Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial. Turno de tarde. 2020-2021
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 6.1. Múltiplos, con registros.

Copia tu programa `~/fpi/practica04/multiplos.pas` en el fichero `~/fpi/practica06/multiplos.pas` y modifícalo de la siguiente forma: el programa original tendría uno función (o más de una) con tres paramétros: c1, c2 y c3. Cambia esa función (o funciones) reemplazando estos parámetros por uno solo: un registro, con tres campos.

Práctica 6.2. Uso de registros.

En la práctica 5.1 hiciste un programa para resolver un problema matemático sencillo. Ahora harás un programa similar, con algunos cambios

1. La expresión será distinta a la que usaste en aquel ejercicio. Busca otro ejemplo, parecido.
2. El número de parámetros de la expresión será dos o tres. Por ejemplo calcular el volumen de una esfera no serviría, porque solo depende de un parámetro (el radio).
3. Este ejemplo tiene que tener precondiciones matemáticas. Esto es, valores para los que no puede calcularse la expresión por algún motivo matemático: un cero en un cociente, un número negativo en una raiz, un número fuera del rango -1,1 en un arcoseno, etc.
4. El programa tendrá un procedimiento para leer desde el teclado los valores necesarios. Se llamara *lee_datos* o *read_data*.
   - Tendrá un parámetro de salida, que será un registro conteniendo los valores leídos por teclado.
   - Otro parámetro de salida, booleano, indicará con *TRUE* que el usuario ha seguido las instrucciones y los valores son correctos. Un valor *FALSE* significará que los valores son incorrectos, por ejemplo porque al usuario se le pidió un real y escribió una cadena. Observa que esto es una precondición: *el usuario escribe lo que se le pide*.
5. En tu programa habrá una función llamada *calculo_principal* (o *main_operation*). Esta función:
   - Tendrá un único parámetro de entrada. Será un registro, con dos o tres campos conteniendo los datos de entrada.
   - Devolverá un registro, con dos campos:
     - Un booleano que valdrá *TRUE* si se ha podido calcular la expresión porque las precondiciones matemáticas se cumplían, o *FALSE* en caso contrario.
     - La solución al problema, si es que las precondiciones se cumplían. Si no se cumplían, contendrá un valor indeterminado (esto es, no importa qué valor contiene, no debería consultarse)
   - Naturalmente, llamará a los subprogramas que pueda necesitar.
6. Habrá un procedimiento llamado *muestra_resultado / display_result* que escriba en pantalla la solución del problema, o un error porque no se cumplen las precondiciones matemáticas.

Observaciones

- Naturalmente, si no se cumple la precondición *el usuario escribe lo que se pide*, ya no se llamará a *calculo_principal / main_data*

- El diseño propuesto aquí es muy obvio: un subprograma para leer datos, otro para calcularlos y un tercero para mostrarlos. Si tuvieras más experiencia, la especificación no entraría en estos aspectos tan básicos, es lo que haría cualquier programador con unas mínimas habilidades. Recuerda esta estructura, es muy común y deberías aplicarla con frecuencia en el futuro.
- Observa que el programa usa dos tipos de registro:
  - Uno para contener los datos.
  - Otro para la salida de la función que hace los cálculos. Esta estructura es muy común, deberías usarla con frecuencia en el futuro [1]: cualquier subprograma puede fallar en cualquier momento, así que es típico devolver siempre dos cosas: un campo indicando éxito/error y otro (u otros) con los resultados propiamente dichos.

[1] excepto si usas 'excepciones', una técnica de programación más avanzada.