



## Desarrollo.

### 0.1 Ejercicio 1.

```
1 program ej1
2     !area de variables
3     implicit none
4     integer :: i
5     integer, dimension(20) :: Fibs
6     Fibs(1) = 0
7     Fibs(2) = 1
8     write (*,*) "Valor", 1, ":", Fibs(1)
9     write (*,*) "Valor", 2, ":", Fibs(2)
10    ! bucle desde i = 3, hasta i = 20, aumento de 1 en 1 (aunque no sea
11        necesario, es mejor que no sea "redundante" y este implicito)
12    do i = 3, 20, 1
13        Fibs(i) = Fibs(i - 2) + Fibs(i - 1)
14        write (*,*) "Valor", i, ":", Fibs(i)
15    end do
16 end program ej1
```

Listing 1: Ejercicio 1.

### Ejercicio 2.

```
1 program ej2
2     print *, "Torres de hanoi con 3 discos."
3     call hanoi(3, 'A', 'C', 'B')
4     contains
5         ! declaramos una subrutina recursiva por... , la neta quien sabe
6         ! porque se define asi xD, al parecer segun sus creadores, Fortran
7         ! nunca se penso como un lenguaje que use otro paradigma m s que el
8         ! iterativo, algo que tiene sentido ya que, fue pensado principalmente
9         ! (y casi unicamente al parecer) para trayectorias de misiles y
10        ! cohetes espaciales, estar enteramente pensado para la fisica... ;
11        ! que resuelve el problema de las torres de hanoi
12        recursive subroutine hanoi(n, Origen, Destino, Aux) ! auqnue dice que
13            ! se usen 3 parametros no especifica si el numero de postes se incluye
14            ! , ademas de que se hace complejo por que se pondrian poner 20 postes
15            ! y pensar que solo hubo 3 :v
16        integer :: n
17        character(len=1) :: Origen, Destino, Aux
18        ! Si hay solamente un disco, este se mueve directamente de origen a
19        ! destino
```

```

10     if (n .eq. 1) then
11         print *, Origen, "□-□", Destino
12     else
13         ! mover n-1 discos de origen a auxiliar usando el destino como
14           auxiliar
15         call hanoi(n-1, Origen, Aux, Destino)
16         ! mover el enesimo disco del origen al destino
17         print *, Origen, "□-□", Destino
18         ! mover al enesimo - 1 discos, del poste auxiliar al destino usando
19           origen como auxiliar
20         call hanoi(n-1, Aux, Destino, Origen)
21     end if
22 end subroutine hanoi
23 end program ej2

```

Listing 2: Ejercicio 2.

### Ejercicio 3.

```

1  program ej3
2      implicit none
3      integer :: i, j, n
4      logical :: p
5      print *, "Hasta□que□numero□deseas□saber?"
6      read (*,*) n
7      if (n .lt. 3) then
8          print *, "No□tiene□mucho□sentido□corroborrar□los□numeros□primos□menores
9             □a□3"
10     else
11         do i = 3, n, 1
12             p = .true.
13             if (mod(i, 2) .gt. 0) then
14                 do j = 2, i - 1, 1
15                     if (mod(i, j) .eq. 0) then
16                         p = .false.
17                         exit
18                     else
19                         p = .true.
20                     end if
21                 end do
22             else
23                 p = .false.
24             end if
25             if (p .eqv. .true.) then
26                 print *, i, "es□un□numero□primo"
27             end if
28         end do
29     end if
30 end program ej3

```

Listing 3: Ejercicio 3.

## Resultados.

Fotografías del programa en ejecución

Ejercicio 1.

```
TS git:(master) x gfortran e11.f98
Valor 1 : 1
Valor 2 : 1
Valor 3 : 1
Valor 4 : 2
Valor 5 : 3
Valor 6 : 5
Valor 7 : 8
Valor 8 : 13
Valor 9 : 21
Valor 10 : 34
Valor 11 : 55
Valor 12 : 89
Valor 13 : 144
Valor 14 : 233
Valor 15 : 377
Valor 16 : 610
Valor 17 : 987
Valor 18 : 1597
Valor 19 : 2584
Valor 20 : 4181
TS git:(master) x |
```

Figure 1: Ejercicio 1

Ejercicio 2.

```
TS git:(master) x gfortran e22.f98
Torres de Hanoi con 3 discos.
A -> C
A -> B
B -> C
A -> C
TS git:(master) x |
```

Figure 2: Ejercicio 2.

Ejercicio 3.

```
#
¿hasta que número deseas saber?
43
3 es un número primo
5 es un número primo
7 es un número primo
11 es un número primo
13 es un número primo
17 es un número primo
19 es un número primo
23 es un número primo
29 es un número primo
31 es un número primo
37 es un número primo
41 es un número primo
43 es un número primo
¿13 es primo? ¿si/no
si
¿hasta que número deseas saber?
73
3 es un número primo
5 es un número primo
7 es un número primo
11 es un número primo
13 es un número primo
17 es un número primo
19 es un número primo
23 es un número primo
29 es un número primo
31 es un número primo
37 es un número primo
41 es un número primo
43 es un número primo
47 es un número primo
53 es un número primo
59 es un número primo
61 es un número primo
67 es un número primo
71 es un número primo
73 es un número primo
79 es un número primo
83 es un número primo
89 es un número primo
97 es un número primo
73 es primo ¿si/no
```

Figure 3: Ejercicio 3.