ALGORITMOS Y ESTRUCTURAS DE DATOS II	PRÁCTICO 2 PARTE 4 (PRÁCTICO 4) TIPOS CONCRETOS	2024

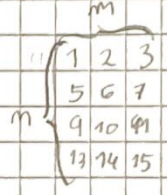
①

```
fun min(a: array[1..m, 1..m] of int) ret elem_min: int

    elem_min := a[1,1]

    For i:=1 To m do
      For j:=1 To m do
        If a[i,j] < elem_min →
          elem_min := a[i,j]
        Fi
      od
    od

end Fun
```

¿QUÉ HACE?
¿CALCULA EL ELEMENTO MÍNIMO DE UNA MATRIZ

¿CÓMO LO HACE?
RECORRE CADA ELEMENTO DE LA MATRIZ UNO POR UNO, COMPARÁNDOLO CON EL ELEMENTO MÍNIMO ENCONTRADO HASTA EL MOMENTO

```
fun minXFila(a: array[1..m, 1..m] of int) ret mins: array[1..m]

    Var min_actual: int

    For i:=1 To m do
      min_actual := a[i,1]
      For j:=1 To m do
        If a[i,j] < min_actual then
          min_actual := a[i,j]
        Fi
      od
      mins[i] := min_actual
    od

end Fun
```



②

2)

```
{- TEMPERATURA MÍNIMA HISTÓRICA -}
fun TempMin(med: array[1930..2016, enero..diciembre, 1..28, Temp..Prec] of nat) ret
                                                                  min_temp_his: int

    min_temp_his := med[1930, enero, 1, TempMin]

    For año := 1930 To 2016 do
      For mes := enero To diciembre do
        For día := 1 To 28 do
          If med[año, mes, día, TempMin] < min_temp_his then
            min_temp_his := med[año, mes, día, TempMin]
          Fi
        od
      od
    od
```

(2) b)

```
Fun TempMaxAño (med: array[1980..2016, enero..diciembre, 1..28, Temp..Prec] of nat) ret
                                                                  maxTempAño: array[1980..2016] of nat
  Var TempMaxAux: nat
  For Year := 1980 To 2016 do
    TempMaxAux = med[year, enero, 1, TempMax]
    For month := enero To diciembre do
      For day := 1 to 28 do
        if med[Year, month, day, TempMax] > TempMaxAux then
          TempMaxAux := med[year, month, day, TempMax]
        fi
      od
    od
    maxTempAño[year] := TempMaxAux
  od
end Fun
```

c)

```
Fun mes_mas_prec (med: array[1980..2016, enero..diciembre, 1..28, Temp..Prec] of nat) ret
                                                                  month_high_Prec: array[1980..2016] of mes
{- FUNCIÓN QUE DEVUELVE LA CANTIDAD DE PRECIPITACIONES EN UN MES -}

fun totalPrecMonth (med: array[1980..2016, ENERO..DICIEMBRE, 1..28, TEMP Prec] of nat, ret totalPrecMes: nat
                                                                  año: nat, mes: mes)

  Var totalPrec: nat
  TotalPrec := 0
  For day := 1 To 28 DO
    TotalPrec := totalPrec + med[año, mes, day, Prec]
  od

end Fun

{- FUNCIÓN QUE PEDÍA EL PROBLEMA -}

Fun mes_mas_Prec(med: array[1980..2016, enero..diciembre, 1..28, Temp..Prec] of nat)
                                                                  ret month_high_Prec: array[1980..2016] of mes
  Var month_Prec_Act: mes

  For Año := 1980 to 2016 DO
    month_Prec_act := enero
    For mes := enero to diciembre do
      if totalPrecMonth(med, año, mes) > totalPrecMonth(med, año, month_Prec_act) then:
        month_Prec_act := mes
      fi
    od
    month_high_Prec[año] := month_Prec_act
  od

End Fun
```

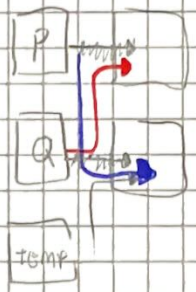
③ a)

```
Proc edadYpeso(in a: array[1..n] of persona, out pesoProm: nat, out edadProm nat)
    Var Pesototal: nat
    Var edadtotal: nat
    Pesototal := 0
    edadtotal := 0

    For persona := 1 to n do
        Pesototal := Pesototal + persona.weight
        edadtotal := edadtotal + persona.age
    od
    pesoProm := Pesototal / n
    edadProm := edadtotal / n

end Proc
```

b) Proc ordenado(in/out a: array[1..n] of Person) (completar)

