aprenderaprogramar.com

Elegir tipos de datos ¿Qué tipo de variable usar al programar? Ejemplos resueltos (CU00206A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº5 del Curso Bases de la programación Nivel II

24

Cuando exista duda en los valores numéricos que podrá adoptar una variable, la declararemos como real. Sólo cuando estemos seguros de su carácter entero, como puede ser por ejemplo con los contadores para bucles, las declararemos como tales. Para transferir datos desde una variable real *A a* una variable *B* habremos de declarar *B* como real, en vez de entera. Se aceptará como alternativa declarar *B* como entera y usar cualquier estrategia que garantice que la variable real transfiere un valor entero como:

B = Redondear(A)

B = Truncar(A)

Veamos un ejemplo de lo que sería declaración de variables en un módulo:

```
Módulo Positiv
    Variables
        Enteras: Valor
        Reales: Numero
    1. Mostrar "Por favor, introduzca un número"
    2. Pedir Numero
    3. Valor = ABS(Redondear(Numero))
    4. Mostrar Valor
FinMódulo
```

Dentro de este módulo se han definido dos variables: una de tipo entero llamada Valor y otra de tipo real llamada Numero. Se transfieren datos desde una variable real a una entera gracias a la eliminación de la parte decimal que pudiera existir.

Al no existir ningún módulo llamado desde el módulo Positiv, las variables locales declaradas en su cabecera son de uso exclusivo para dicho módulo. Una asignación de valor o invocación de estas variables en el algoritmo principal u otro módulo daría lugar a "Error por variable no declarada". Vamos a realizar un ejemplo de aplicación de lo visto hasta ahora.

Se desea realizar un programa que calcule la suma de los términos de la sucesión:

$$\frac{1}{a}, \frac{1}{a-1}, \frac{1}{a-2}, \ldots, \frac{1}{1}$$

siendo a un número entero comprendido entre 1 y 100.

Vamos a tratar de aplicar el enfoque modular y los conceptos de declaración de variables, variables globales y locales, algoritmo principal y módulo. Dispondremos un módulo de entrada de datos en el que se le pide al usuario un número entero comprendido entre 1 y 100. No estamos completamente seguros de que el usuario nos vaya a hacer caso. Por ello estableceremos un mecanismo de seguridad consistente en:

- Si el número introducido es inferior o igual a cero, o mayor o igual que 101, se repite la solicitud de introducción de un número.

- Si el número introducido está en el rango válido pero es decimal, se procederá a su redondeo. Para poder aceptar el decimal la variable es real.

En otro módulo dispondremos el cálculo de la suma. Por un lado, la variable Suma será un acumulador que en cada pasada del bucle añade un término. Por otro lado, la propia variable a será un contador decreciente que define el denominador del término a añadir al acumulador. Otro contador independiente, i, será una variable local que contabiliza cuántos términos se han añadido al acumulador.

En cuanto al algoritmo principal, dirige las llamadas a los módulos y muestra el resultado final.

