aprenderaprogramar.com

Escribir programas o algoritmos en pseudocódigo. Ejemplos resueltos. (CU00135A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel I

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº 34 del Curso Bases de la programación Nivel I

24

CONCATENACIÓN DE ÓRDENES Y ORDENACIÓN SIMBÓLICA DEL PSEUDOCÓDIGO

En algún momento dijimos que un algoritmo se divide en pasos o líneas cuyo contenido y extensión es criterio del autor del algoritmo. Vamos a matizar esta afirmación. Si las instrucciones en un algoritmo se ejecutan de izquierda a derecha y de arriba a abajo, en principio dará igual escribir cuatro instrucciones de izquierda a derecha (en una línea) que de arriba a abajo (en cuatro líneas). La escritura de órdenes una detrás de otra la realizaremos valiéndonos de un elemento de separación, que en nuestro caso serán los dos puntos (:). Así podríamos escribir: A = 3 : B = 2 : C = A * B.

Diferentes órdenes relacionadas a través de dos puntos reciben el nombre de órdenes concatenadas y se ejecutan una detrás de otra, de izquierda a derecha. Como decíamos anteriormente, sería equivalente escribir:

1. Inicio 2. A = 3 : B = 2 : C = A * B : D = C ^2 **3. Fin**	Equivale ⇔	**1. Inicio** 2. A = 3 3. B = 2 4. C = A * B 5. D = C ^2 **6. Fin**

Parece que la concatenación de órdenes redunda en una mejor economía del algoritmo, puesto que se reduce el número de líneas a emplear. Pero ojo: ¿Por qué no escribirlo todo en una sola línea, incluso los indicadores de inicio y fin? La respuesta nos lleva a las formas de percepción y de comprensión humanas. Un libro cualquiera podría ser escrito en una sola línea. Sin embargo, se organiza en párrafos y líneas utilizando efectos visuales como son las sangrías y los márgenes que afectan más a la percepción que al contenido.

En la escritura de pseudocódigo buscaremos claridad y ordenación visual. No es recomendable escribir muchas órdenes en una sola línea. Para ello nos basaremos en sangrías y en delimitación e información de bloques o procesos. Llamaremos “bloque” a un conjunto de órdenes con interdependencia, estrecha relación o agrupadas con un fin. La sangría se hará siempre respecto a una instrucción o comentario que marcan el inicio y fin de la sangría.

```
Inicio sangría
──────►            Instrucciones con sangría
Fin sangría
```

Ejemplos:

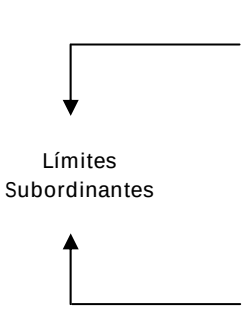
```
[Valor de los parámetros]
                        A = 7
                        B = 16
                        C = 3
[Fin de asignación de valor a parámetros]
```

♦ ♦

```
[Cálculo de superficies]
        S1 = 3 * A
        S2 = 4 * B
        S3 = 2 * C
[Fin de cálculo de superficies]
```

Las sangrías se pueden anidar cuantas veces se quiera.

Ejemplo:




```
[Cálculo de superficies]
    Bloque Subordinado
        [Cálculo de S1]
            S11 = 3 * H
            S12 = 4 * M
            S1 = S11 + S12
        [Fin de cálculo de S1]
        S2 = 4 * B
        S3 = 2 * C
[Fin de cálculo de superficies]
```

No existe norma que diga cuantas sangrías se deben introducir. El exceso o defecto pueden ir en contra de la lectura del programa, y ha de ser el programador el que siguiendo una lógica tal como si estuviera escribiendo una novela, defina su estilo para conseguir la máxima claridad. La subordinación se puede originar a partir de comentarios o a partir de órdenes con principio y fin.

