Práctico 4a
(de entrega obligatoria)

B. Gonzalez Kriegel - N. Wolovick

9 de mayo de 2000

Traducir a ensamlador ARC uno de los 4 códigos escritos en el lenguaje de los comandos guardados de Dijkstra (Dgcl). Estos programas fueron derivados en AyED1 y están en el apunte "Cálculo de Programas".

Se deberá seguir de la mejor manera posible la estructura del programa original, para ello deberá transformar de manera conveniente las variables, expresiones y estructuras de control, en instrucciones ARC.

El código, que será el resultado final del trabajo, deberá estar comentado, indicando las relaciones entre variables y arreglos con registros y posiciones de memoria, asi como que sentencia o parte de sentencia implementa cada instrucción del ARC ISA.

Además en los ejercicios del 0 al 4 habrá que definir un arreglo en la memoria para probar ejecuciones y en el caso del ejercicio 5 sólo basta con cargar los registros que representen x e y con el contenido de dos posiciones de memoria.

Enviar el código debidamente testeado en el ARCSim, junto con el número de grupo y nombres de los integrantes a `odc@hal.famaf.unc.edu.ar` antes del viernes 19 de Mayo, fecha del 2do. parcial. El ejercicio resuelto significa 2 puntos para cada integrante del grupo en el segundo parcial que será por un total de 8 puntos. Los programas enviados que tengan algún error podrán ser corregidos y reenviados cuantas veces sea necesario hasta el día y hora del parcial.

Los programas 4 y 5 tienen mayor dificultad, por lo que no son obligatorios. Sin embargo, las primeras *personas* que envien estos programas sin recibir ayuda alguna en la resolución, tendrán 2 puntos más en el parcial, además de los que puedan conseguir con el ejercicio grupal y el parcial escrito. No se permite que una misma persona acumule más de 4 puntos resolviendo estos ejercicios.

0 - Todos los Elementos Positivos (p.147 - ej.3a)

```
[ con N: Int, A: array [0,N) of Int; var n,r:  Int
  {R : N ≥ 0}
  n,r := 0,True
  {P : r = (∀.i : 0 ≤ i < n : A.i ≥ 0)}
  do
       n≠N → n,r := n+1,r and A.n≥0
  od
  {Q : r = (∀.i : 0 ≤ i < N : A.i ≥ 0)}
]
```

1 - Algún Elemento Positivo (p.147 - ej.3b)

```
[ con N: Int, A: array [0,N) of Int; var n,r:  Int
  {R : N ≥ 0}
  n,r := 0,False
  {P : r = (∃.i : 0 ≤ i < n : A.i ≥ 0)}
  do
       n≠N → n,r := n+1,r or A.n≥0
  od
  {Q : r = (∃.i : 0 ≤ i < N : A.i ≥ 0)}
]
```

2 - Máximo Elemento (p.147 - ej.4)

```
[ con N: Int, A: array [0,N) of Int; var n,r:  Int
  {R : N ≥ 0}
  n,r := 0,−∞
  {P : r = (Max.i : 0 ≤ i < n : A.i)}
  do
       n≠N → n,r := n+1,r max A.n
  od
  {Q : r = (Max.i : 0 ≤ i < N : A.i)}
]
```

3 - Cantidad de Elementos Pares (p.147 - ej.5)

```
[ con N: Int, A: array [0,N) of Int; var n,r:  Int
  {R : N ≥ 0}
  n,r := 0,0
  {P : r = (N.i : 0 ≤ i < n : A.i mod 2 = 0)}
  do
       n≠N → if
                  A.n mod 2 = 0 → n,r := r+1,n+1
                 □A.n mod 2 ≠ 0 → n := n+1
               fi
  od
  {Q : r = (N.i : 0 ≤ i < N : A.i mod 2 = 0)}
]
```

4 - Segmento de Suma Máxima (p.144)

```
[ con N: Int, A: array [0,N) of Int; var n,r,u:  Int
  {R : N ≥ 0}
  n,r := 0,0
  {P : r = (Max.p.q : 0 ≤ p ≤ q ≤ n : sum.p.q) ∧ 0 ≤ n ≤ N ∧ u = (Max.p : 0 ≤ p ≤ n : sum.p.n)}
  [sum.p.q = (∑ i : p ≤ i < q : A.i)]
  do
       n≠N → u := (A.n+u) max 0
               r := r max u
               n := n+1
  od
  {Q : r = (Max.p.q : 0 ≤ p ≤ q ≤ N : sum.p.q)}
  [sum.p.q = (∑ i : p ≤ i < q : A.i)]
]
```

5 - División Entera Mejorada (p.138)

```
[ var x,y,q,r,d,dd:  Int
  {R : x ≥ 0 ∧ y ≥ 0}
  q,r := 0,x
  {Q : x = q × y + r ∧ 0 ≤ r}
  do
       r≥y → d,dd := 1,y
             do
                 r≥ 2×dd → d,dd := 2×d,2×dd
             od
             q,r := q+d,r-dd
  od
  {Q : x = q × y + r ∧ 0 ≤ r ∧ r < y}
]
```