En la parte *Nuevos ejercicios* se encuentran nuevos ejercicios creados por motivos didáticos a ser incorporados en la práctica.

Algoritmos y Estructuras de Datos II

Práctica 3 – Diseño: invariante de representación y función de abstracción

Notas preliminares

- Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.
- Para todos los ejercicios de esta parte, salvo indicación contraria, se pide definir interfaz imperativa (pre y post condición y aspectos de aliasing, no la complejidad), estructura e invariante de representación, función de abstracción y los algoritmos que implementen las operaciones.

Ejercicio 1

El siguiente TAD modela una cola que puede contener a lo sumo n elementos.

TAD ColaAcotada

observadores básicos

verCola	: cacotada	$\longrightarrow$	cola(nat)	
capacidad	: cacotada	$\longrightarrow$	nat	

generadores

vacia	: nat	$\longrightarrow$	cacotada	
encolar	: nat $\times$ cacotada c	$\longrightarrow$	cacotada	$\{(\text{tamaño}(\text{verCola}(c)) < \text{capacidad}(c))\}$

axiomas

verCola(vacia(c))	$\equiv$	vacia
verCola(encolar(a, c))	$\equiv$	encolar(a, verCola(c))
capacidad(vacia(n))	$\equiv$	n
capacidad(encolar(a, c))	$\equiv$	capacidad(c)

Fin TAD

Como estructura de representación se propone utilizar un *buffer circular*. Esta estructura almacena los k elementos de la cola en posiciones contiguas de un arreglo, aunque no necesariamente en las primeras k posiciones. Para ello, se utilizan dos índices que indican en qué posición empieza y en qué posición termina la cola. Los nuevos elementos se encolan "a continuación" de los actuales tomando módulo n, es decir, si el último elemento de la cola se encuentra en la última posición del arreglo, el próximo elemento a encolar se ubicará en la primera posición del arreglo. Notar que para desencolar el primer elemento de la cola, simplemente se avanza el índice que indica dónde empieza la cola (eventualmente volviendo éste a la primera posición del arreglo). En nuestra implementación, además de esto, se pone en cero la posición que se acaba de liberar. Se propone la siguiente estructura de representación.

`cacotada` se representa con `estr`

donde `estr` es `tupla`(*inicio*: `nat`, *fin*: `nat`, *elem*: `array[`$0 \ldots n$`] de nat`)

Se pide:

a) Definir el invariante de representación y la función de abstracción.
b) Escribir la interface completa y todos los algoritmos.

Ejercicio 2 ★

Se propone la siguiente estructura para representar a los polinomios que tienen a lo sumo grado n (ver TAD en Práctica 1):

`polinomio` **se representa con** `estr`

donde `estr` es `tupla`(*grado*: `nat`, *coef*: `array[`$0 \dots n$`] de nat`)

Se pide:

a) Definir el invariante de representación y la función de abstracción.
b) Escribir la interface completa y el algoritmo para la función evaluar.

Ejercicio 3

Los palíndromos son aquellas palabras que pueden leerse al derecho o al revés. El siguiente TAD describe a los palíndromos:

TAD PALÍNDROMO(α)

observadores básicos
ver : palindromo(α) $\longrightarrow$ secu(α)

generadores
medio : α $\longrightarrow$ palindromo(α)
medioDoble : α $\longrightarrow$ palindromo(α)
agregar : $\alpha \times$ palindromo(α) $\longrightarrow$ palindromo(α)

axiomas
ver(medio(a)) $\equiv$ a $\bullet$ <>
ver(medioDoble(a)) $\equiv$ a $\bullet$ a $\bullet$ <>
ver(agregar(a,p)) $\equiv$ a $\bullet$ (ver(p) $\circ$ a)

Fin TAD

Se propone la siguiente estructura de representación:

`palindromo` **se representa con** `estr`

donde `estr` es `tupla`(*long*: `nat`, *palabra*: `secu(`α`)`)

dónde *palabra* representa el palíndromo completo.
Se pide:

a) Definir el invariante de representación y la función de abstracción.
b) Escribir la interface y el algoritmo para la función ver.
c) Rehacer los ítems anteriores si el campo *palabra* en lugar de la palabra completa guardamos sólo la mitad inicial de la palabra (redondeando hacia arriba).

