aprenderaprogramar.com

Comentarios al ejercicio de programación modular con pseudocódigo (CU00209A)

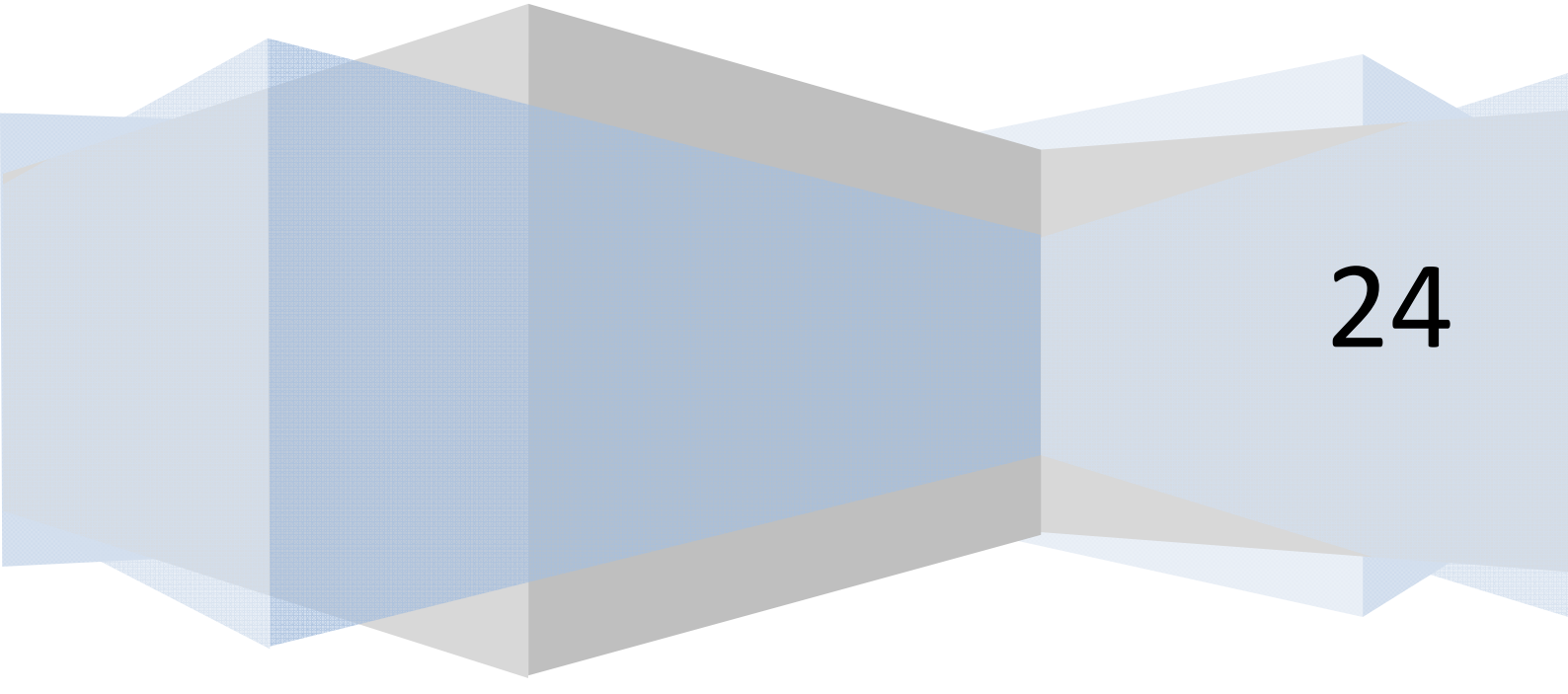
Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº8 del Curso Bases de la programación Nivel II



24

En el anterior epígrafe del curso hemos planteado un ejercicio de programación modular con pseudocódigo y su solución. Vamos a realizar algunos comentarios sobre la solución planteada.

