

Paradigmas de Programación

Control 1 (PAUTA)
9 de diciembre de 2015

Rut: _____

Profesor: _____

Parte conceptual (2 pts cada una)

Paradigma imperativo

1. ¿Cómo se estructuran las soluciones en el paradigma imperativo procedural? **bloques de código, subrutinas, procedimientos, funciones**
2. ¿De qué forma(s) podría lograr el comportamiento en C de que se puedan retornar múltiples valores de retorno en un procedimiento/función?
 - i. A través de una estructura con múltiples atributos
 - ii. A través de paso de parámetros por valor
 - iii. A través de paso de parámetros por referencia
 - iv. A través de una enumeración de valores.
 - a. Sólo i
 - b. Sólo iv
 - c. i y ii
 - d. i y iii**
 - e. iii y iv
3. Indique qué comportamiento tiene el operador * en C al utilizarlo como un operador binario (Responder en máximo 1 línea).
Multiplifica los dos operandos
4. Indique las afirmaciones correctas con respecto a los valores compuestos en C
 - i. Están formados únicamente por valores primitivos.
 - ii. Se crean obligatoriamente utilizando punteros.
 - iii. Pueden ser arreglos o estructuras.
 - iv. Los datos se almacenan de forma contigua para un mismo dato compuesto.
 - v. Algunos pueden ser recursivos.
 - a. i y iii
 - b. i, ii, iii y iv

- c. iii, iv y v
- d. i, iii, iv y v.
- e. Todos son correctos

Paradigma funcional

- 5. Si se quiere utilizar el paradigma funcional utilizando el lenguaje C, ¿Qué elemento de los indicados en las alternativas no utilizaría de este lenguaje?
 - a. Funciones
 - b. Uso de variables globales
 - c. Paso de parámetros por valor
 - d. Recursión
 - e. Esto no es posible, C sólo permite el paradigma imperativo
- 6. ¿Qué elementos idealmente debería tener un tipo de dato abstracto (TDA) para efectos de su implementación?. Indicarlos considerando su precedencia, de la más básica elemental a las más superficial.

Representaciones, Constructores, Funciones de Pertenencia, Selectores, Modificadores, Otras funciones

- 7. ¿Qué ventaja principal posee el paradigma funcional frente al paradigma imperativo?
 - a. No hay efectos colaterales
 - b. Un mayor rendimiento al simplificar los lenguajes utilizando sólo funciones.
 - c. La existencia de un binding que permite identificadores más variados que en el paradigma imperativo.
 - d. Facilidad con el control de flujo en bucles anidados.
 - e. El uso de paso de parámetros por referencia, que permite un mayor control de la memoria.

Paradigma lógico

- 8. Indique cómo leería la siguiente cláusula de prolog utilizando cuantificadores universales y existenciales

aveCarrognera(X) :- ave(X), come(X, OtroAnimal), muerto(OtroAnimal).

Para todo "X", "aveCarrognera(X)" es verdadero si existe "OtroAnimal" tal que "ave(X)" es verdad y "come(X, OtroAnimal)" es verdad y "muerto(OtroAnimal)" es verdad.

9. Indique que aseveraciones son correctas con respecto a la siguiente consulta de Prolog:
esArtefactoElectrico(Articulo, amper, Volts, 'ClaseAhorroEnergetico').

- i. El primer parámetro es una variable
- ii. El segundo parámetro es una variable
- iii. Es una cláusula de de aridad 4
- iv. El tercer parámetro es una variable universalmente cuantificada.
- v. El cuarto parámetro es una variable

a. i y iii

b. i, iii y iv

c. i, ii y iii

d. ii, iv y v

e. Todas son correctas

10. F (V o F) Los hechos son cláusulas de Horn que son siempre verdad y están compuestos por variables (Justificar en caso de ser verdadero o falso).

No están compuestos por variables, sino que por términos, los cuales son números o átomos (Basta decir que están compuestos por términos)

Parte programación (40 pts)

Los palíndromos son palabras o expresiones que se leen igual de izquierda a derecha y de derecha a izquierda (ej: “ala”, “ana”, “radar”, “anita lava la tina”).

