Introducción a la Computación (Matemática)

Primer Cuatrimestre de 2018

Tipos Abstractos de Datos. Implementación.

TAD Fecha

Operaciones públicas (para f, f_1, f_2 : Fecha; d, m, a : $\mathbb{Z}$):

- CrearFecha(d, m, a) $\rightarrow$ Fecha
- f.Día() $\rightarrow \mathbb{Z}$
- f.Mes() $\rightarrow \mathbb{Z}$
- f.Año() $\rightarrow \mathbb{Z}$
- f_1.Menor(f_2) $\rightarrow \mathbb{B}$
- f.FechaSiguiente() $\rightarrow$ Fecha

TAD Fecha

Operaciones públicas (para f, f_1, f_2 : Fecha; $d, m, a : \mathbb{Z}$):

- CrearFecha(d, m, a) $\rightarrow$ Fecha
- f.Día() $\rightarrow \mathbb{Z}$
- f.Mes() $\rightarrow \mathbb{Z}$
- f.Año() $\rightarrow \mathbb{Z}$
- f_1.Menor(f_2) $\rightarrow \mathbb{B}$
- f.FechaSiguiente() $\rightarrow$ Fecha

Con esto, un usuario del TAD puede programar algo como:

ContarDías(f_1, f_2) $\rightarrow \mathbb{Z}$:
  $RV \leftarrow 0$
  while (f_1.Menor(f, f_2)) {
    $RV \leftarrow RV + 1$
    $f_1 \leftarrow f_1$.FechaSiguiente()
  }

TAD Fecha - Posible implementación

TAD Fecha - Posible implementación

Primero elegimos una estructura de representación para el TAD Fecha (que es **privada**).

TAD Fecha - Posible implementación

Primero elegimos una estructura de representación para el TAD Fecha (que es **privada**).

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

$\langle ... \rangle$ define una tupla.

TAD Fecha - Posible implementación

Primero elegimos una estructura de representación para el TAD Fecha (que es **privada**).

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

$\langle ... \rangle$ define una tupla.

Después describimos el invariante de representación de esta estructura.

TAD Fecha - Posible implementación

Primero elegimos una estructura de representación para el TAD Fecha (que es **privada**).

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

$\langle ... \rangle$ define una tupla.

Después describimos el invariante de representación de esta estructura.

$$\textit{DMAVálidos}(\textit{día}, \textit{mes}, \textit{año})$$

donde:

$$\begin{aligned} \textit{DMAVálidos}(d, m, a) \equiv\ & \big(1 \leq d \leq 31\ \wedge\ (m = 1 \vee m = 3 \vee \dots)\big) \vee \\ & \big(1 \leq d \leq 30\ \wedge\ (m = 4 \vee m = 6 \vee \dots)\big) \vee \\ & \big(1 \leq d \leq 29\ \wedge\ m = 2 \wedge Bisiesto(a)\big) \vee \\ & \big(1 \leq d \leq 28\ \wedge\ m = 2 \wedge \neg Bisiesto(a)\big) \end{aligned}$$

$$Bisiesto(a) \equiv \big(a \text{ div } 4 = 0 \wedge (a \text{ div } 100 \neq 0 \vee a \text{ div } 400 = 0)\big)$$

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Por último, damos los algoritmos de las operaciones del TAD Fecha para la estructura de representación elegida:

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Por último, damos los algoritmos de las operaciones del TAD Fecha para la estructura de representación elegida:

CrearFecha$(d, m, a) \rightarrow$ Fecha: (Pre: *DMAVálidos*(d, m, a))

TAD Fecha - Posible implementación

Fecha $==$ $\langle$*día*: $\mathbb{Z}$, *mes*: $\mathbb{Z}$, *año*: $\mathbb{Z}\rangle$

Por último, damos los algoritmos de las operaciones del TAD Fecha para la estructura de representación elegida:

CrearFecha$(d, m, a) \rightarrow$ Fecha: (Pre: *DMAVálidos*(d, m, a))

RV.*día* $\leftarrow d$
RV.*mes* $\leftarrow m$
RV.*año* $\leftarrow a$

donde $d, m, a : \mathbb{Z}$.