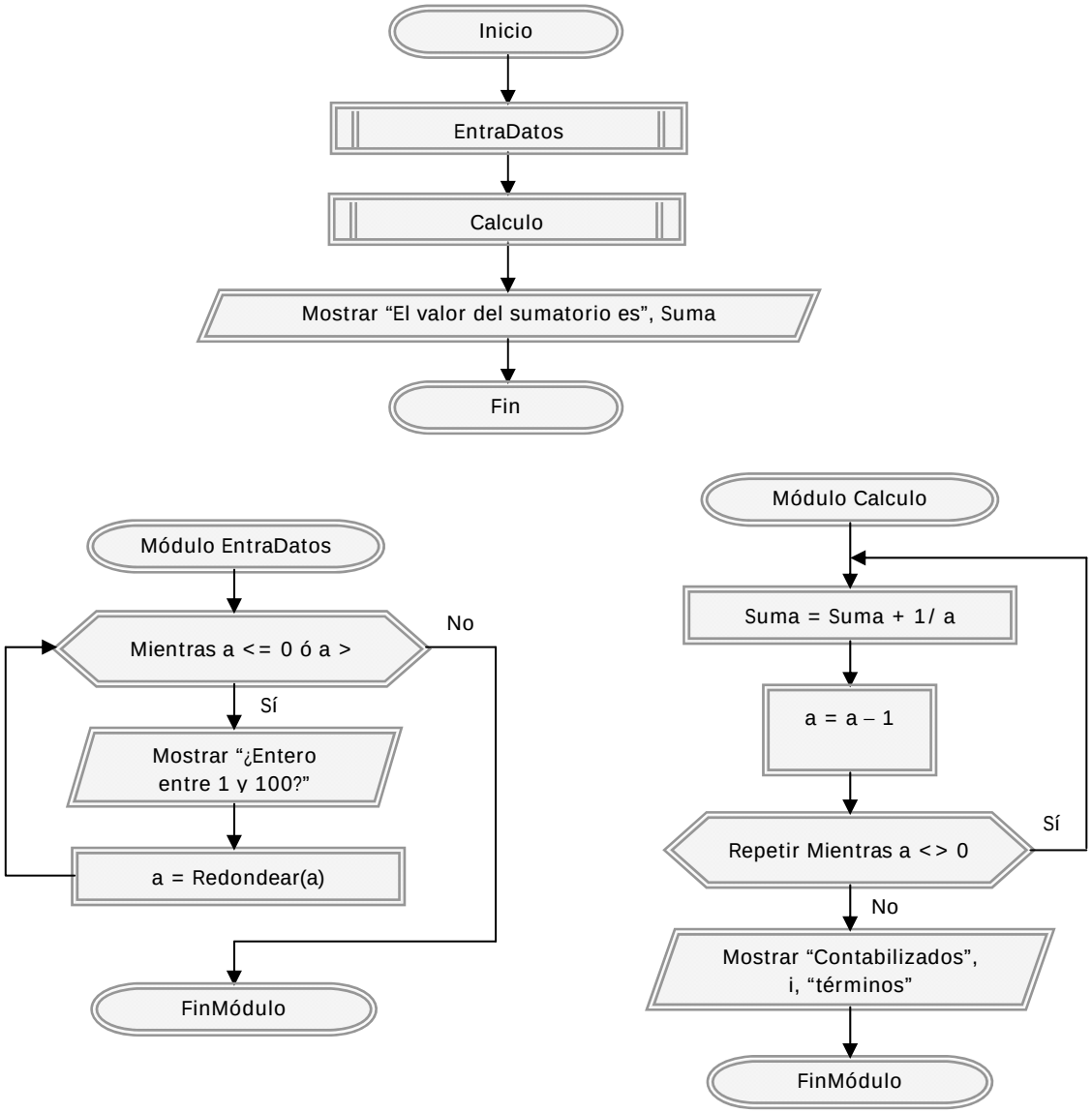
```
PROGRAMA SUC 01 [Ejemplo aprenderaprogramar.com]

    Variables
        Reales: a, Suma

        1. Inicio
            2. Llamar EntraDatos
            3. Llamar Calculo
            4. Mostrar "El valor del sumatorio es", Suma
        5. Fin

        Módulo EntraDatos
            1. Mientras a <= 0 ó a > 100 Hacer
                1.1 Mostrar "Por favor introduzca un número entero comprendido
                        entre 1 y 100"
                1.2 Pedir a
                1.3 a = Redondear(a)
               Repetir
            2. Mostrar "El dato base es", a
        FinMódulo

        Módulo Calculo
            Variables
                Enteras: i
            1. Hacer
                1.1 Suma = Suma + 1 / a
                1.2 a = a – 1
                1.3 i = i + 1
               Repetir Mientras a <> 0
            2. Mostrar "Contabilizados", i, "términos"
        FinMódulo
```

Comentarios: El programa trabaja con tres variables: *a, Suma* y *i*. Las variables *a* y *Suma* son de tipo real, y por estar declaradas en la cabecera del programa son variables globales. La variable *i* es de tipo entero y por estar declarada en la cabecera de un módulo es una variable local de dicho módulo.

