aprenderaprogramar.com

Ejercicio ejemplo de programación en pseucódigo con condicionales y bucles (CU00210A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº9 del Curso Bases de la programación Nivel II

24

EJERCICIO:

Replantear el ejercicio del anterior epígrafe del curso y reescribir el pseudocódigo buscando:

1. Evitar la doble subordinación del núcleo del programa.
2. Bloqueo del producto de los módulos.

SOLUCIÓN:

```
PROGRAMA SUC03 [Pseudocódigo aprenderaprogramar.com]

    Variables
        Enteras: E
        Reales: Dato

    1. Inicio
        2. Mostrar "Elija 1.Cálculo  2.Salir" : Pedir E
        3. Mientras E = 1
                Llamar EntraDatos
                Llamar Proceso(Dato) PorValor
                Mostrar "Elija 1.Cálculo  2.Salir" : Pedir E
           Repetir
    4. Fin

    Módulo EntraDatos
        1. Hacer
                Mostrar "Por favor introduzca número entero entre 0 y 100"
                Pedir Dato
                Dato = Redondear(Dato)
           Repetir Mientras Dato < 0 ó Dato > 100
    FinMódulo

    Módulo Proceso(Num: Enteros)
        Variables
                Enteras: Datobase
                Reales: Raiz01, Raiz02, Suce
        1. Datobase = Num
        2. Suce = 0
        3. Raiz01 = SQR(Num)
        4. Raiz02 = – Raiz01
        5. Mientras Num > 0 Hacer
                Suce = Suce + SQR(Num)
                Num = Num – 1
           Repetir
        6. Mostrar "Datobase =", Datobase
        7. Mostrar "Raiz01=", Raiz01
        8. Mostrar "Raiz02 =", Raiz02
        9. Mostrar "Valor de la suma =", Suce
    FinMódulo
```

Comentarios: Este programa cuenta con cuatro variables locales por ninguna del programa anterior (*SUC02*) y con un módulo de producto cerrado por ninguno del programa anterior. Analizaremos los cambios y sus implicaciones:

- En el algoritmo principal se ha eliminado la doble subordinación, apreciándose una mayor claridad. Por ese lado, de acuerdo. Pero por otro, téngase en cuenta que ahora una entrada de usuario distinta de 1 lleva siempre a salir del programa, mientras que en *SUC02* la única entrada que llevaba a salir era 2. Es decir, si el usuario pone "5", sale del programa. Esto no es demasiado deseable, pero habrá de ser el programador quien valore si es aceptable o no. Las alternativas estarían de nuevo en la doble subordinación o en el control directo del flujo.

- El módulo EntraDatos no se ha modificado y en cambio el módulo Proceso reúne dos módulos anteriores: Proceso y Resultados. La entrada de datos generalmente nos interesará tenerla "aparte", de forma que podamos variar entre que el dato lo proporcione el usuario o que sea leído de un archivo, variables, etc. sin tener que hacer más modificaciones.

La unificación de *Proceso* y *Resultados* en un módulo nos permite trabajar con variables locales y ofrece un producto cerrado. Esto nos puede interesar o no: depende de las circunstancias.

Se ha introducido la variable local Datobase, que es una copia del valor que se pasa como dato de entrada para el módulo genérico. En este caso, es una copia de la variable global Dato. Datobase nos permite realizar manipulaciones conservando el valor original, que es mostrado al final. Nótese que disponer de Datobase nos permitiría prescindir de las variables Raiz01 y Raiz02 ya que podrían ser sustituidas por las expresiones primitivas. En este caso las líneas 7 y 8 serían:

7. Mostrar "Raiz01 =", SQR(Datobase)

8. Mostrar "Raiz02 =", −SQR(Datobase)

Usar o no Raiz01 y Raiz02 será decisión del programador en función de las circunstancias.

¿Podríamos prescindir de Datobase usando Dato?

No, una variable transferida no debe aparecer dentro del módulo al que se ha transferido. Independientemente de que se haya transferido por valor o por variable. Hacerlo supondría introducir confusión y resultados impredecibles: lo contrario de lo que debe hacer un programador.

¿Podríamos llamar al módulo Proceso por variable en vez de por valor?

Sí. En este caso nos es indiferente el valor final que quede almacenado en Dato puesto que no lo volvemos a usar. Llamar a los módulos por valor siempre es más seguro. El motivo: de este modo estamos seguros de que no hay manipulaciones de la variable y nos evitamos sorpresas (que aparezca un director comercial modificando cosas sin permiso del director general). Sin embargo puede suponer mayores requerimientos de memoria, aspecto a tener en cuenta por el programador.

Veamos los resultados que generaría el programa en distintos supuestos:

Dato	Datobase	Raiz01	Raiz02	Valor Suma
0	0	0	0	0
1	1	1	– 1	1
2	2	1,414	– 1,414	2,414
5	5	2,236	– 2,236	8,382
9	9	3	– 3	19,306
20	20	4,472	– 4,472	61,666
50	50	7,071	– 7,071	239,036
85	85	9,219	– 9,219	526,847

La comprobación manual es sencilla para valores bajos y tediosa para valores altos.

Próxima entrega: CU00211A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=36&Itemid=60