Comentarios: El programa consta de algoritmo principal y tres módulos. Maneja cinco variables globales, ninguna local y un parámetro. Las variables son cuatro tipo real y una tipo entera. El parámetro es tipo entero. El algoritmo principal muestra una rama izquierda de "sí proceso" y una rama derecha de "no proceso". Los módulos que son el núcleo del programa se encuentran subordinados a un *Si ... Entonces.*

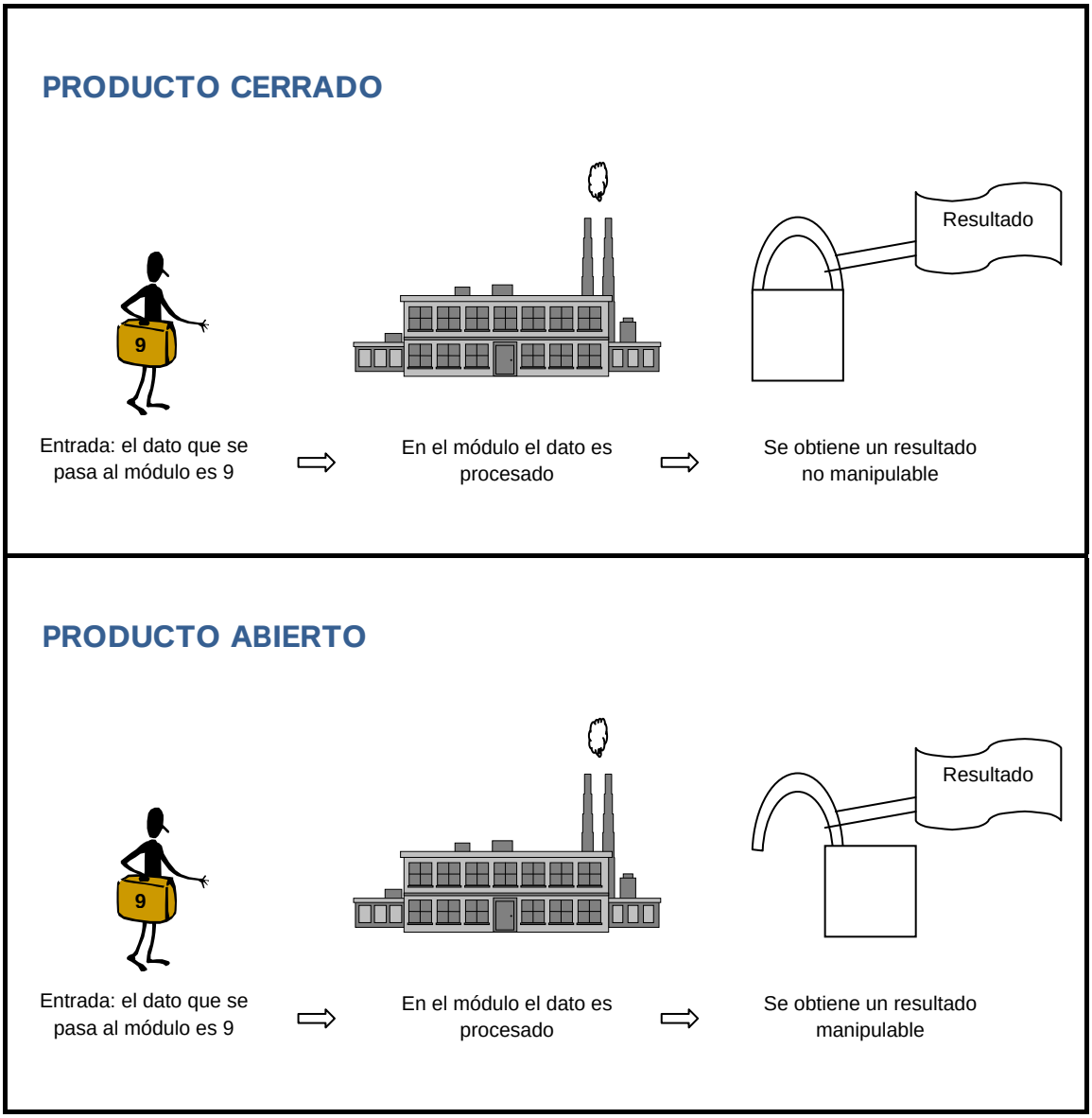
Dado que el módulo genérico *Proceso* nos pide un entero, en el módulo *EntraDatos* se fuerza a la variable real *Dato* a convertirse en entero (si no lo es ya). En cambio, la variable *E*, siendo una entrada de usuario, se espera entera. Podríamos haberla protegido de igual manera. Las entradas de usuario, datos de un archivo, etc. son muchas veces problemáticas por no obtener de ellas lo que se espera. Podemos programar mecanismos que eviten el fallo, pero cuando los programas son muy largos quizás resulta demasiado trabajoso prever todo lo que puede pasar. Por ello, para este tipo de problemas y para muchos otros existe una rama de la programación a la que denominamos "Gestión de errores", que estudia y permite abordar estas situaciones. Los diferentes lenguajes nos proporcionan herramientas "preventivas", "correctivas" y "curativas" frente a los errores. Hablaremos de ello más adelante.