Las variables globales pueden ser invocadas en cualquier parte del programa. Las locales sólo en el módulo en que se han declarado (o en varios módulos si son llamados desde el módulo donde se han declarado). Supongamos que en el algoritmo principal hubiéramos escrito:

4. Mostrar "El valor del sumatorio es", Suma, "con", i, "términos"

El resultado sería: error por variable invocada no declarada. El interés de declarar variables locales radica en repartir el peso del programa en los subprogramas para una mayor eficiencia y facilidad de comprensión. Pero hay que recordar que dichas variables están restringidas a ciertas partes del programa. Al igual que con la declaración de tipos, tenemos que estar seguros o escoger la opción más amplia entre las posibles. El diagrama de flujo sería el siguiente:



El algoritmo principal controla la ejecución de instrucciones y la entrada en acción de los distintos módulos. El módulo EntraDatos no tiene variables locales. Por tanto, trabaja únicamente con variables globales. El módulo Calculo tiene una variable local y trabaja tanto con esta variable local como con las variables globales.

La variable a se declara como real por desconocerse cómo actuará el usuario con seguridad. Se espera que introduzca un número entero, pero por error o confusión podría introducir un número no deseado. Aunque esté declarada como real para evitar problemas con la entrada, se forzará que sea un entero a través de las instrucciones definidas en el módulo EntraDatos.

La variable i en realidad no es necesaria. Tal y como está planteado el programa el número de términos del sumatorio necesariamente coincide con el dato base a. Es decir, si el dato base es 7 el número de términos será 7. Siendo puristas, deberíamos eliminarla para evitar procesos innecesarios. Pero por otro lado, podríamos buscarle un sentido a su existencia. Téngase en cuenta que si consideramos que el número de iteraciones es el dato base estamos trabajando con un argumento lógico, razonado, esperado... Mientras que el número de iteraciones que nos indica i no es un valor lógico, razonado, esperado... sino un valor reflejo de lo que realmente se ha producido al procesar el programa. Este matiz puede tener cierta importancia con programas largos y complejos, sobre todo de cara a la comprobación de que el funcionamiento real sea igual al esperado.

El programa consta del algoritmo principal y dos módulos. ¿Por qué no tres módulos para representar las tres fases de Datos → Cálculo → Resultados? Cuando creamos módulos buscamos:

- Facilitar la escritura: en cada módulo nos concentramos en lograr el mejor diseño posible para una parte del programa.
- Facilitar la lectura: resumimos en una palabra un proceso que consta de varios pasos.
- Facilitar la corrección: un error será achacable a un módulo del programa, sin ser necesaria su revisión completa.

En el programa que nos ocupa, crear un módulo de resultados no resulta interesante si supone una única línea. Con su creación no estaríamos resumiendo nada, puesto que ya está resumido. Quizás supondría al contrario, un enrevesamiento o exceso de llamadas. En definitiva, siempre podemos contar con el algoritmo principal para agrupar en él instrucciones breves entre módulos o procesos de muy pocas instrucciones.

Atendamos ahora al “recorrido” que sigue la variable a:

1º) Adquiere un valor real o entero según proceda el usuario.
2º) Se transforma en entero (*a = Redondear(a)*).
3º) Si no está en el rango deseado se fuerza a que esto ocurra.
4º) Se tiene un dato base.
5º) Sucesivamente aporta valores para el cálculo de la suma de la sucesión transformándose en *a – 1, a – 2, ..., a – n, 0* (salida del bucle).

En definitiva, a se ha transformado en cero y no tenemos registro de cuál era el valor del dato base.

Por otro lado, ¿Qué pasaría si en vez de realizar el proceso a la variable a tuviéramos que hacerlo con tres variables a, b ó c? Quizás la opción más razonable parece la de una variable correo. Pero, en general, no será necesario ya que podremos crear módulos genéricos en los cuales la variable que se procesa será indicada oportunamente, sin necesidad de que tenga un nombre específico. A continuación estudiaremos tanto la transformación de variables dentro de los módulos como la creación de módulos genéricos.

Próxima entrega: CU00207A

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