < m.num\_filas ˄ 0 <= col < m.num\_columnas }

{Post m.Matrix[ fil ] [ col ] = v }

**proc** sumar(**in** m1, m2: Matriz ; **out** mf: Matriz)

{Pre m1.num\_filas == m2.num\_filas ˄

m1. num\_columnas == m2. num\_columnas }

{Post mf.Matrix = <i,j : 0 <= i < m.num\_filas ˄

0 <= j< m.num\_columnas

: m1.Matrix[ i ][ j ] + m2.Matrix[ i ][ j ] > }

**proc** mult\_escalar(**in** m: Matriz ; **in** k : int ; **out** mf:Matriz)

{Pre True }

{Post mf.Matriz = <i,j : 0 <= i < m.num\_filas ˄

0 <= j< m.num\_columnas

: m.Matriz[ i ][ j ] \* k > }

**proc** trasponer(**in** m:Matriz ; **out**  mf: Matriz)

{Pre True }

{Post mf.Matriz = <i,j : 0 <= i < m.num\_filas ˄

0 <= j< m.num\_columnas

: m.Matriz[ j ][ i ] > }

**proc** multiplicar(**in** m1, m2 : Matriz ; **out** mf: Matriz )

{Pre m1.columna == m2.fila }

{Post }

**MODELO CONCRETO**

Especificación B del TAD Matriz (T), refinamiento A

Modelo de Representación

**const** num\_filas, num\_columnas: int

**var** Matrix: arreglo[0..num\_filas – 1) x arreglo[0..num\_columnas - 1) of T

Invariante de Representación

num\_filas >= 0 ˄ num\_columnas >= 0

Relación de Acoplamiento

contenido = {i,j : 0 <= num\_filas ˄ 0 <= num\_columnas : Matrix[ i ] [ j ] }

Operaciones

**proc** crear(**in** p, q: int ; **out** m: Matriz)

{Pre 0 <= p ˄ 0 <= q }

{Post m.num\_filas = p ˄ m.num\_columnas = q ˄ m.Matrix = [ ] }´

**proc** obtener(**in-out** m: Matriz ; **in** fil, col: int ; **out** v: T)

{Pre 0 <= fil <= m.num\_filas ˄ 0 <= col <= m.num\_columnas }

{Post v = m.Matrix[ fil ][ col ] }

**proc** colocar(**in-out** m: Matriz ; **in** fil, col, v: int )

{Pre 0 <= fil <= m.num\_filas ˄ 0 <= col <= m.num\_columnas }

{Post m.Matrix[ fi ][ co ] = v }

**proc** sumar(**in** m1, m2: Matriz ; **out** mf: Matriz)

{Pre m1.num\_filas = m2.num\_filas ˄

m1. num\_columnas = m2. num\_columnas }

{Post (∀i,j : 0 <= i < num\_filas ˄ 0 <= j < num\_columnas:

mf[ i ][ j ] = m1[ i ][ j ] + m2[ i ][ j ] ) }

**proc** mult\_escalar(**in** m: Matriz ; **in** k : int ; **out** mf: **Matriz**)

{Pre True }

{Post (∀i,j : 0 <= i < num\_filas ˄ 0 <= j < num\_columnas:

mf[ i ][ j ] = m1[ i ][ j ] \* k ) }

**proc** trasponer(**in** m:Matriz ; **out**  mf: Matriz)

{Pre True }

{Post (∀i,j : 0 <= i < num\_filas ˄ 0 <= j < num\_columnas:

mf[ i][ i ] = m1[ i ][ j ] )}

**proc** multiplicar(**in** m1, m2 : Matriz ; **out** mf: Matriz )

{Pre m1.num\_filas = m2.num\_filas ˄

m1. num\_columnas = m2. num\_columnas }

{Post }