Paradigmas de Programación (1/4)

PEP 1

Octubre 2016

Rut: _____ **Profesor:** _____

INDICAR ESTOS DATOS EN CADA HOJA DE RESPUESTA. RESPONDER CADA PREGUNTA EN LAS HOJAS PROVISTAS.

Preguntas de Verdadero (V) y Falso (F) debe ir acompañadas de justificación para los casos V y F. Cada pregunta vale 1 pto.

1. __V__ Al pasar una variable por valor se entrega una copia de la información contenida en dicha variable.

__De esta manera no se puede alterar el valor de la variable original __

2. _F__ El operador binario "*" entrega la dirección de memoria de una variable.

Corresponde a la multiplicacion

3. _V__ El paradigma lógico resulta apropiado para la lógica deductiva.

De lo general a lo particular

4. Explique brevemente en qué consiste a que se refiere la expresión de que "las funciones son ciudadanos de primera clase" bajo el paradigma funcional.

Que funciones y valores se encuentran en un mismo nivel que los valores, por lo que pueden ser tratadas como valores (ej: paso de funciones como parámetros o devolver funciones)

5. Dado el siguiente programa en prolog:

```
factorial(0,1):-!.
factorial(N,OUT):- Nant is N-1, factorial(Nant,OUTant), OUT is OUTant*N.
```

Identifique las variables, señale y explique a qué tipo de variables (bajo el prisma de binding y scope) corresponden.

N y OUT son universalmente cuantificables

Nant y OUTant son existencialmente cuantificables

6. _F__ Un programa lógico se basa en el concepto de algoritmo y lógica.

La noción de algoritmo desaparece y se pasa a un enfoque de tipo descriptivo del problema en si. Los procesos algoritmos son "invisibles" para el programador

7. Explique la diferencia entre una cláusula de Horn y una regla.

____Las reglas son clausulas de Horn

8. Al contrastar el paradigma funcional con el paradigma imperativo ¿Existen ventajas?, Si es así, ¿Cuáles serían y por qué se les puede considerar ventajas?

Debido que no hay variables y por lo mismo no hay cambios de estados, la programación funcional es libre de efectos colaterales

9. ¿Cuál es la diferencia entre el paso de parámetros por valor y por referencia?

Que en el paso por valor se pasa una copia del contenido de la variable (valor), mientras que en el paso por referencia se pasa un acceso directo al espacio de memoria de la variable sobre la cual se desea operar.

10. Cuales son las unidades de programación en los siguientes paradigmas?

a) Lógico: reglas/hechos (cláusulas de horn)
b) Funcional: funciones
c) Imperativo-procedural: procedimientos

11. En C. ¿Cuál es el tipo de dato que retorna la función malloc() o calloc() ?

A. Void *
B. Puntero a memoria alocada
C. Void **
D. Int *

Paradigmas de Programación (2/4)

PEP 1

Octubre 2016

Rut: _____ **Profesor:** _____

INDICAR ESTOS DATOS EN CADA HOJA DE RESPUESTA. RESPONDER CADA PREGUNTA EN LAS HOJAS PROVISTAS.

Se requiere implementar un programa bajo el paradigma imperativo-procedural para operar con números complejos en su representación binómica.

1.- Especificar una representación apropiada para estos números **(1 pts).** Además implementar estructura que refleje su representación **(1 pto)**. Implementar además como parte del TDA número complejo, funciones para realizar las cuatro operaciones básica (ver información al final de la pregunta) **(1,5 pto cada una)**.

2.- Implementar solución en pseudo-C que permita operar sobre dos arreglos de números complejos de entrada a través de las cuatro operaciones básica. El resultado de su programa debe ser un nuevo arreglo de números complejos donde cada uno de los elemento corresponde al resultado de operar los elementos correspondientes en cada una de las listas de entrada **(8 pts)**. Procure identificar claramente a través de comentarios los tipos de paso de parámetro utilizados **(2 pto)**. Esquematizar el diseño de su solución **(2 pts)**

((a +bi) (c + di)) + ((e + fi) (g + hi)) = ([(a+b) + (e + f)i] [(c + g) (d + h)i])

<u>Recordatorio:</u>
Número complejo:

$$a + bi$$

Suma:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Resta:

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

Multiplicación:

$$(a + bi) * (c + di) = (ac + bd) + (ad + bc)i$$

División:

$$\frac{(a + bi)}{(c + di)} = \frac{(ac + bd)}{c^2 + d^2} + \frac{(bc - ad)}{c^2 + d^2} i$$

En pseudo-C

Parte 1

Representación: Par de números donde el primer elemento representa la parte real y el segundo la parte imaginaria.

```
typedef struct  ; para la prueba se puede omitir el typedef
{
        float real;
        float img;
} Complejo

Complejo suma (Complejo c1, Complejo c2)
{
        Complejo c3;
        c3.real = c1.real + c2.real;
        c3.img = c1.img + c2.img;
        return c3;
}

Complejo resta(Complejo c1, Complejo c2)
{
        Complejo c3;
        c3.real = c1.real - c2.real;
        c3.img = c1.img - c2.img;
        return c3;
}

Complejo multiplicacion (Complejo c1, Complejo c2)
{
        Complejo c3;
        c3.real = c1.real * c2.real +c1.img * c2.img;
        c3.img = c1.real*c2.img + c1.img * c2.real;
        return c3;
}

Complejo division(Complejo c1, Complejo c2)
{
        Complejo c3;
        float denominador = sqr(c2.real)+sqr(c2.img);
        c3.real = (c1.real * c2.real +c1.img * c2.img)/denominador;
        c3.img = (c1.img*c2.real + c1.real * c2.img)/denominador;
        return c3;
}
```

Parte 2

Complejo[] sumaArreglos(Complejo[] arreglo1, Complejo[] arreglo2);
Complejo[] restaArreglos(Complejo[] arreglo1, Complejo[] arreglo2);
Complejo[] multiplicacionArreglos(Complejo[] arreglo1, Complejo[] arreglo2);
Complejo[] divisionArreglos(Complejo[] arreglo1, Complejo[] arreglo2);

Paradigmas de Programación (3/4)

PEP 1

Octubre 2016

Rut: _____ **Profesor:** _____

INDICAR ESTOS DATOS EN CADA HOJA DE RESPUESTA. RESPONDER CADA PREGUNTA EN LAS HOJAS PROVISTAS.

Se requiere implementar un programa bajo el paradigma funcional para operar con números complejos en su representación binómica.

1.- Implementar solución en pseudo-scheme para soportar las 4 operaciones básicas sobre números complejos (considere que la aridad de estas funciones es 2) **(1,5 pts cada una)**. Procure especificar y explicar una representación apropiada para estos números **(1 pts).** Además implementar constructor **(1 pto)** y selectores naturales **(2 pts)**.

2.- Implementar una función en pseudo-scheme que reciba dos listas de números complejos y una operación para números complejos. La función debe entregar una lista donde cada uno de los elemento corresponde al resultado de operar los elementos correspondientes en cada una de las listas de entrada **(8 pts)**. Especificar y explicar el tipo de recursión utilizada **(2 pto)**.

Ej:

((a +bi) (c + di)) + ((e + fi) (g + hi)) = ([(a+b) + (e + f)i] [(c + g) (d + h)i])

Recordatorio:
Número complejo:

$$a + bi$$

Suma:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Resta:

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

Multiplicación:

$$(a + bi) * (c + di) = (ac + bd) + (ad + bc)i$$

División:

$$\frac{(a + bi)}{(c + di)} = \frac{(ac + bd)}{c^2 + d^2} + \frac{(bc - ad)}{c^2 + d^2} i$$

Parte 1

; representación
par de números **;1**

```
#lang racket
;constructor ;1
(define (complejo r i)
        (cons r i)
)

;selectores naturales
(define real car) ;1
(define img cdr) ;1

;operaciones
(define (suma c1 c2)    ;1.5
        (complejo (+ (real c1) (real c2)) (+ (img c1) (img c2)))
)

(define (resta c1 c2)  ;1.5
        (complejo (- (real c1) (real c2)) (- (img c1) (img c2)))
)

(define (multiplicacion c1 c2)  ;1.5
        (complejo (+ (* (real c1) (real c2)) (* (img c1) (img c2)))
                  (+ (* (real c1) (img c2) (* (img c1) (real c2))))
        )
)

;se asume sqr para calcular cuadrado o bien se puede multiplicar dos veces el operando
(define (division c1 c2) ;1.5
```

```
  (let ([denominador (+ (sqr (real c2)) (sqr (img c2)))])
        (complejo (/ (+ (* (real c1) (real c2)) (* (img c1) (img c2))) denominador)
                  (/  (- (* (img c1) (real c2)) (* (real c1) (img c2))) denominador)
          )
)
)
```