Ejercicio 4

Se propone la siguiente estructura para representar el TAD árbol binario:

`ab` **se representa con** `puntero(estr)`

donde `estr` es `tupla`(*altura*: `nat`, *izq*: `puntero(estr)`, *raiz*: α, *der*: `puntero(estr)`)

Se pide:

a) Definir el invariante de representación en lenguaje natural.
b) Definir la función de abstracción usando los observadores en lenguaje natural.
c) Función de abstracción utilizando los generadores.
d) Definir el invariante de representación en lenguaje formal (recordar que puede utilizarse primer órden y funciones auxiliares de TADs)
e) Definir la función de abstracción usando los observadores en lenguaje formal (recordar que puede utilizarse primer órden y funciones auxiliares de TADs)

Ayuda: Tanto el invariante de representación como la función de abstacción pueden definirse recursivamente.

Ejercicio 5 ★

Se decidió utilizar la siguiente estructura para representar una fila en un banco (Ejercicio 10, Práctica 1):

`banco` **se representa con** `estr`

donde `estr` es `tupla`(*entraron*: `conj(persona)`, *fila*: `cola(persona)`, *colados*: `conj(persona)`, *atendidos*: `conjunto(persona)`)

Donde:

- *Entraron* es un conjunto con todas las personas que alguna vez estuvieron en la fila.
- *Colados* son las personas que están actualmente en la fila y se colaron al llegar.
- *Atendidos* son las personas que fueron atendidas en el banco.

Escribir invariante de representación y función de abstracción.

Ejercicio 6 *(Alta fiesta)* **★**

El salón Alta Fiesta se encuentra últimamente en dificultades para coordinar el desarrollo de las reuniones que se realizan en sus facilidades. Hoy en día las fiestas se desarrollan de una manera muy particular. Los invitados llegan en grupos, por lo general numerosos, aunque también a veces de una sola persona. Que un invitado llegue sin un regalo está considerado una falta grave a las buenas costumbres. Tanto es así que a dichos individuos no se les permite el ingreso a las fiestas. Lo que sí se permite es que los invitados hagan regalos en grupo: los invitados que llegan en grupo traen un único regalo de parte de todos. Como es habitual, sólo se permite la entrada a la fiesta a aquellas personas que han sido invitadas.

Al ingresar un grupo de invitados a la fiesta, éste se identifica con un nombre: por ejemplo "Los amigos de la secundaria" o "La familia de la novia". Este nombre se usa por ejemplo para los juegos grupales que van a hacer los animadores durante la fiesta. Igualmente, dado que el comportamiento de las personas en masa no siempre es civilizado, se quiere poder saber de manera eficiente cuál es el nombre del grupo más numeroso para poder seguirle el rastro.

Además, se desea tener un registro de todos los regalos, junto con el grupo de personas que lo hicieron, y se desea conocer en todo momento qué personas se encuentran ya en la fiesta.

Se nos ha encomendado realizar un sistema que permite llevar el control del estado de la fiesta en un momento dado. La cátedra ha realizado la especificación del sistema y ha armado una estructura de representación para el diseño del mismo.

TAD AltaFiesta

observadores básicos

invitadosPendientes : AltaFiesta $\longrightarrow$ conj(Persona)
regalos : AltaFiesta $\longrightarrow$ conj(Regalo)
personasPorRegalo : AltaFiesta a $\times$ Regalo r $\longrightarrow$ conj(Persona) $\{r \in regalos(a)\}$
grupoMasNumeroso : AltaFiesta $\longrightarrow$ Grupo

generadores

iniciarFiesta : conj(Persona) *invitados* $\longrightarrow$ AltaFiesta
lleganInvitados : AltaFiesta a $\times$ conj(Persona) c $\times$ Grupo g $\times$ Regalo r $\longrightarrow$ AltaFiesta
$\{c \subseteq invitadosPendientes(a) \wedge r \notin regalos(a)\}$

axiomas

...

Fin TAD

`altafiesta` **se representa con** `estr`

donde `estr` es `tupla`(*invitados*: `conj(persona)`, *presentes*: `cola(persona)`, *grupoDe*: `dicc(grupo, conj(persona))`, *regaloDeGrupo*: `dicc(grupo, regalo)`, *grupoMasNumeroso*: `grupo`)

grupo, persona y regalo **son** string

Informalmente, esta representación cumple las siguiente propiedades:

- En *invitados* están todos los invitados a la fiesta, incluyendo también a aquellos que ya llegaron.
- En *presentes* están los invitados que ya llegaron a la fiesta.
- En *grupoDe* se encuentra, para cada identificador de grupo i, las personas que al llegar agrupadas se identificaron como i.
- En *regaloDeGrupo* se encuentra qué regalo trajo cada grupo.
- *grupoMasNumeroso* contiene al identificador del grupo de más personas. En caso de empate, contiene al lexicográficamente menor de los empatados (se asume que la función $<_{string}$ está definida).

