VNIVERSIDAD NACIONAL
AVTÓNOMA DE
MEXICO

PERFILES
EDUCATIVOS
ISSN 0185-2698

**Ruiz Velasco Sánchez, Enrique (1989)**

**"ESTILOS DE PROGRAMACIÓN INFORMÁTICA PARA ALUMNOS Y DOCENTES"**

**en Perfiles Educativos, No. 45-46 pp. 77-80.**

CESU Centro de Estudios sobre la Universidad

IRESIE
Banco de Datos sobre Educación

**PERFILES EDUCATIVOS**
Enero • Junio 1989
Núm. 45-46, pp. 77-80

# ESTILOS DE PROGRAMACIÓN INFORMÁTICA PARA ALUMNOS Y DOCENTES

Enrique RUIZ-VELASCO SÁNCHEZ*

Cada vez que el alumno o el docente es invitado a programar, cabe la pregunta de cuál sería el estilo de programación más apropiado a su actividad. En el caso del alumno, la programación, en el contexto de un curso de informática, puede considerarse un método de aprendizaje. En cambio, el docente se plantea el estilo de programación cuando sus alumnos programan y, más específicamente, cuando él desarrollo herramientas informáticas para realizar la investigación y al docencia.

## Estilos de programación

Un estilo de programación se define principalmente por las estructuras de control, las estructuras de datos y el rol de las instrucciones o primitivas disponibles en el medio ambiente de la programación. Estas características son difícilmente disociables.

En el presente estudio abordaremos únicamente las estructuras de control que determinan el flujo de instrucciones que constituyen el texto de un programa; no obstante, es necesario distinguir los estilos llamados secuenciales de los no secuenciales.

## Estilos secuenciales

Una distinción fundamental entre estilos de programación radica en el encadenamiento secuencia o no secuencial de las instrucciones.

Los estilos secuenciales o iterativos predeterminan las acciones que la computadora habrá de realizar cuando ejecuta un programa. Para compararlos con los estilos de expresión del lenguaje natural, se podría hablar de estilos en prosa o narrativos.

Los estilos narrativos son más compatibles con el modo de funcionamiento de los procesadores, por tanto, son convenientes para las aplicaciones rápidas y para desarrollar aquellas que no se prestan a una formalización en términos de funciones, reglas o transmisiones de mensajes. Asimismo, resultan apropiados para instrumentar eficazmente lenguajes y sistema poéticos.

La mayor parte de los lenguajes de programación requiere la formulación de instrucciones de acuerdo con el orden de su ejecución. Como la ejecución varía en función de los datos del problema, es necesario un mecanismo de liga (apuntadores).

* Investigador del CISE.

**Programación por saltos**

En muchos lenguajes de bajo nivel, o bien, en algunos evolucionados cuya concepción estuvo inspirada por el FORTRAN, el mecanismo de orientación es el salto. Para efectuar un salto es necesario poner antes de las instrucciones etiquetas que sirvan para orientar la ejecución; es necesario, igualmente, una primitiva que permita saltar de una instrucción a otra que no la siga directamente. Ésta es la célebre instrucción GOTO que significa “ir a”.

El empleo de saltos ha sido fuertemente criticado por numerosos metodólogos de la programación. Ellos argumentan que el uso de estas estructuras de control conduce a producir programas confusos. Su corrección y mejoramiento constituye una tarea muy laboriosa y delicada. Algunas modificaciones producen consecuencias inesperadas. Los saltos no son visualizados gráficamente en el texto del programa. La figura 1 muestra esquemáticamente un texto de un programa con saltos.

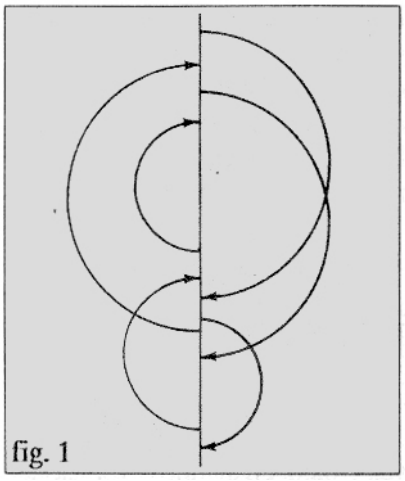
fig. 1

**Programación por bloques**

La programación estructurada por bloques favorece la práctica de una estrategia de resolución de problemas que consiste en subdividir el proyecto que se quiere programar en una serie de subproyectos, y aplicar este proceso de subdivisión tantas veces como sea necesario hasta alcanzar los primitivos del lenguaje. Instrucciones especiales –llamadas de control- permiten la manipulación de los bloques; la figura 2 muestra las estructuras de control de la programación por bloques.

