Paradigmas de Programación

PEP 1
13 de Mayo de 2015

Rut: _____
Profesor: _____

I. Paradigma imperativo-procedural

Fundamentos (1 pto. cada una)

1. Indique de forma breve lo que ocurre en cada una de las siguientes líneas (ej: resultado, significado, error en tiempo de compilación, error en tiempo de ejecución, etc.):

const int valor1 = 5; _____
int valor2 = 10; _____
valor1 = 10; _____
int *const puntero = &valor2; _____
*puntero = 15; _____
const int *const puntero2 = &valor1; _____
puntero2 = puntero; _____

const int valor1 = 5; Asignación a la variable valor1 del tipo entero constante con valor 5.
int valor2 = 10; Asignación a la variable valor2 del tipo entero con valor inicial 10.
valor1 = 10; Error de compilación, valor1 es constante
int *const puntero = &valor2; Asignación a la variable puntero del tipo puntero constante a entero con valor memoria de la variable valor2
*puntero = 15; Desreferenciación de variable puntero. valor2 ahora tiene el valor 15.
const int *const puntero2 = &valor1; Asignación a la variable puntero2 del tipo puntero constante a entero constante con valor memoria de la variable valor1.
puntero2 = puntero; Error de compilación, puntero2 es constante

2. ¿Qué relación existe entre programación imperativa y la programación procedural?
_____ (La programación procedural es un tipo de programación imperativa y estructurada)

3. En el contexto de los TDAs provea un ejemplo de representación bajo el paradigma imperativo-procedural y concretamente en el lenguaje C. Su ejemplo debe incluir el elemento a modelar y su respectiva representación. Puede usar el mismo ejemplo empleado en los otros dos paradigmas. _____ Fecha: struct fecha {unsigned int dia; unsigned int mes; int year}

4. ___V__ (V o F con justificación) Asumiendo una máquina de 32bits. El tamaño en espacio de memoria de los punteros siempre es el mismo, independiente del tipo de dato al cual apuntan.

---

5. Considere el problema de encontrar los números dentro de una lista que son menores que el promedio de todos los elementos contenidos en ella. Los números identificados deben ser presentados en orden ascendente. Por ejemplo, dada la lista [1,8,7,6,5,3,2,4,9], el resultado debería ser el siguiente: [1,2,3,4]. Realicé una descomposición descendente para expresar su estrategia de solución. Solo liste los subproblemas que se deben resolver y en el orden en que se deben resolver en no más de una línea. _____

Calcular promedio -> Ordenar Ascendentemente -> Localizar números menores que el promedio
o
Ordenar Ascendentemente -> Calcular promedio -> Localizar números menores que el promedio
o
Calcular promedio -> Localizar números menores que el promedio -> Ordenar Ascendentemente

Desarrollo (15 ptos).

Un número binario tiene la forma B_n.....$B_3B_2B_1B_o$, donde Bi es 0 o 1 (ej: 101). Un número binario se puede convertir a una representación decimal de la siguiente forma: B_n*2^n+....+B_3*2^3+B_2*2^2+B_1*2^1+B_0*2^0 (ej: 1*2^2+0*2^1+1*2^0 = 4+0+1 = 5). Escriba un programa en *pseudo-C* (procure emplear sintaxis lo más cercana a C para describir uso de punteros, tipos de datos, etc. Cualquier notación especial definida por usted debe ser claramente definida) que permita a un usuario ingresar como parámetro de ejecución del programa un número entero positivo (sin signo) de hasta 16 bits en representación binaria (Como un string de caracteres '0' y '1') y entregue como resultado por pantalla su representación decimal. Realice una descomposición adecuada en subproblemas (soluciones implementadas en un solo procedimiento no serán evaluados). Realice validaciones básicas para verificar la entrada. Para su solución sólo se permite la implementación de procedimientos y NO funciones. Solo se permite el uso de libreria <string.h>, <stdlib.h> y <stdio.h>. Cualquier recurso requerido para el resolver el problema que no sea provisto por estas librerías debe ser implementado. Procure que el main de su programa quede limpio y donde solo se hagan llamados a los procedimientos requeridos para resolver el problema. Para convertir un char a int considere el siguiente ejemplo: *char x = 1; int i = x - '0' //luego x es el entero 0*;

```
#include <stdio.h>
#include <string.h>

void potencia(int base, int exp, int *result)
{
   *result = 1;

   for ( int i = 1; i <= exp; i++ )
      *result *= base;
}
```

```
void checkFormat(char *binary, int *valid)
{
   int length = strlen (binary);
   *valid = 1;

   for ( int i = 0; i < length; i++ )
            if ((binary [i] != '0') && (binary [i] != '1'))
            {
              *valid = 0
              break;
            }
}

void convertToDecimal(char *binary, int *decimal)
{
   int length = strlen (binary);
   *decimal = 0;
   for ( int i = 0; i < length; i++ ) {
      int binaryDigit = binary [i] - '0';
      int currentPotencia = 0;
      potencia(2, length - 1 - i, &currentPotencia);
      *decimal += (binaryDigit * currentPotencia);
   }
}

void convertir(char *binary)
{
   int isValid = 0;
   checkFormat(binary, &valid);
   if (valid)
   {
            int decimal = 0;
            convertToDecimal(binary, &decimal);
            printf("Representación decimal es %d",decimal);
   }
   else
            printf("El valor ingresado no es válido\n");

}

int main (int argc, char *argv [])
{
  if (argc==2)
            convertir(argv[1]);
  else
            printf("No se especificaron argumentos");

  return 0;
}
```