Se pide:

a) Realizar el invariante de representación del módulo. Escribirlo en lenguaje formal y en castellano. Para que quede claro cuáles enunciados informales hacen referencia a cuáles enunciados formales se recomienda numerar cada punto del invariante para luego poder hacer referencia al mismo.
b) Escribir la función de abstracción. De considerarse necesario, explicarla en castellano.
c) Escribir una versión imperativa de la función llegaGrupo marcando claramente los puntos de su programa en que alguna parte del invariante de representación se rompe, indicando a qué parte se refiere, y también aquellos puntos donde éste se reestablece.

Ejercicio 7 *(Planilla de actividades justificacion)*

Un consultor independiente desea mantener una planilla con las actividades que realiza cada mes en cada uno de los proyectos en los que participa. La planilla que desea mantener se describe con el siguiente TAD.

TAD PLANILLA

observadores básicos

actividades	: planilla	$\longrightarrow$	conjunto(actividad)		
proyectos	: planilla	$\longrightarrow$	conjunto(proyecto)		
proyecto	: actividad a × planilla p	$\longrightarrow$	proyecto		$\{(a \in \text{actividades}(p))\}$
mes	: actividad a × planilla p	$\longrightarrow$	mes		$\{(a \in \text{actividades}(p))\}$
horas	: actividad a × planilla p	$\longrightarrow$	horas		$\{(a \in \text{actividades}(p))\}$

generadores

nueva	:	$\longrightarrow$	planilla	
ag	: actividad a × proyecto p × mes m × horas h × planilla q	$\longrightarrow$	planilla	$\{a \notin \text{actividades}(q)\}$

otras operaciones

totProyxMes	: proyecto p × mes m × planilla q	$\longrightarrow$	planilla	$\{(p \in \text{proyectos}(q))\}$
proysMasHoras	: planilla	$\longrightarrow$	conj(proyecto)	

...

Fin TAD

Se propone la siguiente estructura para representar dicho TAD

`planilla` **se representa con** `estr`

donde `estr` es `tupla`(*detalle*: `dicc(actividad, tupla⟨proy: proyecto, mes: mes, horas: nat⟩)`, *horasPorMes*: `dicc(proyecto, array[mes] de horas)`, *ConMasHoras*: `conj(proyectos)`)

mes es un entero en el rango $1 \dots 12$

Se pide:

a) Escribir formalmente y en castellano el invariante de representación.
b) Escribir la función de abstracción.

Ejercicio 8 *(Oficina estatal)* ★

Considerar el siguiente TAD que modela el comportamiento de una oficina del Estado que procesa trámites. Cada trámite está identificado por un ID y se le asigna una categoría al momento de ingresar. Las categorías de la oficina no son fijas, y pueden agregarse nuevas categorías en cualquier momento. En cualquier momento se puede dar prioridad a una categoría. Todos los trámites pendientes que pertenecen a una categoría prioritaria se consideran prioritarios (Notar que en esta oficina, como buena dependencia estatal, un trámite nunca concluye):

TAD OFICINA

géneros oficina

observadores básicos

categorias	: oficina	$\longrightarrow$	conj(categoria)	
pendientes	: oficina	$\longrightarrow$	secu(id)	
prioritarias	: oficina	$\longrightarrow$	conj(categoria)	
catTram	: id i × oficina o	$\longrightarrow$	categoria	$\{(i \in \text{pendientes}(o))\}$

generadores

nuevo	:	$\longrightarrow$	oficina	
nuevaCat	: categoria c × oficina o	$\longrightarrow$	oficina	$\{(c \notin \text{categorias}(o))\}$
nuevoTram	: id i × categoria c × oficina o	$\longrightarrow$	oficina	$\{(i \notin \text{pendientes}(o) \wedge\ c \in \text{categorias}(o))\}$
priorizar	: categoria c × oficina o	$\longrightarrow$	oficina	$\{(c \in \text{categorias}(o))\}$

otras operaciones

pendPrioritarios	: oficina	$\longrightarrow$	secu(id)	
filtrarPorCategorias	: secu(id) s × conj(categoria) × oficina o	$\longrightarrow$	secu(id)	$\{((\forall i : \text{nat})(0 \leq i < \text{long}(s) \Rightarrow_{\text{L}} s[i] \in \text{pendientes}(o)))\}$

...

