Paradigmas de Programación

Control 1 (PAUTA)
6 de Mayo de 2015

Rut: _____
Profesor: _____

Parte conceptual (2,5 pts cada una)

Paradigma imperativo

1. __V__ (V o F) El concepto de binding time además de lo que indica su definición está relacionado con el rendimiento. (Justifique tanto si es verdadero como falso).

Es así porque si el binding time se realiza en etapas más tempranas (compilación por ejemplo) se logra un mayor rendimiento que si se realiza en tiempo de ejecución. Esto porque se realizan ciertas operaciones por parte del compilador en lugar que cuando el programa se esté ejecutando. (Cualquiera de ambas explicaciones es suficiente).

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. Retornando una estructura con múltiples atributos
    ii. Realizando paso de parámetros por valor
    iii. Realizando paso de parámetros por referencia
    iv. Retornando 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).
    Multiplica los dos operandos_____
4. Indique las afirmaciones correctas con respecto a los valores compuestos en C
    i. Están formados únicamente por valores primitivos.
    ii. Su tamaño en memoria es igual a la suma del tamaño de sus componentes más 4 u 8 (dependiendo de la arquitectura de la máquina).
    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. Funciones como ciudadanos de primer orden
    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 (35 pts)

1. Los números feos son números cuyos factores primos sólo son 2, 3 o 5. La secuencia 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, . . . muestra los 11 primeros números feos. Por convención, el 1 se incluye. Notar que el 14 no es feo, porque uno de sus factores es 7. Realice un procedimiento en pseudo C que reciba un arreglo de enteros y entregue los 1500 primeros números feos (**5 pts**), haga que el arreglo se le asigne memoria dentro del procedimiento (**3 pts**), haga un main que llame al procedimiento construido y muestre por pantalla estos números (**2 pts**).

```
int main() {
  int n = 1500; //cantidad de números feos que se quieren imprimir
  int *array;
  entregaFeos(&array, n);
  int i;
  for (i = 0; i < n; i++) printf(“%d: %d\n”, i, array[i]);
  return 0;
```

```
}

void entregaFeos(int **array, int n) {
   int *arrayAux = (int *)malloc(n*sizeof(int));
   *array = arrayAux;
   int contador = 1; //para contar la cantidad de números feos que se tienen hasta el
      momento
   int posibleFeo = 1, numeroFeo, anterior;

   while(contador < n+1){
      numeroFeo = posibleFeo;

      while(1) {
            anterior = numeroFeo;
            if(numeroFeo % 5 == 0) numeroFeo /= 5;
            if(numeroFeo % 3 == 0) numeroFeo /= 3;
            if(numeroFeo % 2 == 0) numeroFeo /= 2;

            if(numeroFeo == 1){
                  arrayAux[contador] = posibleFeo;
                  contador++;
                  break;
            }

            if(numeroFeo == anterior) break; //No se pudo seguir dividiendo
      }
      posibleFeo++;
   }
}
```

2. Se desea implementar el TDA hora. Implementar al menos una función por cada nivel del TDA (a excepción de los selectores que den ser implementados en su totalidad) e indicar la representación escogida. Como función complementaria del TDA puede implementar una función que use estratégicamente los elementos subyacentes del TDA (ej: siguienteHora, siguienteMinuto, siguienteSegundo). Procurar hacer validaciones correspondientes (**15 pts**)

Representación: Lista de tres elementos para formato 24 horas, 4 para formato am pm) **(2 pts)**

```
;Constructor  (5 pts)
(define (hora h m s)
        (if (and (integer? h) (integer? m) (integer? s)
                (>= h 0) (<= h 23) (>= m 0) (<= m 59) (>= s 0) (<= s 59))

                (list h m s)
                null
        )
)

;Función de pertenencia
(define (hora? h)
        (and (list? h) (= (length h) 3) (not (null? (hora (car h) (cadr h) (caddr h)))))
 )

 ;selectores  (3 pts , 1pto c/u)
 (define (getHora h)
        (if (hora? h)
            (car h)
            -1
        )
 )

 (define (getMinuto h)
        (if (hora? h)
            (cadr h)
            -1
        )
 )

 (define (getSegundo h)
        (if (hora? h)
            (caddr h)
            -1
        )
```

```
        )

        ;1 modificador   (1 pto)
        (define (setHora h newHora)
                (if (hora? h)
                    (hora newHora (getMinuto h) (getSegundo h))
                    null
                )
        )

        ;1 funcion extra   (4 pts)
        (define (siguienteHora h)
                (if (hora? h)
                    (if (= (getHora h) 23)
                        (setHora h 0)
                        (setHora h (+ 1 (getHora h))
                    )
                    null
                )
        )
```

3. Implementar una regla en prolog para calcular fibonacci de n. **(10 pts)**
   Considere lo siguiente:
   fibonacci(n) = n ; n=0 o n=1
   fibonacci(n) = fibonacci(n-1)+fibonacci(n-2) ; para otro n

```
fib(0,0).
fib(1,1).
fib(N,OUT) :- N1 is N - 1, N2 is N - 2, fib(N1,OUT1), fib(N2,OUT2), OUT is OUT1+OUT2.
```