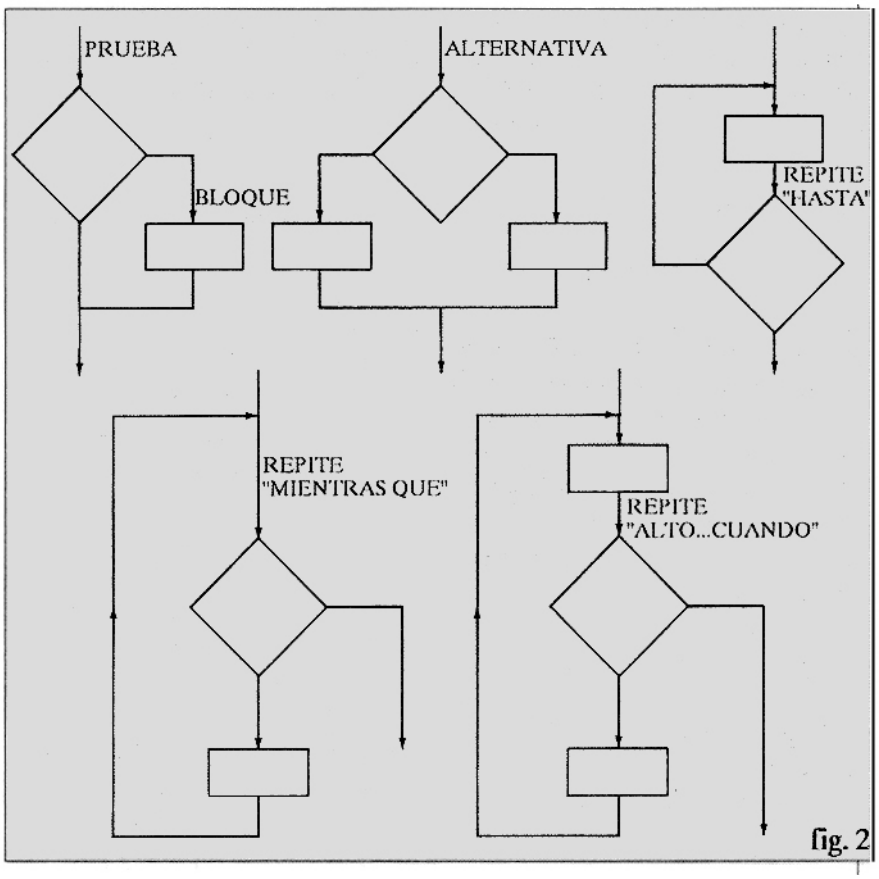

fig. 2

**Programación por llamada**

Para evitar repetir bloques idénticos de instrucciones, se puede llamar simplemente a un bloque por su nombre. Cada bloque es en realidad un subprograma o un procedimiento.

**Estilos no secuenciales**

Ciertos mecanismos de control, explotados sobre todo en el caso de los lenguajes surgidos de la corriente de la Inteligencia Artificial, evitan la programación secuencial que precisa el orden cronológico de las operaciones que la computadora debe ejecutar.

Los llamados estilos aplicativos (de programación funcional) también podrían calificarse de poéticos. Éstos ofrecen una representación intemporal del problema a programar, la cual está referida aun formalismo matemático, lógico o comunicacional. Las acciones precisas de la computadora son más sugeridas que descritas en el texto del programa (de ahí el empleo del término poético). Responden, a su vez, a un programa de base (interpretador o compilador) que deduce una resolución secuencial de un problema, a partir de una representación formalizada del mismo.

Los estilos poéticos son apropiados para resolver problemas complejos, donde toda la atención es puesta en la formalización del problema, independientemente de la máquina. Estos estilos se apegan a formalismos teóricos y abren la vía a una programación desligada del empirismo, puesto que permiten concebir métodos de demostración de la exactitud de los programas.

Actualmente, la mayoría de las aplicaciones están desarrolladas en estilos narrativos; sin embargo los proyectos de la quinta generación de computadoras y el impacto creciente de la Inteligencia Artificial en las aplicaciones comerciales, intensificarán progresivamente la utilización de los sistemas poéticos.

**Programar componiendo funciones**

La programación funcional consiste en utilizar como base de un mecanismo de control, un formulismo matemático. Una función asocia un valor resultante o un valor dado como argumento. Como el resultado de una función puede ser tomado como dato de una función, el modelo del mecanismo de control es la composición de la función fog.

El lenguaje LOGO es un lenguaje funcional, pero muchas de sus funciones primitivas no tienen valor resultante y son, pues, asimilables a procedimientos secuenciales. En otras palabras, el estilo procedural en LOGO no es más que un revestimiento superficial transplantado a un mecanismo funcional. LOGO presenta una alternativa en relación a LISP: mientras que LISP es un lenguaje funcional que puede invocar explícitamente (gracias a una primitiva) el tratamiento secuencial, LOGO es un lenguaje que aparentemente privilegia el tratamiento secuencia, permitiendo invocar explícitamente (gracias a la primitiva REGRESA) el tratamiento funcional. Esto es, el programador en LISP parte de un estilo funcional y “cae” en el estilo secuencial cuando no tiene otra

alternativa, mientras que en LOGO, se parte de un estilo secuencial y se “sube” al estilo funcional, cuando el tratamiento se presta para ello.