TAD Fecha - Posible implementación

Fecha $==$ ⟨*día*: $\mathbb{Z}$, *mes*: $\mathbb{Z}$, *año*: $\mathbb{Z}$⟩

Por último, damos los algoritmos de las operaciones del TAD Fecha para la estructura de representación elegida:

CrearFecha$(d, m, a) \rightarrow$ Fecha: (Pre: *DMAVálidos*(d, m, a))

RV.*día* $\leftarrow d$
RV.*mes* $\leftarrow m$
RV.*año* $\leftarrow a$

donde $d, m, a : \mathbb{Z}$.

Operaciones como esta se conocen como constructores. Permiten armar elementos del TAD.

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

f.Día() $\rightarrow \mathbb{Z}$:

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

f.Día() $\rightarrow \mathbb{Z}$:

$RV \leftarrow f.\textit{día}$

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}\colon \mathbb{Z},\ \textit{mes}\colon \mathbb{Z},\ \textit{año}\colon \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

f.Día() $\to \mathbb{Z}$:

$RV \leftarrow f.\textit{día}$

f.Mes() $\to \mathbb{Z}$:

$RV \leftarrow f.\textit{mes}$

f.Año() $\to \mathbb{Z}$:

$RV \leftarrow f.\textit{año}$

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

f.Día() $\to \mathbb{Z}$:

$RV \leftarrow f.\textit{día}$

f.Mes() $\to \mathbb{Z}$:

$RV \leftarrow f.\textit{mes}$

f.Año() $\to \mathbb{Z}$:

$RV \leftarrow f.\textit{año}$

Estas operaciones se conocen como proyectores.
Permiten inspeccionar elementos de un TAD.

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f_1, f_2: Fecha):

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f_1, f_2: Fecha):

f_1.Menor(f_2) $\rightarrow \mathbb{B}$:

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z},\ \textit{mes}: \mathbb{Z},\ \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f_1, f_2: Fecha):

f_1.Menor(f_2) $\rightarrow \mathbb{B}$:

```
if (f1.Año() < f2.Año()) {
   RV ← TRUE
}
elsif (f1.Año() = f2.Año() ∧ f1.Mes() < f2.Mes()) {
   RV ← TRUE
}
elsif (f1.Año() = f2.Año() ∧ f1.Mes() = f2.Mes() ∧ f1.Día() < f2.Día())
   RV ← TRUE
}
else {
   RV ← FALSE
}
```

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

f.FechaSiguiente() $\rightarrow$ Fecha:

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha):

f.FechaSiguiente() $\to$ Fecha:

```
d  ← f.Día() + 1
m  ← f.Mes()
a  ← f.Año()
if (d > CantidadDeDías(m, a)) {
   d ← 1
   m ← m + 1
   if (m > 12) {
      m ← 1
      a ← a + 1
   }
}
RV ← CrearFecha(d, m, a)
```

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z}, \textit{mes}: \mathbb{Z}, \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha; $m, a : \mathbb{Z}$):

CantidadDeDías(m, a) $\rightarrow \mathbb{Z}$: (Pre: $1 \leq m \leq 12$)

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z},\ \textit{mes}: \mathbb{Z},\ \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha; $m, a : \mathbb{Z}$):

CantidadDeDías$(m, a) \to \mathbb{Z}$: (Pre: $1 \leq m \leq 12$)

if $(m = 1 \vee m = 3 \vee m = 5 \vee m = 7 \vee m = 8 \vee m = 10 \vee m = 12)$ {
  $RV \leftarrow 31$
} elsif $(m = 4 \vee m = 6 \vee m = 9 \vee m = 11)$ {
  $RV \leftarrow 30$
} elsif $(m = 2 \wedge \text{EsBisiesto}(a))$ {
  $RV \leftarrow 29$
} else {
  $RV \leftarrow 28$
}

TAD Fecha - Posible implementación

$$\text{Fecha} == \langle \textit{día}: \mathbb{Z},\ \textit{mes}: \mathbb{Z},\ \textit{año}: \mathbb{Z} \rangle$$

Más operaciones del TAD Fecha (para f: Fecha; $m, a : \mathbb{Z}$):

