Examen - Programación 1

Instituto de Computación
Diciembre 2023

Leer con atención:

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje Pascal tal como fue dado en el curso.
- En todos los problemas se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso.
- Escribir las respuestas de un solo lado de la hoja.
- Entregar solamente las hojas de solución escritas a lápiz.
- En cada hoja entregada se debe incluir nombre, cédula y qué número de hoja es del total entregadas.

Ejercicio 1 (30 puntos)

Se consideran las siguientes declaraciones de constantes y tipos:

```
const MAX = ...; {valor mayor a cero}
type rango = 1..MAX;
     matriz = array[rango, rango] of integer;
```

Escribir una función:

```
function esIdentidad(m:matriz):boolean;
```

Que retorne `True` si la matriz `m` es la matriz identidad y `False` en caso contrario. La matriz identidad es aquella que contiene el valor 1 en todas las celdas de su diagonal y 0 en las demás.

Ejemplos de matrices identidad:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 \end{pmatrix}$$

Solución:

```
function esIdentidad(m:matriz):boolean;
var f, c : integer;
    identidad : boolean;
begin
    f := 1;
    repeat
        c := 1;
        repeat
            identidad := ( (c = f) and (m[f,c] = 1) )
                 or ((c<>f) and (m[f,c] = 0));
            c := c + 1;
        until (c > MAX) or (not identidad);
        f := f + 1;
    until (f > MAX) or (not identidad);

    esIdentidad := identidad;
end;
```

Ejercicio 2 (40 puntos)

Se define la siguiente estructura para representar los robots disponibles en un laboratorio de robótica:

```
const MAX = ...;
type
    TTipoRobot  = (Terreste, Manipulador, Aereo);
    TAgarre = (Garra, Ventosa);

    TRobot = record
        disponible : boolean;
        case tipo : TTipoRobot of
            Terreste: (cantRuedas:integer);
            Aereo:    (cargaUtil:integer);
            Manipulador:(agarre:TAgarre);

        end;

    TLab = record
        robots : array[1..MAX] of TRobot;
        tope : 0..MAX;
    end;

    TReparar = record
        indices : array[1..MAX] of integer;
        tope : 0..MAX;
    end;
```

a) (20 puntos)

Se busca enviar a reparación todos los robots `manipuladores` disponibles que cuentan con un `agarre` de tipo `garra`.

Implemente un procedimiento que cargue en el arreglo `repararIndices` los índices de todos los robots que:

- Estén disponibles
- Sean manipuladores
- Tengan un agarre de tipo garra

Al mismo tiempo, deberá actualizar el estado de disponibilidad de los robots, es decir, aquellos seleccionados para reparación deberán ser marcados como no disponibles.

```
procedure arreglarManipuladores(var stockLab:TLab;
    var repararIndices:TReparar);
```

Solución:

```
procedure arreglarManipuladores(var stockLab:TLab;
    var repararIndices:TReparar);
var i: integer;
begin
    repararIndices.tope := 0;
    for i := 1 to stockLab.tope do
```

```
        if (stockLab.robots[i].tipo = Manipulador)
            and (stockLab.robots[i].agarre = Garra)
            and (stockLab.robots[i].disponible)  then
        begin
            stockLab.robots[i].disponible := False;
            repararIndices.tope := repararIndices.
                tope + 1;
            repararIndices.indices[repararIndices.
                tope] := i;
        end;
```

b) (20 puntos)

Escribir una función que retorne el índice del primer robot terrestre con cantidad de ruedas `cr`, en caso de que este no exista retorna el valor -1.

```
function buscarRobotTerrestre(stockLab:TLab; cr:
    integer):integer;
```

Solución:

```
function buscarRobotTerrestre(stockLab:TLab; cr:
    integer):integer;
var i :integer;
begin
    i := 1;
    while (i <= stockLab.tope) and  not ((stockLab.
        robots[i].tipo = Terreste) and (stockLab.
        robots[i].cantRuedas = cr)) do
        i := i + 1;

    if (i <= stockLab.tope) then
        buscarRobotTerrestre := i
    else
        buscarRobotTerrestre := -1
end;
```

Ejercicio 3 (20 puntos)

Se define la siguiente estructura para representar la cantidad de libros y sus identificadores:

```
type
    ListaLibros = ^Libro;
    Libro = record
        id : integer;
        cantidad: integer;
        sig : ListaLibros;
    end;
```

Escribir un procedimiento

```
procedure agregarLibro(id: integer;var libros:
    ListaLibros);
```

que permita actualizar la lista `libros` de libros. En el caso de que un libro con el identificador `id` dado ya exista en la lista, se deberá aumentar la cantidad de dicho libro en uno, en caso contrario se deberá agregar el libro al inicio de la lista.

Solución:

```
procedure agregarLibro(id: integer;var libros:
    ListaLibros);
var aux: ListaLibros;
begin
    aux := libros;
    while (aux <> nil) and (aux^.id <> id) do
        aux := aux^.sig;

    if (aux <> nil)  then
        aux^.cantidad := aux^.cantidad + 1
    else
        begin
            new(aux);
            aux^.id := id;
            aux^.cantidad := 1;
            aux^.sig := libros;

            libros := aux
        end;
end;
```

Ejercicio 4 (10 puntos)

Dado el siguiente programa, indicar qué despliega la instrucción *writeln* cuando se lee el dígito de su cédula de identidad ANTERIOR al guión. Por ejemplo si su cédula es 1234567-8, se ingresa 7.

```
program examen;
var n, x : integer;

procedure p (n:integer; var y:integer);
begin
   n := n+1;
   y := n
end;

function f (x,y:integer) : integer;
begin
   x := x-y;
   f := x+7
end;

begin
   readLn(n);
   p (n*2, n);
   x := f (n, 2);
   writeln (n,' ',x)
end.
```

Solución:

Entrada	Salida
0	1 6
1	3 8
2	5 10
3	7 12
4	9 14

Entrada	Salida
5	11 16
6	13 18
7	15 20
8	17 22
9	19 24