1. (A) Implementar un procedimiento en pseudo-C que reciba un arreglo de punteros a caracteres P, el tamaño de P, y un arreglo de enteros R. El objetivo del procedimiento es evaluar si los elementos de P son palíndromos y el resultado de cada evaluación (valor booleano) se debe almacenar en la posición correspondiente del arreglo R (**8 pts**). (B) Para la implementación debe asignar memoria al arreglo R (**3 pts**). (C) Además debe expresar en un diagrama simple la descomposición del problema (**3 pts**). (D) Finalmente implementar un main donde se ilustre el uso del procedimiento implementado en (A) y que imprima por pantalla los resultados (**2 pts**) . **Nota:** Si lo requiere, puede hacer uso de las funciones *trim* (la cual recibe un string y retorna un string sin espacios) y *strlen* (la cual retorna el largo de un string).

```

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

void evaluarPalindromos(char **P, int size, int **R);
int esPalindromo(char *expr);

int main() { //2 pts
    const int size = 4;
    char* P[size] = { "ola", "ana", "radar", "iva" };
    int *R;
    evaluarPalindromos(P, size, &R); // 1 pto
    int i;
    for (i = 0; i < size; i++)
        printf("%s: %d\n", P[i], R[i]);
    getchar();
    return 0;
}

void evaluarPalindromos(char **P, int size, int **R) // 8 pts
{
    int *arrayAux = (int *)malloc(size*sizeof(int)); // 2 pts
    *R = arrayAux;
    int i;
    for (i = 0; i < size; i++)
        if (esPalindromo(trim(P[i]))) //Ojo que trim no es parte de string.h, solo se usa como parte del supuesto
            arrayAux[i] = 1;
        else
            arrayAux[i] = 0;
}

int esPalindromo(char *expr)
{
    int i, j;
    for (i = 0, j = (strlen(expr)-1); i != j && (i+1) != j; i++, j--)
        if (expr[i] != expr[j])
            return 0;

    return 1;
}

Descomposición: //3 pts

    evaluarPalindromos----->esPalindromo

```

2. (A) Implementar una función en pseudo-Scheme que reciba una lista de strings LP y retorne una lista de valores booleanos. El objetivo del procedimiento es evaluar si los

elementos de LP son palíndromos y el resultado de cada evaluación (valor booleano) se debe almacenar en la posición correspondiente de la lista resultante (**18 pts**). (B) Debe expresar en un diagrama simple la descomposición del problema (**2 pts**). (C) Indicar el o los tipo de recursión empleadas junto a cada función que haga uso de este recurso de programación (**3 pts**). (D) Finalmente, mostrar cómo se utiliza la función implementada en (A) a través de UN ejemplo e indicar cual sería el resultado para la entrada empleada (**2 pts**) . **Nota:** Si lo requiere puede hacer uso de las siguientes funciones:

- remainder: resto de la división entera
- quotient: división entera
- string->list: convierte un string a lista de caracteres
- eqv?: determina si dos elementos son equivalentes
- length: entrega el largo de una lista
- car, cdr, cons, null?, list?

Cualquier otra función requerida debe ser implementada

```
#lang racket

(define (invertirLista L)
  (define (invertirListaAux L Lout)
    (if (null? L)
        Lout
        (invertirListaAux (cdr L) (cons (car L) Lout))) //Recursión de cola
  )
  )

  (if (null? L)
      null
      (invertirListaAux (cdr L) (cons (car L) null)))
  )
)

(define (palindromo? p)
  (define (palindromoAux? p1 p2 l)
    (if (= l 0)
        #t
        (if (not (eqv? (car p1) (car p2)))
            #f
            (palindromoAux? (cdr p1) (cdr p2) (- l 1))) //Recursión de cola
    )
  )
  )

  (palindromoAux? p (invertirLista p) (quotient (length p) 2))
)
)
```

```

(define (evaluarPalindromos LP)
  (if (null? LP)
      null
      (if (palindromo? (string->list (car LP)))
          (cons #t (evaluarPalindromos (cdr LP))) //Recursión Lineal
          (cons #f (evaluarPalindromos (cdr LP))) //Recursión Lineal
      )
  )
)

```

;Ejemplo de uso

```
(evaluarPalindromos (list "ala" "ola" "ana" "baab"))
```

;Resultado

```
'(#t #f #t #t)
```

Descomposición:

evaluarPalindromos

| _____ palindromo?

| _____ invertirLista -----> invertirListaAux

| _____ palindromoAux?

3. Implementar un predicado en prolog para determinar el coeficiente binomial $(n \ p)$ a partir del triángulo de pascal. **Nota:** NO puede hacer uso de nada relativo a factorial. Asuma que la regla se usará en los rangos válidos para n y p (**5 pts**)

	p=0	p=1	p=2	p=3	p=4	p=5	p=6	p=7	p=8	p=9
n=0	1									
n=1	1	1								
n=2	1	2	1							
n=3	1	3	3	1						
n=4	1	4	6	4	1					
n=5	1	5	10	10	5	1				
n=6	1	6	15	20	15	6	1			
n=7	1	7	21	35	35	21	7	1		
n=8	1	8	28	56	70	56	28	8	1	
n=9	1	9	36	84	126	84	36	9	1	

binomial(0,_,1).

binomial(_,0,1).

binomial(N,N,1).

binomial(N,P,OUT):- N1 is N-1, P1 is P-1, binomial(N1,P1,OUT1),binomial(N1,P,OUT2),OUT is OUT1+OUT2.