CantidadDeDías$(m, a) \to \mathbb{Z}$: (Pre: $1 \leq m \leq 12$)

if $(m = 1 \vee m = 3 \vee m = 5 \vee m = 7 \vee m = 8 \vee m = 10 \vee m = 12)$ {
  $RV \leftarrow 31$
} elsif $(m = 4 \vee m = 6 \vee m = 9 \vee m = 11)$ {
  $RV \leftarrow 30$
} elsif $(m = 2 \wedge \text{EsBisiesto}(a))$ {
  $RV \leftarrow 29$
} else {
  $RV \leftarrow 28$
}

Podemos elegir que CantidadDeDías y EsBisiesto sean operaciones privadas: no accesibles para usuarios del TAD Fecha.

Partes de un Tipo Abstracto de Datos

- Parte pública: Disponible para el usuario externo.
    - Nombre y tipos paramétricos (ej: Fecha, Lista(ELEM)).
    - Operaciones, sus especificaciones y posiblemente sus órdenes de complejidad temporal.

Partes de un Tipo Abstracto de Datos

- Parte pública: Disponible para el usuario externo.
  - Nombre y tipos paramétricos (ej: Fecha, Lista(ELEM)).
  - Operaciones, sus especificaciones y posiblemente sus órdenes de complejidad temporal.

- Parte privada: Sólo accesible desde dentro del TAD. ¡El usuario externo **nunca** debe ver ni meterse en esto!
  - **Estructura de representación** del TAD.
  - **Invariante de representación**: qué propiedades debe cumplir la estructura elegida para que tenga sentido como la representación de una instancia del TAD.
  - **Algoritmos de las operaciones** para la estructura de representación elegida.

TAD Lista(ELEM)

- CrearLista() $\rightarrow$ Lista(ELEM)
- L.Agregar(x)
- L.Longitud$()$ $\rightarrow \mathbb{Z}$
- L.Iésimo(i) $\rightarrow ELEM$
- L.Cantidad(x) $\rightarrow \mathbb{Z}$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice(x) $\rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

TAD Lista(ELEM) - Posible implementación

Vamos a representar una lista como una cadena de nodos.

TAD Lista(ELEM) - Posible implementación

Vamos a representar una lista como una cadena de nodos.
Definamos primero qué es un nodo.

Nodo(ELEM) : $\langle valor : \text{ELEM},\ siguiente : \text{Ref(Nodo)}\rangle$

TAD Lista(ELEM) - Posible implementación

Vamos a representar una lista como una cadena de nodos. Definamos primero qué es un nodo.

Nodo(ELEM) : $\langle valor$: ELEM, $siguiente$: Ref(Nodo)$\rangle$

$\langle ... \rangle$ define una tupla.

TAD Lista(ELEM) - Posible implementación

Vamos a representar una lista como una cadena de nodos. Definamos primero qué es un nodo.

Nodo(ELEM) : $\langle valor$: ELEM, $siguiente$: Ref(Nodo)$\rangle$

$\langle ... \rangle$ define una tupla.

Ref(Nodo) es una referencia a una instancia de tipo Nodo. Puede tomar el valor especial $None$ ("referencia a nada").

TAD Lista(ELEM) - Posible implementación

Vamos a representar una lista como una cadena de nodos. Definamos primero qué es un nodo.

Nodo(ELEM) : $\langle valor$: ELEM, $siguiente$: Ref(Nodo)$\rangle$

$\langle ... \rangle$ define una tupla.

Ref(Nodo) es una referencia a una instancia de tipo Nodo. Puede tomar el valor especial $None$ ("referencia a nada").

Construcción de un nuevo Nodo:

- $n \leftarrow$ Nodo(x, r)

TAD Lista(ELEM) - Posible implementación

Vamos a representar una lista como una cadena de nodos. Definamos primero qué es un nodo.

Nodo(ELEM) : $\langle valor$: ELEM, $siguiente$: Ref(Nodo)$\rangle$

$\langle ... \rangle$ define una tupla.

Ref(Nodo) es una referencia a una instancia de tipo Nodo. Puede tomar el valor especial $None$ ("referencia a nada").

