Paradigmas de Programación

Control 1

Mayo 2016

Rut: _____ **Profesor:** _____

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

Parte conceptual (10 pts - 1 pto c/u)

10 preguntas (tipo V/F o respuesta corta)

1) Paso por valor y paso por referencia

Dado el siguiente programa:

```
#include <stdio.h>
#include <stdlib.h>
int suma(int *a, int *b){
        *a=*a+*b;
        return *a;
}
int main(){
        int a,b;
        a=2;
        b=3;
        suma(&a,&b);
        printf("%i \n",a);
}
```

responda qué valor imprime al ejecutar.

2) Heap vs Stack

Dado el programa de la pregunta anterior, se está utilizando una declaración de variable mediante Heap o Stack? justifique.

Stack. Las variables son de tipo automática/locales. Solo se les reserva memoria en el contexto donde de activación de los procedimientos o funciones. No hay asignación mediante malloc.

---

3) Funciones Lambda

Respecto a funciones lambda:

I. Son funciones anónimas II. Requieren de un define III. Reciben solo un argumento

a) Solo I b) Solo II c) Solo III d) I, II y III

4) Evaluación perezosa

Respecto a la evaluación perezosa se puede decir:

I. La expresión es una promesa de evaluación II.Se puede forzar su evaluación

III. todas las evaluaciones que se realicen quedan pendientes

a) Solo I b) I y II c) I, II y III d) Ninguna

5) Currying

_____ En el currificado las funciones pueden evaluarse parcialmente aunque no se pasen todos los argumentos, quedando a la espera de una evaluación completa cuando reciba el resto de argumentos. Verdadero o falso?

---

6) Verdadero (V) o Falso (F) Justifique

_____ En prolog, para la sentencia padre(X,Y), al realizar una consulta se realiza la combinatoria completa de valores posibles para X e Y. Verdadero o falso?

---

7) Unificación

Dadas las siguientes sentencias:
date(15,mes(mayo),2001)

y las siguientes consultas:

date(D, mes(M), A)

Señale la unificación es:

la unificación es:
D = 15
M = mayo
A = 2001

8) Predicados

Los predicados en prolog se utilizan para referirse a:

a) Objetos
b) Átomos
c) Relaciones
d) Dominios

9) Verdadero (V) o Falso (F) Justifique

____ El orden de las cláusulas no afecta la ejecución de una consulta, solo afecta su eficiencia. Verdadero o falso?

Las relaciones en Prolog son evaluadas en el orden de arriba hacia abajo y a su vez las condiciones de cada cláusula son evaluadas de izquierda a derecha.

10) Objetivos

Sea G un objetivo a satisfacer:

I. Puede haber más de una alternativa que lo satisfaga.
II. En caso de haber un corte !, el objetivo ya se da por satisfecho.
III. G se evalúa con backtraking.

a) Solo I b) Solo II c) Solo III d) I, II y III

Parte Desarrollo (50 pts)

Introducción al problema. Describir TicTacToe

Paradigma Imperativo (17 pts)

a) Establecer una representación para el tablero de TicTacToe (2 pts)
b) ¿Cómo resolvería el problema de verificar si un tablero está completado (verificando todos los casos que dan como resultado un ganador)? Esquematice su solución (3 pts).
c) Implementar una función que permita revisar un tablero se ha completado. La función debe indicar el ganador, si hay un empate, o si aún quedan jugadas por realizar. (8 pts)
d) Implementar un procedimiento que permita retornar a través de sus parámetros el estado de un tablero (1:completo vs 0: incompleto) y el resultado (el ganador, si se trata de un empate, o si aún no se puede determinar). (3 pts) Muestre cómo llamaría este procedimiento en el contexto de la función main (1 pts).

Consideraciones:

- Tablero de 3x3
- Ganador si logra disponer 3 piezas de forma contigua (horizontal, vertical o diagonal)
- Su implementación debe ser en pseudo-C
- Cuenta (si lo desea) con una función o variable que contabiliza la cantidad de jugadas que se han realizado sobre el tablero.

a) representación: char[3][3] board;
b) La verificación del tablero se descompone en verificación de filas, columnas y diagonales

```
#define NADA '\0'
#define EMPATE 'E'
char checkBoard()
{
        char cR = checkRows(); if (cR != NADA) return cR;
        char cC = checkCols(); if (cC != NADA) return cC;
        char cD = checkDiagonals(); if (cD != NADA) return cD;
        int moves = getMoves();
        if (moves == 9) return EMPATE;
        return NADA;
```

```
}
int getMoves()  //alternativamente podrían suponer que se cuenta con un contador de jugadas
{
        int moves = 0;
        int i, j;
        for (i = 0; i < 3; i++)
                for (j = 0; j<3; j++)
                        if ((board[i][j] == 'X') || (board[i][j] == 'O'))
                                moves++;
        return moves;
}
char checkRows()
{
        int i;
        for (i = 0; i < 3; i++)
                if ((board[i][0] != NADA) && (board[i][0] == board[i][1]) && (board[i][0] == board[i][2]))
                        return board[i][0];
        return NADA;
}
char checkCols()
{
        int j=0;
        for (j = 0; j < 3; j++)
                if ((board[0][j] != NADA) && (board[0][j] == board[1][j]) && (board[0][j] == board[2][j]))
                        return board[0][j];
        return NADA;
}
char checkDiagonals()
{
        if ((board[0][0] != NADA) && (board[0][0] == board[1][1]) && (board[0][0] == board[2][2]))
                return board[0][0];

        if ((board[0][2] != NADA) && (board[0][2] == board[1][1]) && (board[0][2] == board[2][0]))
                return board[0][2];
        return NADA;
}
d) void checkBoardProc(int *state, char *result)
  {
        char cB = checkBoard();
        if (cB != NADA)
                *state = 1; //Hay ganador
        else
                *state = 0; //No hay ganador
        *result = cB;
  }
int main()
{
        int state; char result;
        checkBoardProc(&state,&result);
}
```

