aprenderaprogramar.com

Ejercicio algoritmo de manejo de lista de datos. Parte II. Diagramas de flujo. (CU00220A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

Fecha revisión: 2024

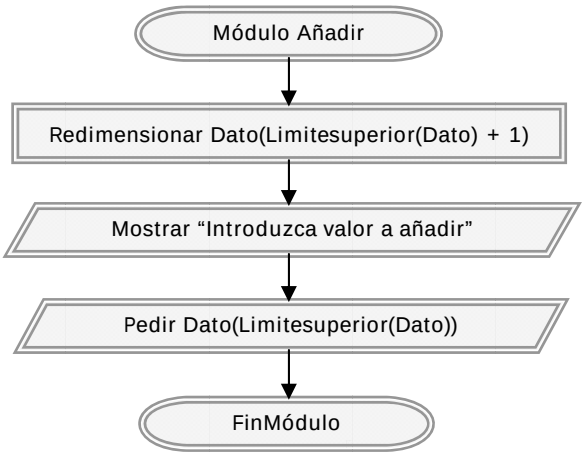
Autor: Mario R. Rancel

Resumen: Entrega nº 19 del Curso Bases de la programación Nivel II

24

CONTINUACIÓN DEL EJERCICIO "ALGORITMO PARA MANEJO DE LISTA DE DATOS"

Continuamos con el ejercicio que habíamos empezado en la entrega anterior. Vamos a ver los diagramas de flujo de los módulos:



