AYED II
Lautaro Gastón Peralta

Práctico 2.1

1. Escribir un algoritmo que dada una matriz a: **array**[1..n,1..m] **of int** calcule el elemento mínimo. Escribir otro algoritmo que devuelva un arreglo **array**[1..n] con el mínimo de cada fila de la matriz a.

```
fun min_arr (a : array [1..n,1..m] of int) ret minimo : int
        minimo := a[1,1]
        for i:=1 to n do
                for j:=1 to m do
                        if a[i,j] < minimo then minimo:=a[i,j] fi
                od
        od
end fun

fun min_fil_arr (a : array [1..n,1..m] of int) ret min_fil : array [1..n] of int
        var minimo : int
        for i:=1 to n do
                minimo := a[i,1]
                for j:=1 to m do
                        if a[i,j] < minimo then minimo:=a[i,j] fi
                od
                min_fil[i] := minimo
        od
end fun
```

2. Dados los tipos enumerados

```
type mes = enumerate
                enero
                febrero
                ...
                diciembre
           end enumerate

type clima = enumerate
                Temp
                TempMax
                TempMin
                Pres
                Hum
                Prec
             end enumerate
```

El arreglo med:**array**[1980..2016,enero..diciembre,1..28,Temp..Prec] **of nat** es un arreglo multidimensional que contiene todas las mediciones estadísticas del clima para la ciudad de Córdoba desde el 1/1/1980 hasta el 28/12/2016. Por ejemplo, med[2014,febrero,3,Pres] indica la presión atmosférica que se registró el día 3 de febrero de 2014. Todas las mediciones están expresadas con números enteros. Por simplicidad asumiremos que todos los meses tienen 28 días.

(a) Dar un algoritmo que obtenga la menor temperatura mínima (TempMin) histórica registrada en la ciudad de Córdoba según los datos del arreglo.

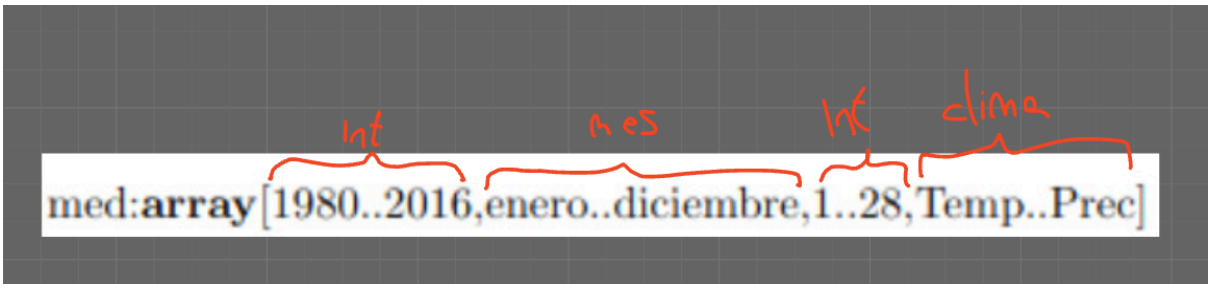
(b) Dar un algoritmo que devuelva un arreglo que registre para cada año entre 1980 y 2016 la mayor temperatura máxima (TempMax) registrada durante ese año.

(c) Dar un algoritmo que devuelva un arreglo que registre para cada año entre 1980 y 2016 el mes de ese año en que se registró la mayor cantidad mensual de precipitaciones (Prec).

(d) Dar un algoritmo que utilice el arreglo devuelto en el inciso anterior (además de med) para obtener el año en que ese máximo mensual de precipitaciones fue mínimo (comparado con los de otros años).

(e) Dar un algoritmo que obtenga el mismo resultado sin utilizar el del inciso (c).

(a)




```
fun min_temp_hist (med : array [1980..2016,enero..diciembre,1..28,Temp..Prec] of nat) ret
temp_min : nat
        temp_min:=med[1980,enero,1,TempMin]
        for i:=1980 to 2016 do
                for j:=enero to diciembre do
                        for k:=1 to 28 do
                                if med[i,j,k,TempMin] <= temp_min then
                                        temp_min:= med[i,j,k,TempMin]
                                fi
                        od
                od
        od
end fun
```

(b)

```
fun max_temp_anual (med : array [1980..2016,enero..diciembre,1..28,Temp..Prec] of nat)
ret tempmax_a : array [1980..2016] of TempMax
        var tempmax_aux : nat
        for i:=1980 to 2016 do
                tempmax_aux:=med[i,enero,1,TempMax]
                for j:=enero to diciembre do
                        for k:=1 to 28 do
                                if tempmax_aux <= med[i,j,k,TempMax] then
                                        tempmax_aux:= med[i,j,k,TempMax]
                                fi
                        od
                od
        tempmax_a[i]:=tempmax_aux
        od
end fun
```