Supongamos que una instrucción asigna a la variable SUMA el resultado de sumar una serie de variables. Escribiríamos:

```
Suma de variables (SUMA)
        A
        C
        D
        M
Fin de suma de variables
```

El inicio y fin de la instrucción funcionarían como límites subordinantes mientras que la lista de variables sería el bloque subordinado. Igualmente aceptable sería el no haber utilizado sangría. Sin embargo, es preferible usarla para mayor claridad.

En cuanto a la delimitación e información de bloques y procesos, se trata de buscar que la presentación del programa sea tal que permita buscar e identificar con rapidez las distintas partes del mismo. Para ello nos apoyamos en la introducción de comentarios delimitadores y en sangrías. Veamos con un ejemplo muy “gráfico” lo que sería el mismo pseudocódigo con cuatro formas de presentarlo.

VERSIÓN 1

1. **Inicio**
2. T = 32 : TT = 11 : CT = 40 : CTT = 65 : NC = T * CT + TT * CTT
3. **Fin**

♦♦

VERSIÓN 2

1. **Inicio**
2. T = 32
3. TT = 11
4. CT = 40
5. CTT = 65
6. NC = T * CT + TT * CTT
7. **Fin**

♦♦

VERSIÓN 3

1. **Inicio**
2. [Definición de vehículos aprenderaprogramar.com]
3. Turismos = 32
4. Todoterrenos = 11
5. [Fin de definición de vehículos]
6. [Definición de capacidad de depósitos]
7. Capturismos = 40
8. Captodot = 65
9. [Fin de definición de capacidad de depósitos]
10. [Cálculo de necesidades de combustible]
11. Necesidadescom = Turismos * Capturismos + Todoterrenos * Captodot
12. [Fin de cálculo de necesidades de combustible]
13. **Fin**

♦♦♦

VERSIÓN 4

```
1. Inicio
    2. [Definición de vehículos]
        3. Turismos = 32
        4. Todoterrenos = 11
    5. [Fin de definición de vehículos]
    6. [Definición de capacidad de depósitos]
        7. Capturismos = 40
        8. Captodot = 65
    9. [Fin de definición de capacidad de depósitos]
    10. [Cálculo de necesidades de combustible]
        11. Necesidadescom = Turismos * Capturismos + Todoterrenos * Captodot
    12. [Fin de cálculo de necesidades de combustible]
13. Fin
```

Comentaremos una a una las diferentes versiones del algoritmo.

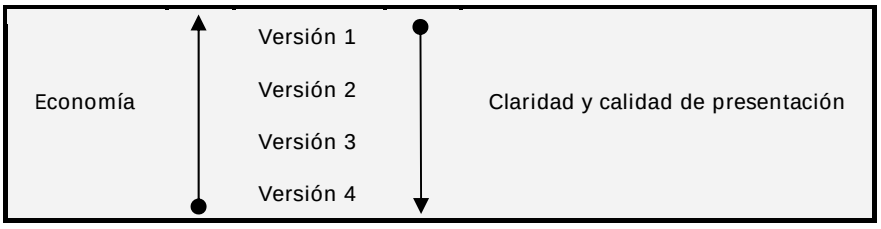
La versión 1 es la menos extensa al reunir todo el proceso en una línea. Sin embargo, es difícilmente interpretable pues no contiene información a modo de comentarios. Tampoco se aprecia delimitación de procesos.

La versión 2 permite identificar mejor los distintos pasos, aunque sigue siendo difícilmente interpretable.

La versión 3 es de mayor longitud pero aporta información que hace interpretable el algoritmo, quedando además delimitados los distintos procesos.

La versión 4 no varía en longitud respecto a la tercera, pero mejora la calidad de presentación a través de sangrías.

Esquemáticamente tendremos:



Buscaremos la máxima claridad y calidad de presentación, pero sin exageraciones. La identificación de procesos, variables, etc. será la justa y necesaria.

Próxima entrega: CU00136A

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=28&Itemid=59