

```

*****
*****                                Interfaz                                *****
*****

```

VectorPointer(α) es Vector(puntero(α))

Este iterador deberia estar en Vector

itVectorPointer(α)

Se explica con iterador unidireccional

Se considera n igual a longitud(vec).

```

*****
*****                                Operaciones                                *****
*****

```

CrearIt(in vec : VectorPointer(α)) \rightarrow res : itVectorPointer(α)

Pre \equiv {true}

Post \equiv {res =obs CrearItUni(in)}

Complejidad: $O(1)$

Descripcion: Crea un iterador unidireccional del VectorPointer(α).

HayMas?(in it : itVectorPointer(α)) \rightarrow res : bool

Pre \equiv {true}

Post \equiv {res =obs HayMas?(it)}

Complejidad: $O(n)$

Descripcion: Devuelve true si y solo si en el iterador quedan elementos para avanzar.

Actual(in it: itVectorPointer(α)) \rightarrow res : puntero(α)

Pre \equiv {HayMas?(it)}

Post \equiv {alias(res =obs Actual(it))}

Complejidad: $O(1)$

Descripcion: Devuelve el elemento actual del iterador.

Aliasing: res y su contenido no son modificables.

Avanzar(in/out it : itVectorPointer(α))

Pre \equiv {it = it₀ \wedge HayMas?(it)}

Post \equiv {it =obs Avanzar(it₀)}

Complejidad: $O(n)$

Descripcion: Avanza a la posicion siguiente del iterador.

```

*****
*****                                Representación                                *****
*****

```

itVectorPointer(α) se representa con iter(α),

donde iter(α) es: tupla(

actual : nat,

len : nat,

vec : puntero(VectorPointer(α)))

Rep: \wedge (iter(α) \rightarrow boolean

($\forall e : \wedge$ (iter(α)) Rep(e) \equiv true \iff

(e.actual < e.len) \wedge (\neg (e.vec =obs NULL) \wedge e.len = Longitud(*e.vec))

Abs: \wedge (iter(α)) i \rightarrow IteradorUnidireccional(α)

{Rep(i)}

($\forall e : \wedge$ (iter(α)) Abs(e) =obs m /

Longitud(Siguientes(m)) =obs (e.len - e.actual) \wedge

($\forall i : \text{nat}$) (i < e.len \wedge *e.vec[i] =obs Siguietes(m)[i])

```

*****
*****                               Algoritmos                               *****
*****

iCrearIt(in input_vec : VectorPointer( $\alpha$ )) → res: itVectorPointer( $\alpha$ )
    res ← {actual: 0, len: Longitud(input_vec), vec: &input_vec}
end function


iActual(in it: iter( $\alpha$ )) → res: puntero( $\alpha$ )
    res ← (it.vec)[iter.actual]
end function


iHayMas?(in it : iter( $\alpha$ )) → res : bool
    var next : nat ← it.actual

    while next < len ∧ (*it.vec)[next] == NULL do
        next ← next + 1
    end while

    res ← next != len
end function


iAvanzar(in/out it : iter( $\alpha$ ))
    var next : nat ← it.actual + 1

    while (*it.vec)[next] == NULL do
        next ← next + 1
    end while

    it.actual ← next
end function

```