④ a)

```
Proc intercambiar(in/out P,Q Pointer to int)
    Var temP: Pointer to int
    temP := Q
    Q := P
    P := temP

end Proc
```

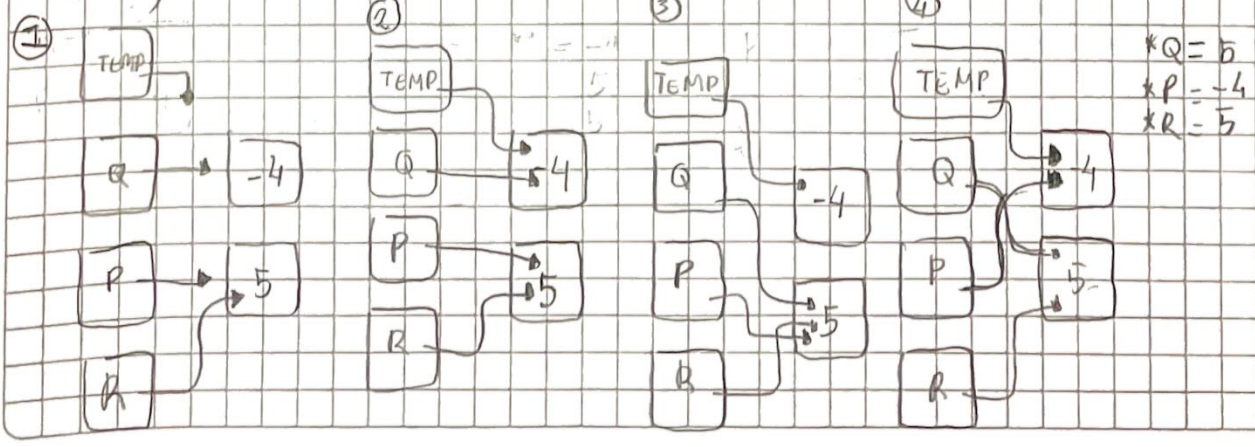


b)

```
Proc intercambiar2(in/out P,Q Pointer to int)
    Var temp: Pointe to int

    temp := *Q
    *Q := *P
    *P := temp
EndProc
```

Pointer to int

PASO 2)



⑤ a)
```
Fun lex_less (a,b array[1..m]) ret res: bool
    res := False
    Var i: nat
    i := 1
    Var b: bool
    b := True
    While (i≤m ∧ b) do
      If a[i] < b[i] →
          res := True
          b := False
      [] a[i] > b[i] →
          res := False
          b := False
      Fi
    od
endFun
```

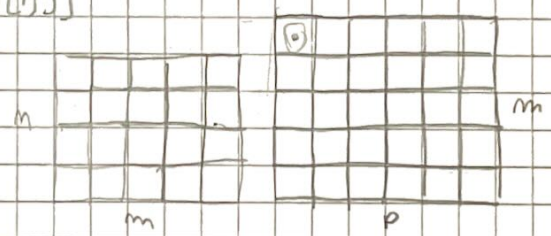
b)
```
Fun lex_less_or_equal (a,b: array[1..m] of nat) ret res: bool
    Var i: nat
    i := 1
    While (a[i] = b[i] ∧ i ≤ m) do
        i := i+1
    do
    ret := a[i] ≤ b[i] ∨ i > m
endFun
```

(5) c) fun rel compare (a,b array nat) ret res: ord

```
var i: nat
  i := 1
var b: bool
  b := true
res igual
while (i≤m ∧ b) do
  if a[i] < b[i] →
    res = menor
    b = false
  if a[i] > b[i] →
    res = mayor
    b = false
  od
endfun
```

(6) a) fun sum_mat (a,b: array[1..n, 1..m] of nat) ret sum: array[1..n, 1..m]

```
For i:=1 To n do
  For j:=1 To m do
    sum[i,j] := a[i,j] + b[i,j]
  od
od
endfun
```



(7) fun Prod_mat (a: array[1..n, 1..m] of nat, b: array[1..m, 1..p]) ret Prod: array[1..n, 1..p]

```
For rowA := 1 To n do
  FOR colB := 1 To p do
  For colA := 1 To m do {- TMB PODRÍA SER LAS FILAS DE B -}
    Prod[rowA, colB] := Prod[rowA, colB] + a[rowA, colA] * b[colA, colB]
  od
 od
od

endfun
```

// REVISAR PORQUE TENDRÍA QUE INICIALIZAR Prod CON 0 CREO