Fin TAD

Se decidió utilizar la siguiente estructura como representación:

`oficina` **se representa con** `estr`

donde `estr` es `tupla`(*catPrioritarias*: `conj(categoria)`, *tramites*: `dicc(id, categoria)`, *tramites*×*Cat*: `dicc(categoria, conj(id))`, *pendPrioritarios*: `secu(id)`, *pendientes*: `secu(id)`)

Informalmente, *catPrioritarias* representa el conjunto de todas las categorías a las que se ha dado prioridad, *tramites* asocia a cada trámite su categoría mientras que *tramites*×*Cat* describe todos los trámites asociados a cada categoría. *pendPrioritarios* contiene la secuencia de trámites pendientes que tienen una categoría prioritaria mientras que *pendientes* contiene todos los trámites pendientes (incluso a los prioritarios).

a) Escribir en castellano y formalmente el invariante de representación.
b) Escribir formalmente la función de abstracción.

Ejercicio 9 *(Planta industrial)* ★

Considere la siguiente especificación que modela el funcionamiento de alarmas en una planta industrial. La planta cuenta con un conjunto de alarmas asociadas a distintos sensores. Cada sensor está asociado a varias alarmas y cada alarma puede tener varios sensores asociados. Una alarma está activa cuando la medición de al menos uno de sus sensores asociados supera el valor umbral definido para ese sensor. Considere la siguiente especificación.

TAD PLANTA

observadores básicos

esAlarma	: planta × alarma	⟶	bool	
esSensor	: planta × sensor	⟶	bool	
sensoresAlarma	: planta p × alarma a	⟶	conj(sensor)	{esAlarma(p, a)}
umbral	: planta p × sensor s	⟶	nat	{esSensor(p, s)}
medicion	: planta p × sensor s	⟶	nat	{esSensor(p, s)}

generadores

crear	:	⟶	planta	
agregarAlarma	: planta p × alarma a	⟶	planta	{¬esAlarma(p, a)}
agregarSensor	: planta p × sensor s × nat n × conj(alarma) c	⟶	planta	{¬esSensor(p, s) ∧ (∀a: alarma) ($a \in c \Rightarrow$ esAlarma(p, a)) ∧ $n > 0$}
nuevaMedicion	: planta p × sensor s × nat	⟶	planta	{esSensor(p, s)}

otras operaciones

encendida	: planta p × alarma a	⟶	bool	{esAlarma(p, a)}

...

Fin TAD

Se decidió utilizar la siguiente estructura como representación:

`planta` **se representa con** `estr`

donde `estr` es `tupla`(*alarmas*: `dicc(alarma, conj(sensor))`, *sensores*: `dicc(sensor, tupla`⟨*alarmas*`: conj(alarma),` *umbral*`: nat,` *medición*`: nat`⟩))

donde *alarmas* es el diccionario que asocia a cada alarma el conjunto de los sensores asociados a la misma **que están encendidos**, y *sensores* es el diccionario que asocia a cada sensor el conjunto de alarmas que el mismo enciende, el umbral y el valor de la última medición.

Se pide:

a) Escribir formalmente y en castellano el invariante de representación.
b) Escribir la función de abstracción.

1. Introducción a diseño

Ejercicio 10 ★

Diseñar el TAD COLA(NAT) y el TAD PILA(NAT) utilizando una secuencia como estructura de representación.

Ejercicio 11

Diseñar el TAD CONJUNTO(α) sobre secuencia.

Ejercicio 12 ★

Diseñar el TAD CONJUNTO DE NATURALES EN RANGO utilizando un arreglo. El arreglo debe contener la mayor parte de la información, pero la estructura de representación puede contener la información adicional que sea necesaria.

TAD Conjunto de naturales en rango

géneros conjenrango

observadores básicos

$\bullet \in \bullet$: nat $\times$ conjenrango $\longrightarrow$ bool
lower : conjenrango $\longrightarrow$ nat
upper : conjenrango $\longrightarrow$ nat

generadores

$\emptyset$: nat n $\times$ nat m $\longrightarrow$ conjenrango $\{(n \leq m)\}$
Ag : nat a $\times$ conjenrango c $\longrightarrow$ conjenrango $\{(\text{lower}(c) \leq a \leq \text{upper}(c))\}$

axiomas

$\text{lower}(\emptyset(n, m)) \equiv n$
$\text{lower}(\text{Ag}(a, c)) \equiv \text{lower}(c)$

$\text{upper}(\emptyset(n, m)) \equiv m$
$\text{upper}(\text{Ag}(a, c)) \equiv \text{upper}(c)$

$a \in \emptyset(n, m) \equiv \text{false}$
$a \in \text{Ag}(b, c) \equiv (a =_{\text{nat}} b) \vee (a \in c)$

Fin TAD

Ejercicio 13 ★

Diseñar el TAD Conjunto ajustado de naturales teniendo en cuenta lo siguiente:

> "Los elementos que serán representados por el diseño propuesto pertenecerán *mayoritariamente* al rango acotado que se indica al momento de crear el conjunto ajustado."