Construcción de un nuevo Nodo:

- $n \leftarrow$ Nodo(x, r)

Acceso a los campos de un nodo n:

- $n.valor$ devuelve el campo $valor$.
- $n.siguiente$ devuelve el campo $siguiente$.

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

$$\text{Lista(ELEM)} == \langle cabeza : \text{Ref(Nodo)},\ longitud : \mathbb{Z} \rangle$$

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

$$\text{Lista(ELEM)} == \langle cabeza : \text{Ref(Nodo)},\ longitud : \mathbb{Z} \rangle$$

Después describimos el invariante de representación de esta estructura.

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

$$\text{Lista(ELEM)} == \langle cabeza : \text{Ref(Nodo)},\ longitud : \mathbb{Z}\rangle$$

Después describimos el invariante de representación de esta estructura. En este caso $longitud$ siempre debe ser igual a la cantidad de nodos encadenados a partir de $cabeza$, y en la cadena de nodos no deben formarse ciclos.

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

$$\text{Lista(ELEM)} == \langle cabeza : \text{Ref(Nodo)},\ longitud : \mathbb{Z}\rangle$$

Después describimos el invariante de representación de esta estructura. En este caso $longitud$ siempre debe ser igual a la cantidad de nodos encadenados a partir de $cabeza$, y en la cadena de nodos no deben formarse ciclos.

Por último, damos los algoritmos de las operaciones del TAD Lista(ELEM) para la estructura de representación elegida:

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

$$\text{Lista(ELEM)} == \langle cabeza : \text{Ref(Nodo)},\ longitud : \mathbb{Z} \rangle$$

Después describimos el invariante de representación de esta estructura. En este caso $longitud$ siempre debe ser igual a la cantidad de nodos encadenados a partir de $cabeza$, y en la cadena de nodos no deben formarse ciclos.

Por último, damos los algoritmos de las operaciones del TAD Lista(ELEM) para la estructura de representación elegida:

CrearLista() $\rightarrow$ Lista(ELEM):

TAD Lista(ELEM) - Posible implementación

Primero elegimos una estructura de representación del TAD Lista(ELEM) (que es **privada**).

$$\text{Lista(ELEM)} == \langle cabeza : \text{Ref(Nodo)},\ longitud : \mathbb{Z} \rangle$$

Después describimos el invariante de representación de esta estructura. En este caso $longitud$ siempre debe ser igual a la cantidad de nodos encadenados a partir de $cabeza$, y en la cadena de nodos no deben formarse ciclos.

Por último, damos los algoritmos de las operaciones del TAD Lista(ELEM) para la estructura de representación elegida:

CrearLista() $\rightarrow$ Lista(ELEM):

$RV.cabeza \leftarrow None$

$RV.longitud \leftarrow 0$

TAD Lista(ELEM) - Posible implementación

$L.\text{Agregar}(x)$:

TAD Lista(ELEM) - Posible implementación

L.Agregar(x):

```
nuevo ← Nodo(x, None)
if (L.cabeza = None) {
    L.cabeza ← nuevo
}
else {
    aux ← L.cabeza
    while (aux.siguiente ≠ None) {
        aux ← aux.siguiente
    }
    aux.siguiente ← nuevo
}
L.longitud ← L.longitud + 1
```

TAD Lista(ELEM) - Posible implementación

L.Longitud() $\rightarrow \mathbb{Z}$:

TAD Lista(ELEM) - Posible implementación

L.Longitud() $\rightarrow \mathbb{Z}$:

$RV \leftarrow L.longitud$

L.Iésimo(i) $\rightarrow ELEM$: (Pre: $0 \leq i < L$.Longitud())

TAD Lista(ELEM) - Posible implementación

L.Longitud() $\rightarrow \mathbb{Z}$:

$RV \leftarrow L.longitud$

L.Iésimo(i) $\rightarrow ELEM$: (Pre: $0 \leq i < L$.Longitud())

$aux \leftarrow L.cabeza$
while $(i > 0)$ {
$aux \leftarrow aux.siguiente$
$i \leftarrow i - 1$
}
$RV \leftarrow aux.valor$

Y así con las otras operaciones del TAD Lista(ELEM): Cantidad, Insertar, Indice y BorrarIésimo

