aprenderaprogramar.com

Uso de lenguajes de programación para verificación (prueba) de algoritmos. (CU00232A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº 31 del Curso Bases de la programación Nivel II

24

VERIFICACIÓN (PRUEBA) DE ALGORITMOS MEDIANTE SEGUIMIENTO CON DESARROLLO EN UN LENGUAJE (PROGRAMACIÓN RÁPIDA)

La verificación de algoritmos con un lenguaje de programación consiste en convertir el pseudocódigo en código, y a través de un ordenador o dispositivo programable, verificar que el algoritmo funciona.

Hay que destacar lo siguiente:

- Se trata de una verificación centrada en el funcionamiento. Los aspectos relativos a una adecuada presentación o a cuestiones que no afectan al funcionamiento como los nombres de variables, tabulaciones, uso de instrucciones de control directo de flujos, etc. pasan a un segundo plano siendo el objetivo la prueba rápida. Se admite por tanto, un cierto descuido "formal".
- Las pruebas normalmente no se guardan. Son de no excesiva longitud y contenidos no cuidados, por lo que no son usadas para la construcción final del programa, en la que sí conviene ir "con pies de plomo".
- Las pruebas se realizan en cualquier lenguaje y en cualquier dispositivo programable. Dado que tratamos de probar el algoritmo, no necesariamente hemos de usar el mismo lenguaje en que se va a desarrollar el programa. Por otro lado tampoco hemos de estar delante de un ordenador tal y como lo entendemos habitualmente (monitor, torre, teclado, ratón, etc.) sino que podemos usar cualquier dispositivo, llámese ordenador, portátil, agenda, calculadora o teléfono móvil. El requisito obviamente es disponer de un lenguaje y procesador que permitan ciertos desarrollos.
- Se suele recurrir a una programación rápida del algoritmo cuando se estima que una verificación mental, escrita o con tabla de variables no es efectiva o requiere demasiado tiempo. No es pues una herramienta a la que debamos estar recurriendo continuamente pues el trasvase pseudocódigo → código, aunque sea simplificado, requiere un tiempo.

Esta técnica destaca por su potencia y buenos resultados, por lo que consideramos positivo recurrir a ella siempre que se sigan las pautas indicadas.

Para el ejemplo que venimos trabajando, el código a escribir podría ser algo así:

```
PROGRAM Prueba;

VAR
    i, j, A : Integer

BEGIN
    FOR i : = 1 TO 5 DO
        BEGIN
            FOR j : = 1 TO 5 DO
                A : = i * j
                Write Ln(A)
        END ;
END.
```

La verificación a través de programación rápida en un lenguaje tiene la ventaja de que permite centrarnos en el análisis de resultados o en el de procesos, o en ambos. Lo veremos a continuación cuando hablemos de verificación de algoritmos genéricos. Si tras ejecutar el programa de prueba este no realiza lo esperado se procederá a su revisión, empezando por comprobar que no hay errores de escritura (sintaxis inadecuada para el lenguaje empleado).

VERIFICACIÓN DE ALGORITMOS MEDIANTE SEGUIMIENTO CON UN LENGUAJE PASO A PASO

Para algoritmos de cierta complejidad en los que utilicemos programación rápida, nos puede interesar el uso de herramientas de corrección o depuración de las que dispone el propio lenguaje. Las posibilidades son variadas en función del lenguaje que empleemos, por lo que conviene estudiar a través de la ayuda o de un manual del lenguaje qué recursos hay disponibles. Entre los más habituales están los relacionados con una ejecución del programa paso a paso que permite conocer el flujo del programa y los valores de las variables en cada momento. A través del "paso a paso" podemos ralentizar y analizar procesos que de otra forma ocurrirían muy rápido o generarían demasiada información como para poder detectar dónde está localizado un problema. Recurrir a esta herramienta no será habitual, ya que en muchos casos se puede seguir el algoritmo con una ejecución normal, o introduciendo un par de instrucciones de seguimiento dentro de una ejecución normal. Su uso principal viene dado por circunstancias específicas como algoritmos de muy difícil seguimiento o por errores de difícil detección.

Supongamos que tenemos una agenda portátil programable que nos permite una ejecución paso a paso referenciada a líneas. Si la activamos llamando al programa *Ej* y numerando las líneas de 1 a 6, los resultados serían:

```
[Ejemplo aprenderaprogramar.com]
1. Desde i = 1 hasta 5 Hacer
        2. Desde j = 1 hasta 5 Hacer
                3. A = i * j
                4. Mostrar A
        5. Siguiente j
6. Siguiente i
```

Ejecución normal: 1 2 3 4 5 2 4 6 8 10 3 6 9 12 15 4 8 12 16 20 5 10 15 20 25

Ejecución paso a paso: START Ej-1, Ej-2, Ej-3, Ej-4 "1", Ej-5, Ej-3, Ej-4 "2", Ej-5, Ej-3, Ej-4 "3", Ej-5, Ej-3, Ej-4 "4", Ej-5, Ej-3, Ej-4 "5", Ej-5, Ej-6, Ej-2, Ej-3, Ej-4 "2", Ej-5, Ej-3, Ej-4 "4", Ej-5, Ej-3, Ej-4 "6", Ej-5, Ej-3, Ej-4 "8", Ej-5, Ej-3, Ej-4 "10", Ej-5, Ej-6, Ej-2, Ej-3, Ej-4 "3", Ej-5, Ej-3, Ej-4 "6", Ej-5, Ej-3, Ej-4 "9", Ej-5, Ej-3, Ej-4 "12", Ej-5, Ej-3, Ej-4 "15", Ej-5, Ej-6, Ej-2, Ej-3, Ej-4 "4", Ej-5, Ej-3, Ej-4 "8", Ej-5, Ej-3, Ej-4 "12", Ej-5, Ej-3, Ej-4 "16", Ej-5, Ej-3, Ej-4 "20", Ej-5, Ej-6, Ej-2, Ej-3, Ej-4 "5", Ej-5, Ej-3, Ej-4 "10", Ej-5, Ej-3, Ej-4 "15", Ej-5, Ej-3, Ej-4 "20", Ej-5, Ej-3, Ej-4 "25", Ej-5, Ej-6, END.

La ejecución de todos los pasos normalmente será opcional, ya que procesos tipo bucle pueden hacerse muy tediosos. Así pues, podremos establecer la ejecución de un bucle en forma normal y después una serie de instrucciones que irán paso a paso.

Supongamos ahora que tenemos un problema ya que al intentar ejecutar el algoritmo nos aparece un mensaje “sintaxis error” sin conocer dónde se origina. El pseudocódigo ha sido este:

```
1. Desde i = 1 hasta 5 Hacer
        2. Desde j = 1 hasta 5 Hacer
                3. i * j = A
                4. Mostrar A
        5. Siguiente j
6. Siguiente i
```

La ejecución paso a paso, siguiendo el ejemplo anterior, daría:

START, Ej-1, Ej-2, Ej-3, Sintaxis error Ej-3

El seguimiento realizado nos permite identificar que el error se produce al llegar a la línea 3. Al revisarla nos percatamos de que ha habido un error de escritura. Reemplazamos el incorrecto *i * j = A* por *A = i * j*, quedando resuelto el problema.

Próxima entrega: CU00233A

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