Universidad Nacional Autónoma de México
Facultad de Ciencias
Materia: Computación | Grupo: 8414
Tarea 3
Autor: E. Alexeiv Rosales Guillen
Fecha: 14 de octubre del 2024

Desarrollo.

0.1 Ejercicio 1.

```
program ej1
    !area de variables
    implicit none
    integer :: i
    integer, dimension(20) :: Fibs
    Fibs(1) = 0
    Fibs(2) = 1
    write (*,*) "Valor", 1, ":", Fibs(1)
    write (*,*) "Valor", 2, ":", Fibs(2)
    ! bucle desde i = 3, hasta i = 20, aumento de 1 en 1 (aunque no sea
       necesario, es mejor que no sea "redundante" y este implicito)
    do i = 3, 20, 1
        Fibs(i) = Fibs(i - 2) + Fibs(i - 1)
        write (*,*) "Valor ", i, ":", Fibs(i)
    end do
end program ej1
```

Listing 1: Ejercicio 1.

Ejercicio 2.

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

```
        if (n .eq. 1) then
            print *, Origen, "␣-␣", Destino
        else
            ! mover n-1 discos de origen a auxiliar usando el destino como
               auxiliar
            call hanoi(n-1, Origen, Aux, Destino)
            ! mover el enesimo disco del origen al destino
            print *, Origen, "␣-␣", Destino
            ! mover al enesimo - 1 discos, del poste auxiliar al destino usando
                origen como auxiliar
            call hanoi(n-1, Aux, Destino, Origen)
        end if
    end subroutine hanoi
end program ej2
```

Listing 2: Ejercicio 2.

Ejercicio 3.

```
program ej3
    implicit none
    integer :: i, j, n
    logical :: p
    print *, "Hasta␣que␣numero␣deseas␣saber?"
    read (*,*) n
    if (n .lt. 3) then
        print *, "No␣tiene␣mucho␣sentido␣corroborrar␣los␣numeros␣primos␣menores
           ␣a␣3"
    else
        do i = 3, n, 1
            p = .true.
            if (mod(i, 2) .gt. 0) then
                do j = 2, i - 1, 1
                    if (mod(i, j) .eq. 0) then
                        p = .false.
                        exit
                    else
                        p = .true.
                    end if
                end do
            else
                p = .false.
            end if
            if (p .eqv. .true.) then
                print *, i, "es␣un␣numero␣primo"
            end if
        end do
    end if
end program ej3
```

Listing 3: Ejercicio 3.

Resultados.

Fotografías del programa en ejecución

Ejercicio 1.

Figure 1: Ejercicio 1

Ejercicio 2.

Figure 2: Ejercicio 2.

Ejercicio 3.

Figure 3: Ejercicio 3.