```
*****************************************************************************
************                          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)) → res : itVectorPointer(α)
Pre ≡ {true}
Post ≡ {res =obs CrearItUni(in)}
Complejidad: O(1)
Descripcion: Crea un iterador unidireccional del VectorPointer(α).

HayMas?(in it : itVectorPointer(α)) → res : bool
Pre ≡ {true}
Post ≡ {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)) → res : puntero(α)
Pre ≡ {HayMas?(it)}
Post ≡ {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 ≡ {it = it₀ ∧ HayMas?(it)}
Post ≡ {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: ^(iter(α) → boolean
(∀e : ^(iter(α)) Rep(e) ≡ true ⟺
    (e.actual < e.len) ∧ (¬(e.vec =obs NULL) ∧ₗ e.len = Longitud(*e.vec))

Abs: ^(iter(α)) i → IteradorUnidireccional(α)                          {Rep(i)}
(∀e : ^(iter(α)) Abs(e) =obs m /
    Longitud(Siguientes(m)) =obs (e.len - e.actual) ∧
    (∀i : nat) (i < e.len ∧ₗ *e.vec[i] =obs Siguientes(m)[i])

*****************************************************************************
************                       Algoritmos                          ************
*****************************************************************************
iCrearIt(in input_vec : VectorPointer(α)) → res: itVectorPointer(α)
    res ← ⟨actual: 0, len: Longitud(input_vec), vec: &input_vec⟩
end function
```

```
iActual(in it: iter(α)) → res: puntero(α)
    res ← (it.vec)[iter.actual]
end function

iHayMas?(in it : iter(α)) → 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(α))
    var next : nat ← it.actual + 1

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

    it.actual ← next
end function
```