Proponga una representación que tenga en cuenta este contexto de uso.
Considere la resolución del ejercicio anterior. ¿Se ajusta a lo pedido? Si es así, intente encontrar otra solución distinta; si no, ¿encuentra alguna forma de utilizarla?

TAD Conjunto ajustado de naturales

géneros conjajust

observadores básicos

$\bullet \in \bullet$: nat $\times$ conjajust $\longrightarrow$ bool
lower : conjajust $\longrightarrow$ nat
upper : conjajust $\longrightarrow$ nat

generadores

$\emptyset$: nat n $\times$ nat m $\longrightarrow$ conjajust $\{(\text{n} \leq \text{m})\}$
Ag : nat a $\times$ conjajust c $\longrightarrow$ conjajust

axiomas

$\text{lower}(\emptyset(n, m)) \equiv n$
$\text{lower}(\text{Ag}(a, c)) \equiv \text{lower}(c)$

$\text{upper}(\emptyset(n, m)) \equiv m$
$\text{upper}(\text{Ag}(a, c)) \equiv \text{upper}(c)$

$a \in \emptyset(n, m) \equiv \text{false}$
$a \in (\text{Ag}(b, c)) \equiv a = b \vee a \in c$

Fin TAD

Ejercicio 14

Tomando como partida el TAD *rosetree*(α) presentado en la práctica de tipos abstractos de datos, proponer:

a) Una estructura de representación.
b) Definir el invariante de representación.
c) Definir la función de abstracción.

Ejercicio 15

Diseñar el TAD Diccionario(String, Nat).

a) Represente los diccionarios sobre secuencias.
b) Ahora adapte el diseño a la siguiente información sobre el contexto: es habitual preguntarle al diccionario por la existencia de una clave para luego, en caso de existir, pedir su significado.

Ejercicio 16

Diseñe el TAD Multiconjunto(α), considerando un contexto de uso en el que los multiconjuntos suelen tener pocos elementos distintos y abundante cantidad de repeticiones de cada uno. Escriba los algoritmos para las siguientes operaciones:

- *Vacío*, que devuelve un multiconjunto sin elementos.
- *Agregar*, que agrega una sola repetición del elemento indicado.
- *Eliminar*, que elimina una sola repetición del elemento indicado.
- *#Repeticiones*, que devuelve la cantidad de repeticiones del elemento indicado que hay en el multiconjunto.

Ejercicio 17

Dado el siguiente TAD:

TAD Secundario

observadores básicos

alumnos	: secundario	$\longrightarrow$	conj(alumno)	
#faltas	: alumno a $\times$ secundario s	$\longrightarrow$	nat	$\{a \in alumnos(s)\}$
notas	: alumno a $\times$ secundario s	$\longrightarrow$	multiconj(nota)	$\{a \in alumnos(s)\}$

generadores

nuevo	: conj(alumno)	$\longrightarrow$	secundario	
nota	: alumno a $\times$ nota n $\times$ secundario s	$\longrightarrow$	secundario	$\{a \in alumnos(s)\}$
falta	: alumno a $\times$ secundario s	$\longrightarrow$	secundario	$\{a \in alumnos(s)\}$

Fin TAD

Se pide:

a) Proponer una estructura de representación.
b) Definir el invariante de representación y la función de abstracción.

Nuevos ejercicios

Esta sección contiene ejercicios a ser incorporados en la guía, no necesariamente al final de la misma.

Ejercicio 18

Sea la siguiente estructura para representar diccionario(α, β):

`dicc`(α, β) **se representa con** `estr`

donde `estr` es `tupla`(*claves*: `secu`(α), *valores*: `secu`(β))

a) Escribir con sus palabras cómo puede pensarse esta estructura para reprensentar diccionarios. Escribir una función de abstracción que explique este mapeo.

b) ¿Qué restricciones son necesarias para que la función anterior tenga sentido? Escribir el invariante de representación para esta estructura.

