

## Trabajo Práctico Número 2

---

Algoritmos y Estructuras de Datos II

**Grupo: 21**

Integrante	LU	Correo electrónico
Langberg, Andrés	249/14	andreslangberg@gmail.com
Walter, Nicolás	272/14	nicowalter25@gmail.com
Sticco, Patricio Bernardo	337/14	pbsticco@hotmail.com
Len, Julián	467/14	julianlen@gmail.com



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

<http://www.fcen.uba.ar>

# 1. Diseño del Tipo CAMPUS

## 1.1. Especificación

Se usa el TAD CAMPUS especificado por la cátedra.

## 1.2. Aspectos de la interfaz

### 1.2.1. Interfaz

Se explica con especificación de CAMPUS

Género *campus*

Operaciones básicas de Campus

**CREARCAMPUS**(*in c: nat, in f: nat*)  $\rightarrow$  *res: campus*

**Pre**  $\equiv \{ true \}$

**Post**  $\equiv \{ res =_{\text{obs}} \text{crearCampus}(c, f) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Crea un campus de c columnas y f filas.

**FILAS?**(*in c: campus*)  $\rightarrow$  *res: nat*

**Pre**  $\equiv \{ true \}$

**Post**  $\equiv \{ res =_{\text{obs}} \text{filas}(c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve la cantidad de filas en el campus.

**COLUMNAS?**(*in c: campus*)  $\rightarrow$  *res: nat*

**Pre**  $\equiv \{ true \}$

**Post**  $\equiv \{ res =_{\text{obs}} \text{columnas}(c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve la cantidad de columnas en el campus.

**OCUPADA?**(*in c: campus, in p: posicion*)  $\rightarrow$  *res: bool*

**Pre**  $\equiv \{ \text{posValida}(p, c) \}$

**Post**  $\equiv \{ res =_{\text{obs}} \text{ocupada?}(p, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve *true* sii p esta ocupada por un obstaculo.

**AGREGAROBSTACULO**(*in/out c: campus, in p: posicion*)  $\rightarrow$

**Pre**  $\equiv \{ c =_{\text{obs}} c_0 \wedge \text{posValida}(p, c) \wedge_L \neg \text{ocupada?}(p, c) \}$

**Post**  $\equiv \{ c =_{\text{obs}} \text{agregarObstaculo}(p, c_0) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve *true* sii p esta ocupada por un obstaculo.

**POSVALIDA?**(*in c: campus, in p: posicion*)  $\rightarrow$  *res: bool*

**Pre**  $\equiv \{ true \}$

**Post**  $\equiv \{ res =_{\text{obs}} \text{posValida?}(p, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve *true* sii p es parte del mapa.

**ESINGRESO?**(*in c: campus, in p: posicion*)  $\rightarrow$  *res: bool*

**Pre**  $\equiv \{ true \}$

**Post**  $\equiv \{ res =_{\text{obs}} \text{esIngreso?}(p, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve *true* sii p es un ingreso.

VECINOS(**in** *c: campus*, **in** *p: posicion*)  $\rightarrow res : conj(posicion)$

**Pre**  $\equiv \{ posValida(p, c) \}$

**Post**  $\equiv \{ res =_{obs} vecinos(p, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve el conjunto de posiciones vecinas a p.

VECINOSCOMUNES(**in** *c: campus*, **in** *p: posicion*, **in** *p2: posicion*)  $\rightarrow res : conj(posicion)$

**Pre**  $\equiv \{ posValida(p, c) \wedge posValida(p2, c) \}$

**Post**  $\equiv \{ res =_{obs} vecinos(p, c) \cap vecinos(p2, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve el conjunto de vecinos comunes. La complejidad es  $\mathcal{O}(1)$  dado que los vecinos son a lo sumo 4, o sea, constantes

VECINOSCOMUNES(**in** *c: campus*, **in** *p: posicion*, **in** *p2: posicion*)  $\rightarrow res : conj(posicion)$

**Pre**  $\equiv \{ posValida(p, c) \wedge posValida(p2, c) \}$

**Post**  $\equiv \{ res =_{obs} vecinos(p, c) \cap vecinos(p2, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve el conjunto de vecinos comunes entre dos posiciones. La complejidad es  $\mathcal{O}(1)$  dado que los vecinos son a lo sumo 4, o sea, constantes.

PROXPOSICION(**in** *c: campus*, **in** *dir: direccion*, **in** *p: posicion*)  $\rightarrow res : posicion$

**Pre**  $\equiv \{ posValida(p, c) \}$

**Post**  $\equiv \{ res =_{obs} proxPosicion(p, d, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve la posicion vecina a p que esta en la direccion dir.