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM)
- L.Agregar(x)
- L.Longitud$()$ $\rightarrow \mathbb{Z}$
- L.Iésimo(i) $\rightarrow ELEM$
- L.Cantidad(x) $\rightarrow \mathbb{Z}$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice(x) $\rightarrow \mathbb{Z}$

donde L : Lista(ELEM), i : $\mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $\quad O(1)$
- L.Agregar(x)
- L.Longitud$()$ $\rightarrow \mathbb{Z}$
- L.Iésimo$(i) \rightarrow ELEM$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $O(1)$
- L.Agregar(x) $O(n)$
- L.Longitud$()$ $\rightarrow \mathbb{Z}$
- L.Iésimo$(i) \rightarrow ELEM$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $\quad O(1)$
- L.Agregar(x) $\quad O(n)$
- L.Longitud$()$ $\rightarrow \mathbb{Z}$ $\quad O(1)$
- L.Iésimo$(i) \rightarrow ELEM$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $\quad O(1)$
- L.Agregar(x) $\quad O(n)$
- L.Longitud$()$ $\rightarrow \mathbb{Z}$ $\quad O(1)$
- L.Iésimo$(i) \rightarrow ELEM$ $\quad O(n)$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $\quad O(1)$
- L.Agregar(x) $\quad O(n)$
- L.Longitud$()$ $\rightarrow \mathbb{Z}$ $\quad O(1)$
- L.Iésimo$(i) \rightarrow ELEM$ $\quad O(n)$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$ $\quad O(n)$
- L.Insertar(i, x)
- L.BorrarIésimo(i)
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $O(1)$
- L.Agregar(x) $O(n)$
- L.Longitud() $\rightarrow \mathbb{Z}$ $O(1)$
- L.Iésimo$(i) \rightarrow ELEM$ $O(n)$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$ $O(n)$
- L.Insertar(i, x) $O(n)$
- L.BorrarIésimo(i)
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $O(1)$
- L.Agregar(x) $O(n)$
- L.Longitud$()$ $\rightarrow \mathbb{Z}$ $O(1)$
- L.Iésimo$(i) \rightarrow ELEM$ $O(n)$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$ $O(n)$
- L.Insertar(i, x) $O(n)$
- L.BorrarIésimo(i) $O(n)$
- L.Indice$(x) \rightarrow \mathbb{Z}$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Lista(ELEM) - Posible implementación

Complejidad temporal de los algoritmos vistos para esta estructura de representación.

- CrearLista() $\rightarrow$ Lista(ELEM) $\quad O(1)$
- L.Agregar(x) $\quad O(n)$
- L.Longitud$()$ $\rightarrow \mathbb{Z}$ $\quad O(1)$
- L.Iésimo$(i) \rightarrow ELEM$ $\quad O(n)$
- L.Cantidad$(x) \rightarrow \mathbb{Z}$ $\quad O(n)$
- L.Insertar(i, x) $\quad O(n)$
- L.BorrarIésimo(i) $\quad O(n)$
- L.Indice$(x) \rightarrow \mathbb{Z}$ $\quad O(n)$

donde L : Lista(ELEM), $i : \mathbb{Z}$, x : ELEM (entero, char, etc.).

TAD Pila(ELEM)

Operaciones:

- CrearPila() $\rightarrow$ Pila(ELEM): Crea una pila vacía.
- P.Apilar(x): Inserta el elem. x sobre el tope de la pila P.
- P.Vacía?() $\rightarrow \mathbb{B}$: Dice si la pila P está vacía.
- P.Tope() $\rightarrow ELEM$: Devuelve el elemento del tope de P.
  Precondición: $\neg P$.Vacía?().
- P.Desapilar(): Borra el elemento del tope de P.
  Precondición: $\neg P$.Vacía?().

donde P : Pila(ELEM), x : ELEM (entero, char, etc.).

Las pilas tienen estrategia LIFO (*last in, first out*).