Parte 2

```
(define (operar L1 L2 op)  ;8
        (define (operarAux L1 L2 op)
                (if (null? L1)
                        null
                        ;se usa recursion lineal ya que se deja estado pendiente 2 pts
                        (cons (op (car L1) (car L2)) (operarAux (cdr L1) (cdr L2) op))
                )
        )
        (if (= (length L1) (length L2))
                (operarAux L1 L2 op)
                null
        )
)
```

Paradigmas de Programación (4/4)

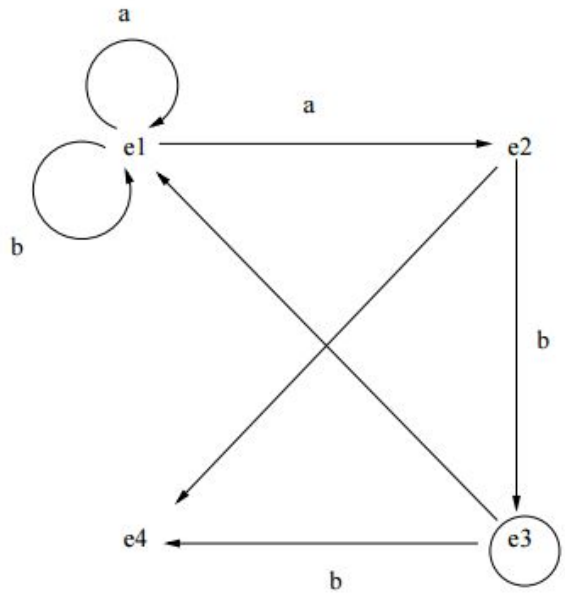
PEP 1

Octubre 2016

Rut: _____ **Profesor:** _____

INDICAR ESTOS DATOS EN CADA HOJA DE RESPUESTA. RESPONDER CADA PREGUNTA EN LAS HOJAS PROVISTAS.

El siguiente esquema representa un autómata no-determinista compuesto por los estados e1, e2, e3 y e4 además de las transiciones entre estos. En este autómata el estado e3 es estado final. Este autómata permite procesar cadenas de texto compuestas por los caracteres a y b. Por ejemplo, partiendo en el estado e1 las cadenas “abab”, “aaaab”, “abaaab” son reconocidas dado que la secuencia de transiciones termina en el estado e3. Por otro lado, si se parte en el estado e1, el string “aaabb” no es reconocido ya que termina en el estado e4. Cabe señalar que este autómata se considera no determinista porque para un estado existen más de una transición para un mismo símbolo (ej: estado e1 con dos transiciones para el símbolo a). En este autómata también existen transiciones nulas, esto es, que no hay símbolo de por medio.



1) Representar el autómata utilizando las siguientes relaciones **(3 pts)**
   - final(X) que se verifica si X es el estado final.
   - trans(E1,X,E2) que se verifica si se puede pasar del estado E1 al estado E2 usando la letra X.
   - nulo(E1,E2) que se verifica si se puede pasar del estado E1 al estado E2 mediante un movimiento nulo

```
final(e3).

trans(e1,a,e1).
trans(e1,a,e2).
trans(e1,b,e1).
trans(e2,b,e3).
trans(e3,b,e4).

nulo(e2,e4).
nulo(e3,e1).
```

2) Definir la relación acepta(E,L) que se verifique si el autómata, a partir del estado E, acepta la lista L de símbolos. Por ejemplo **(5 pts)**:

- acepta(e1,[a,a,a,b])
  true
- acepta(e2,[a,a,a,b])
  false

```
acepta(E,[]) :- final(E).
acepta(E,[X|L]) :- trans(E,X,E1), acepta(E1,L).
acepta(E,L) :- nulo(E,E1), acepta(E1,L).
```

3) Mostrar a través de una traza cómo se evaluaría la consulta acepta(e1,[a,a,a,b]) **(2 pts)**

Miguel, hacer esto por favor