Ejercicio 19

Los palíndromos son aquellas palabras que pueden leerse al derecho o al revés. El siguiente TAD describe a los palíndromos:

TAD PALÍNDROMO(α)

observadores básicos
ver : palindromo(α) $\longrightarrow$ secu(α)

generadores
medio : α $\longrightarrow$ palindromo(α)
medioDoble : α $\longrightarrow$ palindromo(α)
agregar : $\alpha \times$ palindromo(α) $\longrightarrow$ palindromo(α)

axiomas
ver(medio(a)) $\equiv$ a $\bullet$ <>
ver(medioDoble(a)) $\equiv$ a $\bullet$ a $\bullet$ <>
ver(agregar(a,p)) $\equiv$ a $\bullet$ (ver(p) $\circ$ a)

Fin TAD

Se propone el siguiente módulo de diseño para este TAD donde la variable *palabra* en la estructura de representación representa el palíndromo completo.

Módulo Palindromo(α)

Interfaz

se explica con: PALÍNDROMO(α)

géneros: `palindromo`(α)

Operaciones

MEDIO(**in** a : α) $\rightarrow$ res : `palindromo(`α
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} medio(a)$}
Complejidad: $\Theta(copy(a))$
Descripción: Crea un palíndromo con un solo elemento.
Aliasing: No tiene.

MEDIODOBLE(**in** a : α) $\rightarrow$ res : `palindromo(`α
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} medioDoble(a)$}
Complejidad: $\Theta(copy(a))$
Descripción: Crea un palíndromo con el elemento repetido.
Aliasing: No tiene.

AGREGAR(**in/out** p : `palindromo`(α), **in** a : α)
Pre $\equiv$ {$p_0 =_{\text{obs}} p$}
Post $\equiv$ {$res =_{\text{obs}} agregar(a, p)$}
Complejidad: $\Theta(copy(a))$
Descripción: Agrega el elemento al palíndromo en ambas puntas.
Aliasing: No tiene.

VER(**in** p : `palindromo`(α)) $\rightarrow$ res : `secu`(α)
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{obs} ver(p)$}
Complejidad: $\Theta(1)$
Descripción: Devuelve la secuencia de elementos del palíndromo
Aliasing: El resultado se devuelve como referencia no modificable.

Representación

`palindromo` se representa con `estr`

donde `estr` es `tupla`(*long*: `nat`, *palabra*: `secu`(α))

Rep : estr $\longrightarrow$ bool
Rep(e) $\equiv$ true $\iff$ e.long = long(e.palabra) $\wedge$ e.palabra $=_{obs}$ reverso(e.palabra)

Abs : estr e $\longrightarrow$ palindromo(α) {Rep(e)}
Abs(e) $\equiv$ p / ver(p) $=_{obs}$ e.palabra

Algoritmos

---

iMedio(**in** a : α) $\rightarrow$ res : estr

1: $res \leftarrow \langle 1, AgregarAtras(a, vacio())\rangle$ ▷ $\Theta(copy(\alpha))$

Complejidad: $\Theta(copy(\alpha))$

---

iMedioDoble(**in** a : α) $\rightarrow$ res : estr

1: $res \leftarrow \langle 2, AgregarAtras(a, AgregarAtras(a, vacio()))\rangle$ ▷ $\Theta(copy(\alpha))$

Complejidad: $\Theta(copy(\alpha))$

---

iAgregar(**in** a : α, **in/out** e : $estr$)

1: e.long += 1 ▷ $\Theta(1)$
2: AgergarAtras(a, AgergarAdelante(a, e.palabra)) ▷ $\Theta(copy(\alpha))$

Complejidad: $\Theta(copy(\alpha))$

---

iVer(**in/out** e : $estr$) $\rightarrow$ res: secu(α)

1: $res \leftarrow e.palabra$ ▷ $\Theta(1)$ (Por referencia)

Complejidad: $\Theta(1)$

Considerando que esta representación tiene mucha información redundante (la mitad del palíndromo), se propone revisar la misma de forma que `palabra` solo guarde la mitad inicial del palíndromo (redondeando para arriba). La estructura de representación será la misma, pero utilizada de forma distinta. Para realizar este cambio se pide:

a) Repensar la función de abstracción teniendo en cuenta este nuevo uso de la estructura.
b) Redefinir el invariante de representación.
c) Revisar si los algoritmos siguen funcionando y cambiar los que sean distintos.
d) Modificar la interfaz para reflejar los cambios, incluyendo pre y post condiciones, complejidad y aspectos de aliasing.