TAD Pila(ELEM) - Posible implementación

TAD Pila(ELEM) - Posible implementación

Estructura de representación del TAD Pila(ELEM):

$$\text{Pila(ELEM)} == \langle milista : \text{Lista(ELEM)} \rangle$$

TAD Pila(ELEM) - Posible implementación

Estructura de representación del TAD Pila(ELEM):

$$\text{Pila(ELEM)} == \langle milista : \text{Lista(ELEM)} \rangle$$

Invariante de representación de esta estructura:

TAD Pila(ELEM) - Posible implementación

Estructura de representación del TAD Pila(ELEM):

$$\text{Pila(ELEM)} == \langle milista : \text{Lista(ELEM)}\rangle$$

Invariante de representación de esta estructura:
En este caso no hay nada que decir. Según la estructura de representación elegida, cualquier lista es una representación válida de una pila.

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):

$RV.milista \leftarrow CrearLista()$

P.Apilar(x):

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):

$RV.milista \leftarrow CrearLista()$

P.Apilar(x):

$P.milista$.Agregar(x)

P.Vacía?$() \rightarrow \mathbb{B}$:

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):

$RV.milista \leftarrow CrearLista()$

P.Apilar(x):

$P.milista.$Agregar(x)

P.Vacía?() $\rightarrow \mathbb{B}$:

$RV \leftarrow (P.milista.$Longitud$() = 0)$

P.Tope() $\rightarrow ELEM$: (Pre: $\neg P$.Vacía?())

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):

$RV.milista \leftarrow CrearLista()$

P.Apilar(x):

$P.milista.\text{Agregar}(x)$

P.Vacía?() $\rightarrow \mathbb{B}$:

$RV \leftarrow (P.milista.\text{Longitud}() = 0)$

P.Tope() $\rightarrow ELEM$: (Pre: $\neg P$.Vacía?())

$RV \leftarrow P.milista.\text{Iésimo}(P.milista.\text{Longitud}() - 1)$

P.Desapilar(): (Pre: $\neg P$.Vacía?())

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):

$RV.milista \leftarrow CrearLista()$

P.Apilar(x):

$P.milista.$Agregar(x)

P.Vacía?() $\rightarrow \mathbb{B}$:

$RV \leftarrow (P.milista.$Longitud$() = 0)$

P.Tope() $\rightarrow ELEM$: (Pre: $\neg P$.Vacía?())

$RV \leftarrow P.milista.$Iésimo$(P.milista.$Longitud$() - 1)$

P.Desapilar(): (Pre: $\neg P$.Vacía?())

$P.milista.$BorrarIésimo$(P.milista.$Longitud$() - 1)$

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM)
- P.Apilar(x)
- P.Vacía?$() \rightarrow \mathbb{B}$
- P.Tope$() \rightarrow ELEM$
- P.Desapilar$()$

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM) $O(1)$
- P.Apilar(x)
- P.Vacía?$()$ $\rightarrow \mathbb{B}$
- P.Tope$()$ $\rightarrow ELEM$
- P.Desapilar$()$

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM) $\quad O(1)$
- P.Apilar(x) $\quad O(n)$
- P.Vacía?$()$ $\rightarrow \mathbb{B}$
- P.Tope$()$ $\rightarrow ELEM$
- P.Desapilar$()$

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM) $O(1)$
- P.Apilar(x) $O(n)$
- P.Vacía?() $\rightarrow \mathbb{B}$ $O(1)$
- P.Tope() $\rightarrow ELEM$
- P.Desapilar()

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM) $O(1)$
- P.Apilar(x) $O(n)$
- P.Vacía?() $\rightarrow \mathbb{B}$ $O(1)$
- P.Tope() $\rightarrow ELEM$ $O(n)$
- P.Desapilar()

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

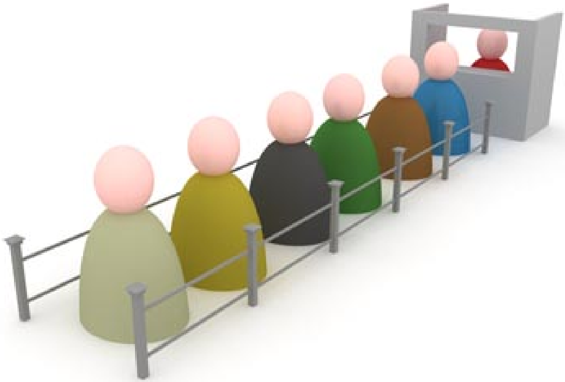
- CrearPila() $\rightarrow$ Pila(ELEM) $\quad O(1)$
- P.Apilar(x) $\quad O(n)$
- P.Vacía?() $\rightarrow \mathbb{B}$ $\quad O(1)$
- P.Tope() $\rightarrow ELEM$ $\quad O(n)$
- P.Desapilar() $\quad O(n)$