(c)

```
fun max_prec_mensual (med : array [1980..2016,enero..diciembre,1..28,Temp..Prec] of
nat) ret maxprec_a : array [1980..2016] of mes
        var maxpres : nat
        var pres_mes_cont : nat
        for i:=1980 to 2016 do
                maxpres:=0
                maxprec_a[i]:=enero
                for j:=enero to diciembre do
                        pres_mes_cont:=0
```

```
                    for k:=1 to 28 do
                        pres_mes_cont:=pres_mes_cont + med[i,j,k,Prec]
                    od
                    if pres_mes_cont > maxpres then
                        maxpres:= pres_mes_cont
                        maxprec_a[i]:=j
                    fi
                od
            od
end fun
```

3. Dado el tipo

```
type person = tuple
                name: string
                age: nat
                weight: nat
              end tuple
```

(a) escribí un algoritmo que calcule la edad y peso promedio de un arreglo a : **array**$[1..n]$ **of** *person*.

(b) escribí un algoritmo que ordene alfabéticamente dicho arreglo.

(a)

defino previamente una tupla en donde almacenar mi resultado

```
type prom =         tuple
            edadpromedio : nat
            pesopromedio : nat
                end tuple
fun cal_prom (a : array[1..n] of person) ret res : prom
    res.edadpromedio:=0
    res.pesopromedio:=0
    for i:=1 to n do
        res.edadpromedio:= res.edadpromedio + a[i].age
        res.pesopromedio:= res.pesopromedio + a[i].weight
    od
    res.edadpromedio:= res.edadpromedio/n
    res.pesopromedio:= res.pesopromedio/n
end fun
```

(b)

```
proc ord_alf (in/out a:array[1..N] of person)
  var j : nat
  for i := 2 to N do
    j:=i
    while j>1 ^ (a[j].name < a[j-1].name) do
      swap(a,j-1,j)
      j:=j-1
    od
  od
end proc
```

4. Dados dos punteros p, q : **pointer to int**

   (a) escribí un algoritmo que intercambie los valores referidos sin modificar los valores de p y q.

   (b) escribí otro algoritmo que intercambie los valores de los punteros.

   Sea un tercer puntero r : **pointer to int** que inicialmente es igual a p, y asumiendo que inicialmente $*p = 5$ y $*q = -4$ ¿cuáles serían los valores de $*p$, $*q$ y $*r$ luego de ejecutar el algoritmo en cada uno de los dos casos?

"*p es una expresión que me da el valor guardado en el lugar de memoria apuntado por p. si p es de tipo pointer to T, entonces el tipo de *p es T." ejemplo. se puede escribir (*p + 24) * 2.

a) proc intercambiar_ref **(in/out** p,q **: pointer to** int**)**
```
    var temp : int
    temp := *p
    *p:=*q
    *q:=temp
  end proc
```
b)
```
  proc intercambiar_val (in/out p,q : pointer to int)
    var temp : pointer to int
    tem:=p
    p:=q
    q:=temp
  end proc
```

6. Escribir un algoritmo que dadas dos matrices a, b: **array**$[1..n,1..m]$ **of nat** devuelva su suma.

```
fun suma (a : array [1..N,1..M] of nat, b : array [1..N,1..M] of nat) ret c : array [1..N,1..M] of nat
    for i:=1 to N do
      for j:=1 to M do
        c[i,j]:=a[i,j] + b[i,j]
      od
    od
end fun
```

7. Escribir un algoritmo que dadas dos matrices a: **array**$[1..n,1..m]$ **of nat** y b: **array**$[1..m,1..p]$ **of nat** devuelva su producto.

```
fun prod_,mat (a : array [1..N,1..M] of nat, b : array [1..M,1..P] of nat) ret c : array [1..M,1..P] of nat
    for i:=1 to N do
      for j:=1 to P do
        c[i,j]:=0
        for k:=1 to M do
          c[i,j]:= c[i,j] + a[i,k] * b[k,j]
        od
      od
    od
end fun
```

```
fun max_ganancia (v : array[0..n,0..m] of int, D : nat) ret ganancia : float
        var valor_actual : float
        var max_futuro : float
        var acciones_compradas : float
        for i:=0 to n do
                for j:=0 to m-1 do
                        valor_actual := v[i,j]
                        max_futuro := valor_actual
                        for k:=j+1 to m do
                                if v[i,k] > max_futuro then
                                        max_futuro := v[i,k]
                                fi
                        od
                        if max_futuro > valor_actual then
                                acciones_compradas := D / valor_actual
                                ganancia := ganancia * (max_futuro - valor_actual)
                        fi
                od
        od
end fun
```