Uno de los aspectos interesantes de los módulos es el poder dotarlos de carácter realmente independiente: recibir entradas y dar lugar a productos, pero productos no manipulables por nadie, ni por otro módulo ni por el algoritmo principal.

Supongamos un ejecutivo de una gran empresa que siendo director general tiene a su cargo a tres personas: el director de producto, el director comercial y el director de logística. El director general prepara un archivo con una serie de instrucciones y el objetivo de producción para el año: 40000 unidades. Y lo sitúa en el servidor de la empresa, al que tienen acceso todos los mandos. Habla por teléfono con el director de producto y le dice: "mira el archivo a ver qué te parece". Tras mirarlo, el director de producto indica que es necesario cambiar una instrucción imposible de cumplir y que hay que reducir el objetivo de producción a 38000 unidades, lo cual hace él mismo tras ser autorizado. El proceso se repite hasta que el director de logística hace las últimas correcciones y le pasa el documento al departamento de compras para realizar los primeros encargos en base a instrucciones y objetivo de producción. ¿Cuál es el problema? El que haya múltiples vías de modificación puede dar lugar a errores o malos entendidos. Por ejemplo, que con o sin mala intención se haga una modificación de la que no tiene conocimiento el director general y con la que no está de acuerdo.

Algo así nos puede pasar con un módulo cuyos resultados se pueden manipular desde cualquier punto del programa, como el módulo Proceso del ejemplo. Porque si en el módulo Resultados escribimos Raiz01 = 0 estaremos creando un malentendido: en un módulo se manipula equivocadamente lo que ha dicho otro. ¿Cuál sería la solución? Sin excluir otros recursos una alternativa pude ser independizar el módulo de forma que no haya acceso a sus procesos o resultados desde otro módulo. Así podríamos hablar de módulos de producto abierto y módulos de producto bloqueado o cerrado en función de si sus resultados son variables globales (manipulables fuera del módulo) o locales (no manipulables fuera). Además de protegernos contra manipulaciones, utilizar variables locales nos permite ahorrar recursos de nuestro ordenador.

En resumen, conviene recordar que las variables globales son “peligrosas”, sobre todo cuando los programas son largos. Por ello podemos decir que el uso de parámetros y de variables locales son buenas prácticas de programación, ya que nos permiten alcanzar en nuestros programas características que hemos citado como propias de la programación modular. A saber, independencia de funcionamiento entre módulos, posibilidad de traslado o copia de organización y facilidad de organización y comprensión.



En el ejemplo que venimos viendo Raiz01, Raiz02 y Suce podrían haber sido variables locales de un módulo llamado ProcesoyResultados

Una última cuestión, obsérvese que el acumulador Suce hay que “resetearlo” cada vez que se entra en el proceso. En caso contrario el resultado sería erróneo por estar acumulando resultados anteriores.

Próxima entrega: CU00210A

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