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM) $O(1)$
- P.Apilar(x) $O(n)$
- P.Vacía?() $\rightarrow \mathbb{B}$ $O(1)$
- P.Tope() $\rightarrow ELEM$ $O(n)$
- P.Desapilar() $O(n)$

¿Se les ocurre otra estructura de representación que nos permita bajar todos estos a O(1)?



TAD Cola(ELEM)

Operaciones:

- CrearCola() $\rightarrow$ Cola(ELEM): Crea una cola vacía.
- C.Encolar(x): Agrega el elemento x al final de la cola C.
- C.Vacía?() $\rightarrow \mathbb{B}$: Dice si la cola C está vacía.
- C.Primero() $\rightarrow ELEM$: Devuelve el primer elemento de C.
  Precondición: $\neg C$.Vacía?().
- C.Desencolar(): Borra el primer elemento de C.
  Precondición: $\neg C$.Vacía?().

donde C : Cola(ELEM), x : ELEM (entero, char, etc.).

Las colas tienen estrategia FIFO (*first in, first out*).
Implementación: Muy parecida a la de Pilas. Ejercicio.

TAD Conjunto(ELEM). Operaciones.

- CrearConjunto() $\to$ Conjunto(ELEM): Crea un conjunto vacío.
- C.Agregar(x): Agrega el elemento x al conjunto C.
- C.Pertenece?$(x) \to \mathbb{B}$: Dice si el elemento x está en C.
- C.Eliminar(x): Elimina el elemento x de C.
- C.Tamaño() $\to \mathbb{Z}$: Devuelve la cantidad de elementos de C.
- C_1.Igual?$(C_2) \to \mathbb{B}$: Dice si los dos conjuntos son iguales.
- C_1.Unión$(C_2) \to$ Conjunto(ELEM): Devuelve un nuevo conj. $C_1 \cup C_2$.
- C_1.Intersección$(C_2) \to$ Conjunto(ELEM): Devuelve nuevo conj. $C_1 \cap C_2$.
- C_1.Diferencia$(C_2) \to$ Conjunto(ELEM): Devuelve un nuevo conj. $C_1 \backslash C_2$.
- C.ListarElementos() $\to$ Lista(ELEM): Devuelve una lista de los elementos de C, en cualquier orden.
- C.AgregarTodos(L): Agrega todos los elementos de L en C.

donde C, C_1, C_2: Conjunto(ELEM), x: ELEM, L: Lista(ELEM).

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

MOMENTO. ¿No dijimos que las listas no son buenas para representar conjuntos?

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

MOMENTO. ¿No dijimos que las listas no son buenas para representar conjuntos?

Correcto, pero acá *encapsulamos* las dificultades de representar conjuntos con listas, y el usuario del TAD Conjunto no se entera de las mismas.

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

MOMENTO. ¿No dijimos que las listas no son buenas para representar conjuntos?

Correcto, pero acá *encapsulamos* las dificultades de representar conjuntos con listas, y el usuario del TAD Conjunto no se entera de las mismas.

Invariante de representación de esta estructura:

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

MOMENTO. ¿No dijimos que las listas no son buenas para representar conjuntos?

Correcto, pero acá *encapsulamos* las dificultades de representar conjuntos con listas, y el usuario del TAD Conjunto no se entera de las mismas.

Invariante de representación de esta estructura:
La lista ls no puede tener elementos repetidos.