INGRESOSMASCERCANOS(**in** *c: campus*, **in** *p: posicion*)  $\rightarrow res : conj(posicion)$

**Pre**  $\equiv \{ posValida(p, c) \}$

**Post**  $\equiv \{ res =_{obs} ingresosMasCercanos(p, c) \}$

**Complejidad:**  $\mathcal{O}(1)$

**Descripción:** Devuelve el conjunto de ingresos mas cercanos a p.

### 1.3. Pautas de implementación

#### 1.3.1. Estructura de representación

*campus* se representa con *estr*  
 donde *estr* es  
*tupla*(  
 filas: *nat*  $\times$   
 columnas: *nat*  $\times$   
 mapa: *vector(vector(bool))*  
 )

#### 1.3.2. Justificación

#### 1.3.3. Invariante de Representación

**Informal**

1. El mapa debe tener tantas filas como indica la estructura, lo mismo con las columnas.

**Formal**

Rep : *estr*  $\rightarrow$  *boolean*

( $\forall e : estr$ )

$\text{Rep}(e) \equiv (\text{true} \iff$   
 (1)  $e.\text{filas} = \text{longitud}(e.\text{mapa}) \wedge_L (\forall i : \text{nat})(i \leq e.\text{filas} \Rightarrow \text{longitud}(e.\text{mapa}[i]) = e.\text{columnas}))$

#### 1.3.4. Función de Abstracción

$\text{Abs} : \text{estr } e \longrightarrow \text{campus}$   $\{\text{Rep}(e)\}$   
 $(\forall e : \text{estr}) \text{ Abs}(e) =_{\text{obs}} c : \text{campus} /$   
 $\left( \text{filas}(c) = e.\text{filas} \wedge \text{columnas}(c) = e.\text{columnas} \wedge_L (\forall p : \text{posicion})(p.X \leq e.\text{filas} \wedge \right.$   
 $\left. p.Y \leq e.\text{columnas} \Rightarrow_L \text{ocupada?}(p, c) \Leftrightarrow (e.\text{mapa}[p.X])[p.Y]) \right)$

**1.3.5. Algoritmos**


---

```

1: function iCREARCAMPUS(in c: nat, in f: nat)  $\longrightarrow$  res: estr  $\triangleright \mathcal{O}(f^2 * c^2)$ 
2:   var vector(vector(bool)) mapa  $\leftarrow$  vacia(vacia())  $\triangleright \mathcal{O}(1)$ 
3:   var nat i  $\leftarrow$  0  $\triangleright \mathcal{O}(1)$ 
4:   while i  $\leq$  f do  $\triangleright \mathcal{O}(f)$ 
5:     var vector(bool) nuevo  $\leftarrow$  vacia()  $\triangleright \mathcal{O}(1)$ 
6:     var nat j  $\leftarrow$  0  $\triangleright \mathcal{O}(1)$ 
7:     while j  $\leq$  c do  $\triangleright \mathcal{O}(c)$ 
8:       AgregarAtras(nuevo, false)  $\triangleright \mathcal{O}(c)$ 
9:       j++  $\triangleright \mathcal{O}(1)$ 
10:    end while
11:    AgregarAtras(mapa, nuevo)  $\triangleright \mathcal{O}(f)$ 
12:    i++  $\triangleright \mathcal{O}(1)$ 
13:  end while
14:  res  $\leftarrow$  < f, c, mapa >  $\triangleright \mathcal{O}(1)$ 
15: end function

```

---

```

1: function iAGREGAROBSTACULO(in/out e: estr, in p: posicion)  $\longrightarrow$  res: estr  $\triangleright \mathcal{O}(\text{longitud}(e.\text{mapa}[p.X]))$ 
2:   Agregar(e.mapa[p.X], p.Y, true)  $\triangleright \mathcal{O}(\text{longitud}(e.\text{mapa}[p.X]))$ 
3: end function

```

---

```

1: function iFILAS?(in e: estr)  $\longrightarrow$  res: nat  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  e.filas  $\triangleright \mathcal{O}(1)$ 
3: end function

```

---

---

```

1: function iCOLUMNAS?(in e: estr)  $\longrightarrow$  res : nat  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  e.columns  $\triangleright \mathcal{O}(1)$ 
3: end function

```

---

```

1: function iOCUPADA?(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  (e.mapa[p.X])[p.Y]  $\triangleright \mathcal{O}(1)$ 
3: end function

```

---

```

1: function iPosVALIDA?(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  ( $0 < p.X$ )  $\wedge$  ( $p.X \leq e.filas$ )  $\wedge$  ( $0 < p.Y$ )  $\wedge$  ( $p.Y \leq e.columns$ )  $\triangleright \mathcal{O}(1)$ 
3: end function

```

---

```

1: function iESINGRESO?(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  ( $p.Y = 1$ )  $\vee$  ( $p.Y = e.filas$ )  $\triangleright \mathcal{O}(1)$ 
3: end function

```

---

```

1: function iVECINOS(in e: estr, in p: posicion)  $\longrightarrow$  res : conj(posicion)  $\triangleright \mathcal{O}(1)$ 
2:   var conj(posicion) nuevo  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
3:   Agregar(nuevo, (p.X+1,p.Y))
4:   Agregar(nuevo, (p.X-1,p.Y))
5:   Agregar(nuevo, (p.X,p.Y+1))
6:   Agregar(nuevo, (p.X,p.Y-1))
7:   var itConj(posicion) it  $\leftarrow$  crearIt(nuevo)
8:   while haySiguiente(it) do  $\triangleright \mathcal{O}(c)$ 
9:     if iPosValida?(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
10:      avanzar(it)  $\triangleright \mathcal{O}(1)$ 
11:    else
12:      eliminarSiguiente(it)  $\triangleright \mathcal{O}(1)$ 
13:    end if
14:  end while
15:  res  $\leftarrow$  nuevo  $\triangleright \mathcal{O}(1)$ 
16: end function