El estilo funcional transforma a la computadora en una evaluadora de funciones. Sin embargo, algunas operaciones no pueden ser tomadas en cuenta por el mecanismo funcional. Este es el caso de todas las acciones de control de los periféricos (entrada/salida).

Para solucionar esto, se programan efectos secundarios que permitirán que ciertas funciones sean truncadas y así, se evita que arrojen resultados o argumentos.

Así pues, el medio ambiente gráfico en LOGO es obtenido gracias a un conjunto de efectos secundarios astutamente organizados y transplantados en las primitivas. Por ejemplo, la primitiva AVANZA tiene un argumento, ella no produce ningún resultado y el efecto secundario que ha sido programado consiste en el desplazamiento de un triángulo luminoso, llamado “tortuga”, sobre la pantalla de la computadora.

Ejemplos:

En LOGO, una función completa con un argumento:
PRIMERO.
Una función con dos argumentos, sin resultado y con efecto secundario: REPITE.
Una función sin argumento, con resultado y con efecto secundario:CAP.
Una función sin argumento, sin resultado y con efecto
secundario: SUBELAPIZ.

**Programar enunciando reglas**

El estilo de programación lógica caracteriza a los sistemas de bases de conocimiento y a la programación en lenguaje PROLOG. Este lenguaje fue concebido específicamente fue concebido específicamente para el tratamiento del lenguaje natural. En estilo lógico se programa enunciando reglas lógicas y hechos (o axiomas).

Las reglas:

mortal (x) hombre (x);
mortal (X) perro (x);

significa que x es mortal cuando x es un hombre, o cuando x es un perro.

Si tenemos ahora los hechos incondicionales

hombre (Sócrates) _____;
hombre (Carlos) _____ ;
perro (Maxi) _____;
mueble (librero) _____;

un compilador PROLOG (que “comprende” un formulismo lógico) responderá a la pregunta ¿mortal (x)?

Existe un x, tal que ¿x es mortal? Deduciendo los hechos posibles para el objeto x, es decir:

[x = Sócrates]
[x = Carlos]
[x = Maxi]

Estas deducciones sustituyen la lógica de primer orden, puesto que hacen intervenir variables.

Otro ejemplo:

Pescado (mojarra) _____;
pescado (huachinango) _____;
carne (borrego) _____;
carne (pollo) _____;
platillo (P) ___ pescado (P);
platillo (P) ___ carne (P);

con esta base de datos, la interrogación: ¿platillo (P)? dará los valores posibles para P:

[P = mojarra]
[P = huachinango]
[P = borrego]
[P = pollo]

Se dice que PROLOG funciona en encadenamiento hacia atrás: se trata de probar un hecho hipotético regresando hacia los hechos existentes en la base de conocimiento. El encadenamiento hacia delante consistiría en deducir hechos nuevos, a partir de hechos existentes.

En PROLOG la suma de dos números (x = y + z) se realiza formalmente gracias a una primitiva que deduce x, tal que x vale la suma de y y z. Así tenemos que, cuando y vale 3 y z vale 5,

SUMA ¿(x 35)?
da como resultado
[x = 8]

**Programar enviando mensajes a objetos**

Otro formalismo consiste en definir objetos asociados a mensajes y atributos (o variables) que los caracterizan. La programación en lenguaje SMALLTALK consiste en enviar mensajes a objetos.

Un objeto “responde” a un mensaje ejecutando el método (procedimiento) que corresponde al mensaje.

Los métodos y los mensajes son definidos por una clase de objetos. Cada clase describe un molde. Por ejemplo, los objetos de la clase PLUMA en el sistema SMALLTALK-80 son asimilables a las tortugas LOGO, puesto que ellas comprenden mensajes como:

AVANZA: PASOS
DERECHA: GRADOS

Un objeto de una clase RECTÁNGULO podría responder a mensajes como:

DIBUJATETU
CRECEDE:x
DOBLATUSDIMENSIONES
etcétera.

Las clases pueden jerarquizarse de tal manera que, si A es una subclase de B, todo objeto de la clase A "comprende" los mensajes definidos por la clase A y posee las características de los objetos de esta clase, pero, si hay un sobrecrecimiento, hereda las propiedades (mensajes y variables) definidas por los objetos de la clase B. La transmisión de mensajes a objetos permite un estilo de programación "conceptual", puesto que se trata ante todo de organizar el problema a programar en una jerarquía de conceptos.

Se ha visto que los problemas de aritmética elemental podrían ser programados en un estilo lógico. Pueden también ser adaptados al formalismo de la transmisión de mensajes. Así, efectuar la suma 3 + 5 consiste en enviar a 3 el mensaje + con argumento 5. El número 3, en tanto que representante de la clase de número "comprende" el mensajes + y "responde" enviando el objeto 8 que es un representante de la misma clase. Esta formalización de la aritmética no es habitual, pero es práctica, ya que permite desarrollar programas coherentes utilizando un solo estilo de programación.