II. Paradigma funcional

Fundamentos (1 pto. cada una)

1. Indique el resultado de la siguiente expresión:
   (map (lambda (l) (sort l <)) (list (list 1 4 2 3 9 0) (list 9 8 1 2 3 5))
   _____ '((0 1 2 3 4 9) (1 2 3 5 8 9))
2. ¿Qué elemento resulta determinante en la distinción entre recursión lineal y de cola?

_____ Los estados pendientes o que quedan operaciones por resolver

3. En el contexto de los TDAs provea un ejemplo de representación bajo el paradigma funcional y concretamente en el lenguaje Scheme. Su ejemplo debe incluir el elemento a modelar y su respectiva representación. Puede usar el mismo ejemplo empleado en los otros dos paradigmas.

_____ Fecha : (list dia mes año) con dia mes y año números enteros

4. Dentro de la siguiente expresión, encierre en un círculo los parámetros formales:
(define (hipotenusa x y) (sqrt(+ (* x x) (* y y))))

5. Scheme se relaciona con:
   i) Lenguaje de programación funcional
   ii) Dialecto de LISP
   iii) Soporte para funciones de orden superior

   a) Solo i y ii b) Solo ii y iii c) Solo iii d) Todas las Anteriores e) Ninguna de las anteriores

Desarrollo (15 ptos).

Implementar dos funciones en Scheme que permitan filtrar listas de elementos (de cualquier tipo). La primera función debe hacerlo empleando recursión lineal **(5 pts.)**, mientras que la segunda lo debe hacer por medio de recursión de cola **(5 pts.)**. En ambos casos se debe encapsular el tipo de implementación (descuentos hasta 2 ptos. por no hacerlo). Además en sus implementaciones procure indicar el tipo de recursión y el o los elementos distintivos de cada tipo de recursión **(2 ptos)**. Las funciones deben recibir como parámetros una lista de los elementos a filtrar y un filtro apropiado para operar sobre los elementos contenidos en la lista. El resultado de la función es una lista con los elementos filtrados independientemente del orden. La correspondencia entre el tipo de los elementos de la lista y los filtros proporcionados es responsabilidad del usuario, por lo tanto no realizar validaciones a este nivel. Para la construcción de listas solo puede usar la función *cons*. Finalmente demuestre el uso de sus implementaciones (llamado y resultado) usando como lista de entrada (list 1 2 3 4 5 6) y un criterio que permita seleccionar los elementos mayores que 3. Este filtro debe ser pasado como una función anónima **(3 ptos.)**.

```
;Recursion Lineal
(define (filter1 L f)
    (if (list? L)
        (if (null? L)
            null
            (if (f (car L))
                (cons (car L) (filter1 (cdr L) f))   ;quedan estados pendientes
                (filter1 (cdr L) f)
            )
        )
```

```
        null
    )
  )

;Recursión de Cola
(define (filter2 L f)
   (define (filterAux L f cola)  ;se encapsula el tipo de recursion
      (if (null? L)
          cola  ;el resultado queda en la cola
          (if (f (car L))
              (filterAux (cdr L) f (cons (car L) cola)) ;no hay estados pendientes. Ojo que
      ;invierte la lista. Realice esta implementacion pues creo que la mayoria de los
      ;alumnos que resuelvan el problema haran esto, sin embargo el problema de esta
      ;aproximacion es que invierte la lista. Ahora, el construirla sin inversión no es
      trivial, ;pues requiere aplanar la lista una vez generada.
              (filterAux (cdr L) f cola)
          )
      )
   )
   (if (list? L)
       (filterAux L f null)
       null
   )
)

(filter1 (list 1 2 3 4 5 6) (lambda (x) (> x 3)))
    '(4 5 6)
(filter2 (list 1 2 3 4 5 6) (lambda (x) (> x 3)))
    '(6 5 4)
```