Paradigma Funcional (23 pts)

a) Especificar una representación para el tablero (3 pts)
b) ¿Cómo resolvería el problema de verificar si un tablero está completado? Esquematice su solución (3 pts).
c) Implementar una función que permita verificar si hay un ganador en el tablero (true o false) solo en base a jugadas horizontales y verticales. No debe indicar quién ha ganado. (14 pts)
d) Usar recursividad en su implementación. Indicar el tipo de recursividad empleada e indicar por qué es el tipo de recursividad señalada. (3 pts)

Consideraciones:

- Su implementación debe ser en pseudo-Scheme
- La dimensión de los tableros a evaluar es de NxN
- Suponga que cuenta con la función (getMaxDim B) que retorna la dimensión N del tablero B

```
Representación 1: '( '(#\_ #\_ #\_) '(#\_ #\_ #\_) '(#\_ #\_ #\_))  ; lista con 3 sublistas
Representación 2: '( '(0 0 #\_) '(0 1 #\_) '(0 2 #\_) '(1 0 #\_) '(1 1 #\_) '(1 2 #\_) '(2 0 #\_) '(2 1 #\_) '(2 2 #\_) )  ; lista de celdas

;Usando representación 2.
;se asume que largo mínimo es mayor que 2
(define (checkList R)
        (if (= (length R) 2)
                (eq? (car R) (cadr R))
                (and (eq? (car R) (cadr R)) (checkList (cdr R)))))

;función de orden superior que permite reconstruir filas o columnas dependiendo de la función f que se pasa como parametro
(define (getList B p f)
        (if (null? B)
                null
                (if (= (f B) p) ;aquí se aplica la función f para extraer el dato que señala fila o columna
                        (cons (caddar (car B)) (getList (cdr B) p)
                        (getList (cdr B) p))))

;función de orden superior que verifica fila o columna a partir de funciones entregadas
(define (checkBoardAux B f)
        (define (checkHelper B p n)
                (if (= p n)
                        #f
                        (or (checkList (getList B p f) (checkHelper B (+ p 1) n)))))
        (checkHelper B 0 (- (getMaxDim B) 1))  ;getMaxDim entrega dimension tablero
)

(define (checkBoard B)
        (or (checkBoardAux B caar) (checkBoardAux B cadar))
```

Paradigma Lógico (10 pts)

Dado un tablero de TicTacToe de 3x3 indicar si una jugada J sobre el tablero es una jugada ganadora.

a) Especificar una representación (2 pts)
b) Implementar cláusulas necesarias para satisfacer el requerimiento (7 pts).
c) Demostrar el uso de su programa mediante una consulta que dé como resultado "true" y otra que dé como resultado "falso" (1 pto).

Solución:

a) La representación del tablero se especifica mediante una lista que contiene nueve variables que pueden adoptar el valor (Valor X ó O).

Se define el predicado "jugada_ganadora", como:

jugada_ganadora(jugador1,Valor, [X1,X2,X3,X4,X5,X6,X7,X8,X9]): La variable Valor es una jugada ganadora del jugador1.

Una jugada_ganadora es tal si cumple con ser una posición_ganadora. Para tal efecto se define la regla :

b) Implementación mediante claúsulas

```
jugada_ganadora(jugador1,Valor) :-
        posición_ganadora(Valor, [X1,X2,X3,X4,X5,X6,X7,X8,X9]).
```

Finalmente se define una posición_ganadora como:

```
Posicion_ganadora( P, [X1,X2,X3,X4,X5,X6,X7,X8,X9]) :-
 igual(X1,X2,X3,P);   % primera fila
 igual(X4,X5,X6,P);   % segunda fila
 igual(X7,X8,X9,P);   % tercera fila
 igual(X1,X4,X7,P);   % primera columna
 igual(X2,X5,X8,P);   % segunda columna
 igual(X3,X6,X9,P);   % tercera columna
 igual(X1,X2,X3,P);   % primera diagonal
 igual(X1,X2,X3,P);   % segunda diagonal

 igual(X,X,X,X).
```

c) Consulta para una por jugada_ganadora.

```
jugada_ganadora(jugador1,X,[X,_,O,_,X,_,O,_,X]).
```

Esta consulta es verdadera porque al coincidir el predicado "jugada_ganadora" se sustituye la variable Valor por X, y luego se intenta probar si es verdadero el predicado "posición_ganadora" con los valores de argumentos

que recibe. Este último predicado es verdadero si se logra probar que al menos uno de los predicados “igual” es verdadero. Considerando que se ha declarado el hecho que cualquier combinación de filas, columnas o diagonal que contengan solo Xs, hace verdadero al predicado “igual”, luego esto hace verdadero al predicado “posición_ganadora” y este a su vez hace verdadero al predicado jugada_ganadora.

Caso en que jugada_ganadora es falsa.

jugada_ganadora(jugador1,X,[X,O,O,X,O,O,X,X])

Esta regla es falsa porque no existe un hecho (igual”) que haga verdadera la regla.