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):

$RV.ls \leftarrow$ CrearLista()

C.Pertenece?$(x) \rightarrow \mathbb{B}$:

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):

$RV.ls \leftarrow$ CrearLista()

C.Pertenece?$(x) \rightarrow \mathbb{B}$:

$RV \leftarrow (C.ls.\text{Cantidad}(x) > 0)$

C.Agregar(x):

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):

$RV.ls \leftarrow$ CrearLista()

C.Pertenece?$(x) \rightarrow \mathbb{B}$:

$RV \leftarrow (C.ls.\text{Cantidad}(x) > 0)$

C.Agregar(x):

if $(\neg C.\text{Pertenece}(x))$ $C.ls.\text{Agregar}(x)$

C.Eliminar(x):

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):

$RV.ls \leftarrow$ CrearLista()

C.Pertenece?$(x) \rightarrow \mathbb{B}$:

$RV \leftarrow (C.ls.\text{Cantidad}(x) > 0)$

C.Agregar(x):

if $(\neg C.\text{Pertenece}(x))$ $C.ls.\text{Agregar}(x)$

C.Eliminar(x):

if $(C.\text{Pertenece}(x))$ $C.ls.\text{BorrarIésimo}(C.ls.\text{Indice}(x))$

C.Tamaño() $\rightarrow \mathbb{Z}$:

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):

$RV.ls \leftarrow$ CrearLista()

C.Pertenece?$(x) \rightarrow \mathbb{B}$:

$RV \leftarrow (C.ls.\text{Cantidad}(x) > 0)$

C.Agregar(x):

if $(\neg C.\text{Pertenece}(x))$ $C.ls.\text{Agregar}(x)$

C.Eliminar(x):

if $(C.\text{Pertenece}(x))$ $C.ls.\text{BorrarIésimo}(C.ls.\text{Indice}(x))$

C.Tamaño$() \rightarrow \mathbb{Z}$:

$RV \leftarrow C.ls.\text{Longitud}()$

TAD Conjunto(ELEM) - Posible implementación

C.ListarElementos() $\rightarrow$ Lista(ELEM):

TAD Conjunto(ELEM) - Posible implementación

C.ListarElementos() $\rightarrow$ Lista(ELEM):

$RV \leftarrow$ CrearLista()
$i \leftarrow 0$
while $(i < C.ls.$Longitud$())$ {
    RV.Agregar($C.ls.$Iésimo(i))
    $i \leftarrow i + 1$
}

C.AgregarTodos(L):

TAD Conjunto(ELEM) - Posible implementación

C.ListarElementos() $\rightarrow$ Lista(ELEM):

$RV \leftarrow$ CrearLista()
$i \leftarrow 0$
while $(i < C.ls.$Longitud$())$ {
    RV.Agregar$(C.ls.$Iésimo$(i))$
    $i \leftarrow i + 1$
}

C.AgregarTodos(L):

$i \leftarrow 0$
while $(i < L.$Longitud$())$ {
    C.Agregar$(L.$Iésimo$(i))$
    $i \leftarrow i + 1$
}

TAD Conjunto(ELEM) - Posible implementación

C_1.Unión(C_2) $\rightarrow$ Conjunto(ELEM):

TAD Conjunto(ELEM) - Posible implementación

C_1.Unión$(C_2) \to$ Conjunto(ELEM):

$RV \leftarrow$ CrearConjunto()
RV.AgregarTodos(C_1.ListarElementos())
RV.AgregarTodos(C_2.ListarElementos())

C_1.Igual?$(C_2) \to \mathbb{B}$:

TAD Conjunto(ELEM) - Posible implementación

C_1.Unión(C_2) $\rightarrow$ Conjunto(ELEM):

```
RV ← CrearConjunto()
RV.AgregarTodos(C1.ListarElementos())
RV.AgregarTodos(C2.ListarElementos())
```

C_1.Igual?(C_2) $\rightarrow$ $\mathbb{B}$:

```
if (C1.Tamaño() = C2.Tamaño()) {
    RV ← TRUE
    L ← C1.ListarElementos()
    i ← 0
    while (i < L.Longitud()) {
        RV ← RV AND C2.Pertenece?(L.Iésimo(i))
        i ← i + 1
    }
} else {
    RV ← FALSE
}
```

Repaso

- Tipos Abstractos de Datos.
  - Parte pública: nombre + especificación de operaciones.
  - Parte privada: estructura, invariante, algoritmos.
- TADs Fecha, Lista(ELEM), Conjunto(ELEM), etc.

Próximos temas

- Técnicas algorítmicas:
  - Backtracking. 8 reinas.
  - (Heurísticas. Algoritmos aproximados.)