III. Paradigma lógico

Fundamentos (1 pto. cada una)

1. Para cada una de las variables en las cláusulas, indique si estas son universalmente o existencialmente cuantificadas y por qué.

antepasado(X,Y) :- padre(X,Y). _____ X e Y son variables universalmente cuantificadas, puesto que se encuentran definidas en la firma de la cláusula.

antepasado(X,Y) :- antepasado(X,Z),antepasado(Z,Y). _____ X e Y son variables universalmente cuantificadas, puesto que se encuentran definidas en la firma de la cláusula. Z es una variable existencialmente cuantificada, puesto que sólo es instanciada en el cuerpo de la cláusula

2. Explique el concepto del Supuesto del Mundo Cerrado (1 línea).

_____
__ Prolog asume que toda la información relevante del problema se encuentra contenida en las cláusulas definidas por el programador, luego si una consulta no puede ser derivada desde las cláusulas quiere decir que el resultado es falso o que es desconocido.

3. En el contexto de los TDAs provea un ejemplo de representación bajo el paradigma lógico y concreto en el lenguaje Prolog. Su ejemplo debe incluir el elemento a modelar y su respectiva representación. Puede usar el mismo ejemplo empleado en los otros dos paradigmas.
   _____ Fecha: hechos del estilo fecha(dia,mes,agno) o listas del estilo [DIA,[MES,[AGNO,]]]
4. Indique que aseveraciones son correctas con respecto a la siguiente consulta de Prolog:

   crearLaberinto(N,M,"DificultadMedia",cantEnemigos).

   i. El primer parámetro es una variable
   ii. El tercer parámetro es una variable
   iii. Es una cláusula de de aridad 4
   iv. El cuarto parámetro es una variable existencialmente cuantificada
   v. La consulta tiene problemas sintácticos

   a. ii, iii y v
   b. i, iii y iv
   c. ii y iii
   d. ii, iii y iv
   e. Todas son correctas
5. F__ (V o F) Considerando los hechos:
   planeta_grande(urano).
   planeta_grande(jupiter).
   planeta(saturno).

   Es válido definir una cláusula para consultar a Prolog si un planeta No es un planeta grande. (Justificar en caso de ser verdadero o falso).

   No es válido debido al supuesto del mundo cerrado. También es correcto: No es válido, debido a que las cláusulas de Horn solo pueden representar respuesta si/fracaso en lugar de si/no.

Desarrollo (15 ptos).

Implementar un programa en Prolog para calcular el producto punto entre dos vectores (10 ptos.).

Ejemplo: V1=(1,2,3); V2=(4,5,6); V1 • V2 = 1*4+2*5+6*3 = 32.

Realice una traza completa para ilustrar el comportamiento de su programa al operar sobre los vectores V1 y V2 (5 ptos.)

Programa:

```
producto_punto([],[],0).
producto_punto([X|X1],[Y|Y2],L) :- PROD is X * Y, producto_punto(X1,Y2,L1), L is L1+PROD.
```

Traza:

producto_punto([1,2,3], [4,5,6], OUT).
-> X = 1, X1=[2,3], Y=4, Y2=[5,6]. PROD is 4, producto_punto([2,3],[5,6], L1)
->X = 2, X1=[3], Y=5, Y2=[6]. PROD is 10, producto_punto([3],[6], L1)
->X=3, X1=[], Y=6, Y2=[]. PROD is 18, producto_punto([],[],L1), L1=0, OUT is 0+18=18
->X = 2, X1=[3], Y=5, Y2=[6]. PROD is 10, producto_punto([3],[6], L1) => L1 = 18, OUT= 18+10=28.
-> X = 1, X1=[2,3], Y=4, Y2=[5,6]. PROD is 4, producto_punto([2,3],[5,6], L1) => L1=28, OUT = 4+28 = 32.

=> producto_punto([1,2,3], [4,5,6], OUT).
OUT = 32.

Llamado	**X**	**X1**	**Y**	**Y1**	**PROD**	**L1**	**L**	**OUT**
pp([1,2,3], [4,5,6], OUT)	1	[2,3]	4	[5,6]	1*4=4		L1+4	
pp([2,3], [5,6], L1)	2	[3]	5	[6]	2*5=10		L1+10	
pp([3], [6], L1)	3	[]	6	[]	3*6=18		L1+18	
pp([], [], L1)						0		
pp([3], [6], L1)							0+18=18	
pp([2,3], [5,6], L1)							18+10=28	
pp([1,2,3], [4,5,6], OUT)							28+4=32	32

Buena suerte!