```

---

```

1: function iVECINOSCOMUNES(in e: estr, in p: posicion, in p2: posicion)  $\longrightarrow$  res : conj(posicion)  $\triangleright \mathcal{O}(1)$ 
2:   var conj(posicion) v  $\leftarrow$  vecinos(e,p)  $\triangleright \mathcal{O}(1)$ 
3:   var conj(posicion) v2  $\leftarrow$  vecinos(e,p2)  $\triangleright \mathcal{O}(1)$ 
4:   var conj(posicion) nuevo  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
5:   var itConj(posicion) it  $\leftarrow$  crearIt(v)  $\triangleright \mathcal{O}(1)$ 
6:   while haySiguiente(it) do  $\triangleright \mathcal{O}(1)$ 
7:     if Pertenece?(v2,Siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
8:       Agregar(nuevo, Siguiente(it))  $\triangleright \mathcal{O}(1)$ 
9:     end if
10:    Avanzar(it)  $\triangleright \mathcal{O}(1)$ 
11:  end while
12:  res  $\leftarrow$  nuevo  $\triangleright \mathcal{O}(1)$ 
13: end function

```

---

---

```

1: function iVECINOSVALIDOS(in e: estr, in ps: conj(posicion))  $\rightarrow$  res : conj(posicion)  $\triangleright \mathcal{O}(1)$ 
2:   var conj(posicion) nuevo  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
3:   var itConj(posicion) it  $\leftarrow$  crearIt(ps)  $\triangleright \mathcal{O}(1)$ 
4:   while haySiguiente(it) do  $\triangleright \mathcal{O}(1)$ 
5:     if PosValida?(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
6:       Agregar(nuevo, siguiente(it))  $\triangleright \mathcal{O}(1)$ 
7:     end if
8:     avanzar(it)  $\triangleright \mathcal{O}(1)$ 
9:   end while
10:  res  $\leftarrow$  nuevo  $\triangleright \mathcal{O}(1)$ 
11: end function

```

---

```

1: function iDISTANCIA(in e: estr, in p: posicion, in p2: posicion)  $\rightarrow$  res : nat  $\triangleright \mathcal{O}(1)$ 
2:  res  $\leftarrow$  |p.X - p2.X| + |p.Y - p2.Y|  $\triangleright \mathcal{O}(1)$ 
3: end function

```

---

```

1: function iPROXPOSICION(in e: estr, in d: direccion, in p: posicion)  $\rightarrow$  res : posicion  $\triangleright \mathcal{O}(1)$ 
2:  var posicion p2  $\leftarrow$  p
3:  if d==izq then
4:    p2  $\leftarrow$  <p2.X+1, p2.Y>  $\triangleright \mathcal{O}(1)$ 
5:  else
6:    if d==der then
7:      p2  $\leftarrow$  <p2.X, p2.Y>  $\triangleright \mathcal{O}(1)$ 
8:    else
9:      if d==arriba then  $\triangleright \mathcal{O}(1)$ 
10:        p2  $\leftarrow$  <p2.X, p2.Y-1>  $\triangleright \mathcal{O}(1)$ 
11:      else
12:        p2  $\leftarrow$  <p2.X, p2.Y+1>  $\triangleright \mathcal{O}(1)$ 
13:      end if
14:    end if
15:  end if
16:  res  $\leftarrow$  p2  $\triangleright \mathcal{O}(1)$ 
17: end function

```

---

```

1: function iINGRESOSMASCERCANOS(in e: estr, in p: posicion)  $\rightarrow$  res : conj(posicion)  $\triangleright \mathcal{O}(1)$ 
2:  var conj(posicion) nuevo  $\leftarrow$  Vacio()  $\triangleright \mathcal{O}(1)$ 
3:  if distancia(e, p, <p.x,1>) < distancia(e, p, <p.x,e.filas>) then  $\triangleright \mathcal{O}(1)$ 
4:    Agregar(nuevo, <p.x,1>)  $\triangleright \mathcal{O}(1)$ 
5:  else
6:    if distancia(e, p, <p.x,1>) > distancia(e, p, <p.x,e.filas(e)>) then  $\triangleright \mathcal{O}(1)$ 
7:      Agregar(nuevo, <p.x,e.filas>)  $\triangleright \mathcal{O}(1)$ 
8:    else
9:      Agregar(nuevo, <p.x,1>)  $\triangleright \mathcal{O}(1)$ 
10:     Agregar(nuevo, <p.x,e.filas>)  $\triangleright \mathcal{O}(1)$ 
11:    end if
12:  end if
13:  res  $\leftarrow$  nuevo  $\triangleright \mathcal{O}(1)$ 
14: end function

```

---