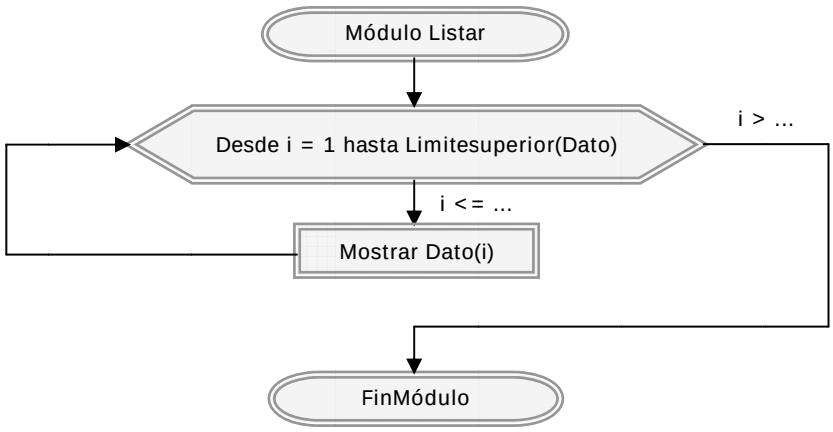
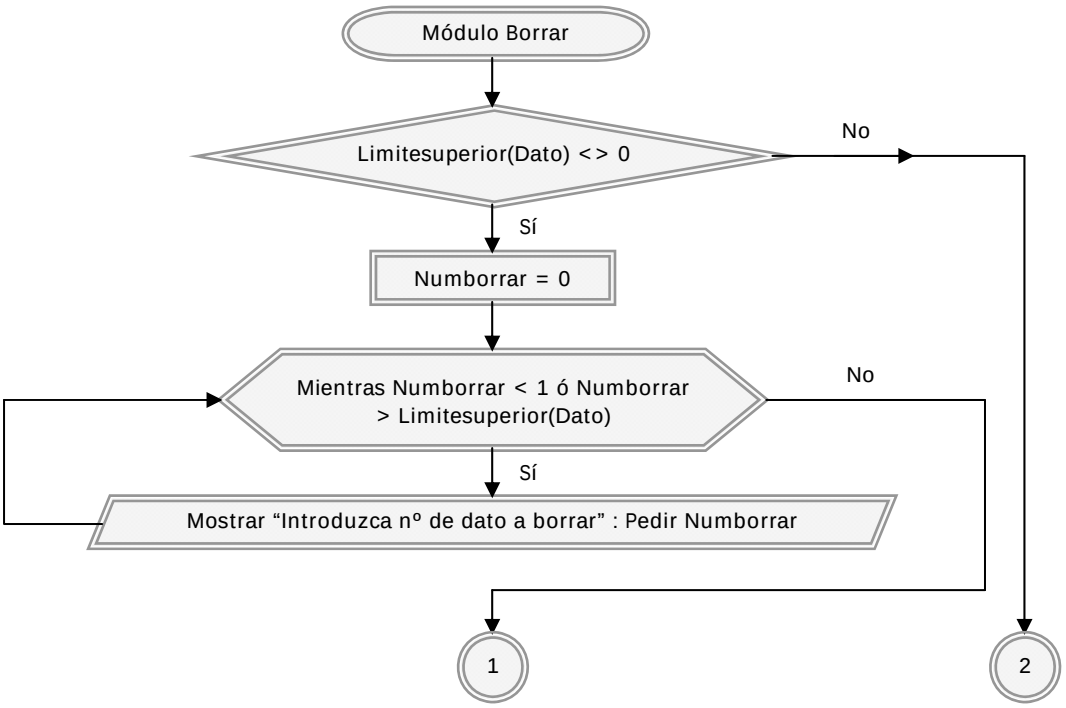
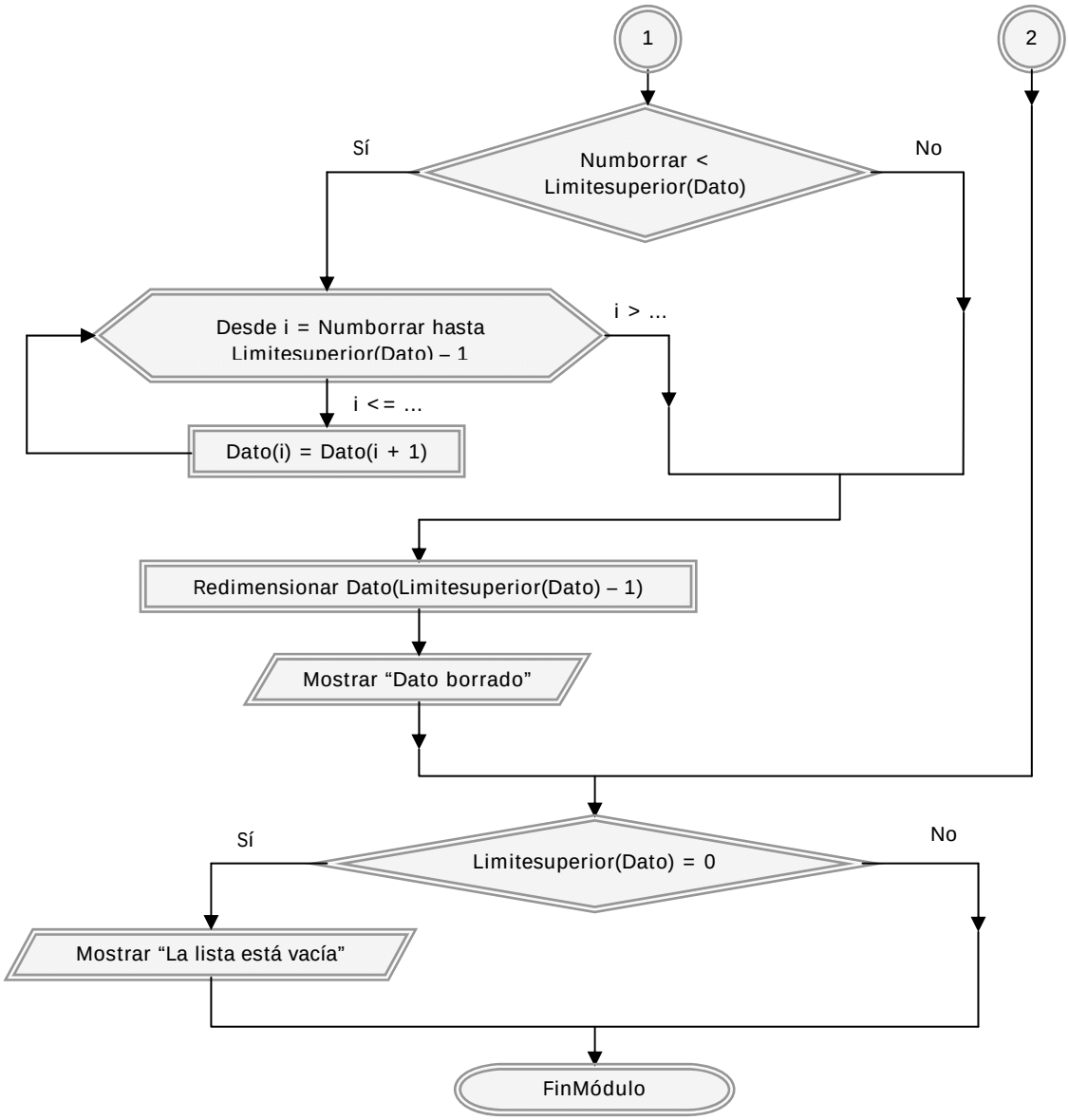


Diagrama de flujo para el módulo Borrar:





Comentarios: El programa consta de un algoritmo principal y tres módulos. Ninguno de los módulos es genérico. ¿Podríamos llamar a un módulo genérico pasando un array como parámetro? Pues sí. Veamos cómo, tomando como ejemplo el módulo *Listar*. Lo escribiríamos así:

```
Módulo Listar(Matriz(): Reales) [Ejemplo aprenderaprogramar.com]

    Variables
        Enteras: i
1.  Desde i = 1 hasta Limitesuperior(Matriz) Hacer
        Mostrar Matriz(i)
    Siguiente
FinMódulo
```

Estaríamos pidiendo el paso de un array sin especificar dimensiones, que en nuestro caso invocaríamos con:

```
Llamar Listar(Dato)
```

Estos procedimientos son más “delicados” y conviene conocer qué condicionantes al respecto nos impone el lenguaje con el que trabajemos. Es recomendable que si se pasa un array como parámetro se haga por variable de modo que se manipulan los propios elementos del array en vez de una copia tipo local. Esto permite una mayor rapidez de los procesos.

Pasar arrays como parámetros puede ser origen de errores por incoherencias entre los datos o incoherencias en el módulo. El array que se pasa debe contener igual tipo de datos que el array que se pide. En caso contrario tendremos un error por no coincidencia de tipos. El módulo no puede llamar a partes del array que no existan en base a un localizador incorrecto, suposición de un número de localizadores erróneo, etc. Ejemplos de incoherencias serían:

a)

```
Llamar Listar(Dato)
        .
        .
        .
Módulo Listar(Matriz(): Enteros)
```

Error: El tipo de Dato no coincide con el tipo de Matriz.

b)

```
Llamar Listar(Dato)
        .
        .
        .
Módulo Listar(Matriz(): Reales)
    1.  Mostrar “La matriz tiene un número de datos igual a”
        Mostrar Limitesuperior(Matriz,1) * Limitesuperior(Matriz,2)
                                .
                                .
                                .
```

Error: No existe un segundo localizador.

En el programa hemos considerado que si no se ha definido ningún elemento del array *Dato* la invocación a *Limitesuperior(Dato)* devuelve cero. Así, si como primera opción antes de existir ningún dato se entrara en el módulo *Borrar,* la situación de *Limitesuperior(Dato) = 0* daría lugar a que apareciera el mensaje: “La lista está vacía” y regresáramos al menú. Si como primera opción entráramos en *Listar* se trataría de entrar en un *Desde i = 1 hasta 0*, con lo cual el bucle no se ejecutaría, según vimos al hablar del *Desde ... Siguiente.* Regresaríamos directamente al menú. Por último, si como primera opción eligiéramos *Conocer* total de datos, se mostraría el mensaje: “El número de datos actualmente en la lista es 0”.

Conviene comprobar cómo trata el lenguaje con el que trabajemos a un array no definido. En un caso como este es posible que sea necesario realizar una declaración como *Redimensionar Dato(0)* para que sea posible realizar la invocación a *Limitesuperior(Dato).*

La obtención de un algoritmo como el contenido en el módulo *Borrar* puede no ser inmediata. Lógicamente interviene la experiencia del programador, pero consideramos recomendable seguir las pautas expuestas en su momento en *"Conocer el problema"*. En este caso, eliminar un elemento de una lista es un problema que intuitivamente conocemos pero cuyo método "paso a paso" no tenemos en la cabeza. No vamos a extendernos puesto que es algo ya tratado en *Conocer el problema / Problemas con resolución intuida,* donde exponíamos el plantear varios casos, tanteo de un método, valoración preliminar, planteamiento de relaciones y procesos con letras, etc.

En este programa hay un redimensionamiento de array por cada valor que se añade o se borra de la lista. Ya hemos dicho que en un proceso repetitivo de asignación de valores a un array, el redimensionamiento continuado puede no ser muy buena estrategia. Sin embargo, en esta ocasión no hay un proceso repetitivo puro, pues depende del usuario el introducir o restar datos a la lista. Al depender del usuario, el proceso es muy lento en comparación con lo que sería la operativa continuada del ordenador, por lo que no hay problema en trabajar así. Además, necesitamos ajustar continuamente el array para poder mostrar datos reales cuando se pida mostrar la lista o conocer el número de elementos. Hacerlo de otra forma será posible, con una doble contabilidad y un sistema de avisos, pero no resulta deseable.

El programa usa reiteradas veces la invocación al límite superior del array, aunque es posible basarse en señaleros y prescindir de *Limitesuperior.*

Próxima entrega: CU00221A

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