aprenderaprogramar.com

Depuración de programas frente a errores. Permisividad o intransigencia (Programación defensiva). (CU00243A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

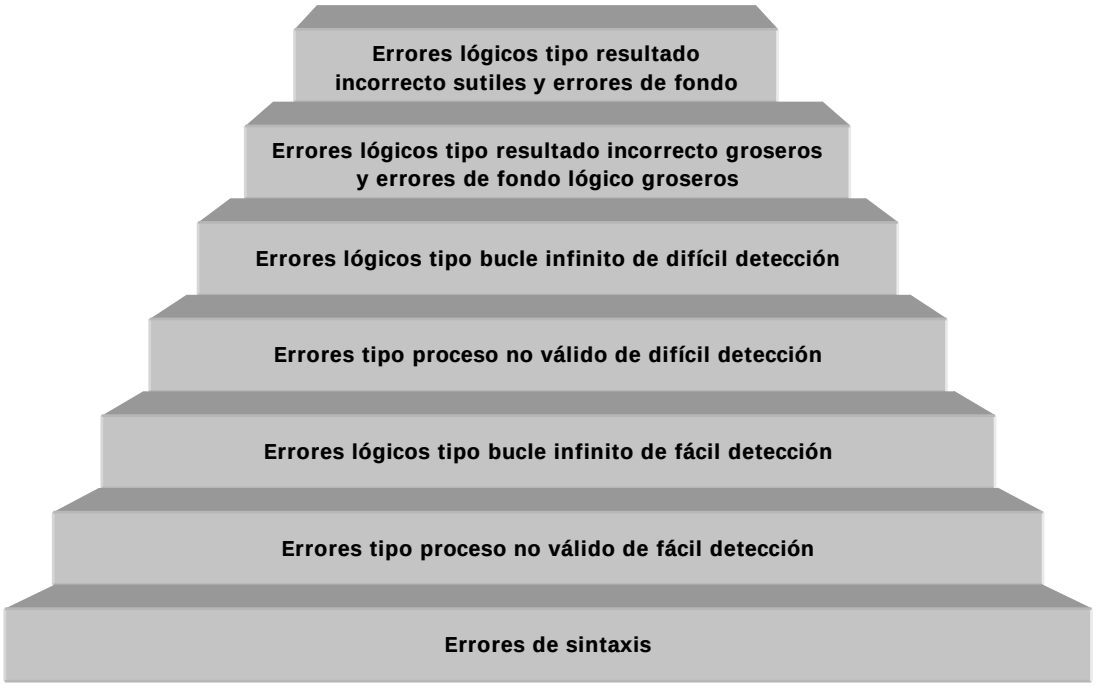
Fecha revisión: 2024

Autor: Mario R. Rancel

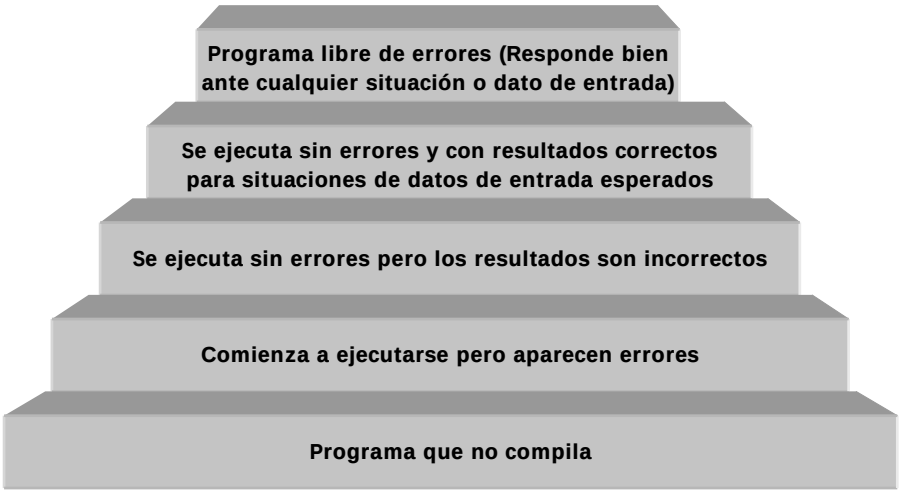
24

DEPURACIÓN DE PROGRAMAS. PERMISIVIDAD O INTRANSIGENCIA (PROGRAMACIÓN DEFENSIVA)

Cuando se ha desarrollado el pseudocódigo de un algoritmo o programa y lo introducimos en el ordenador en forma de código, se intenta la primera ejecución. Con esta primera ejecución se empieza a obtener información respecto al estado del programa en cuanto a errores. Es habitual que el programa no funcione a la primera. Comenzamos un proceso de eliminación de errores que supone ir construyendo un programa sólido a partir de un embrión de programa débil. Vamos a verlo como un proceso de destrucción de una pirámide de errores y construcción de una pirámide de programa sólido.



Pirámide de errores



Pirámide de construcción del programa

La pirámide de errores se desmonta socavando su base, mientras que la pirámide de construcción del programa se empieza a construir por su base. Así, el primer paso de construcción del programa es pasar de un código que no compila a un código que sí compile. Para ello tendremos que depurar los errores de sintaxis y quizás algunos groseros de procesos no válidos o lógicos. El siguiente paso es el de alcanzar un programa que se ejecute sin errores, superando el escalón del programa que no termina de ejecutarse. Para ello necesitamos eliminar los errores de proceso o lógicos de difícil detección. El proceso continúa aunque no necesariamente habrán de alcanzarse la cúspide de las pirámides. El programador habrá de decidir si busca una depuración total (la depuración total es muy difícil de conseguir) y un programa muy sólido o un programa que funcione con algunas limitaciones.

Una forma de ver la pirámide de errores y la pirámide del programa tiene que ver con la probabilidad de que ocurra un error. Cuando un error se ha de producir necesariamente, por estar contenido en el código, decimos que es de obligada ocurrencia. Si un error tiene cierta posibilidad de aparecer en función de las circunstancias decimos que tiene cierta probabilidad de ocurrencia. Un error de sintaxis es de obligada ocurrencia por estar contenido en el código. En cambio, un error por proceso no válido debido a una entrada del usuario incorrecta puede o no puede aparecer y tiene cierta probabilidad de ocurrencia. Muchos programadores dicen que la depuración del programa realmente comienza cuando se han eliminado los errores de obligada ocurrencia, que vienen a ser la base de la pirámide: los errores de sintaxis, y los errores por procesos no válidos y tipo bucle infinito contenidos en el código (de fácil detección). Esto constituye pues una base de partida para enfrentarse a los errores realmente complejos. Llegar a este punto ha de ser fácil si partimos de un correcto diseño de algoritmos y un cierto nivel de conocimiento del lenguaje, y viene a significar situarnos en un nivel de programa que compila y se ejecuta sin errores.

El umbral que hace al programa válido para su uso es el de ejecución sin errores y con resultados correctos para situaciones y datos de entrada esperados. Por tanto este será un objetivo irrenunciable para el programador. Nos situamos en un nivel en el que no existen errores de obligada ocurrencia pero sí cierta probabilidad de aparición de errores ante circunstancias específicas. Hablando pues en términos de probabilidad no nos referimos a número de errores, sino al grado de vulnerabilidad o, si se prefiere, grado de fortaleza de un programa. Si la posibilidad de errores es alta decimos que el programa es débil o vulnerable y si es baja que es poco vulnerable o fuerte. Las equivalencias serían estas:

Probabilidad de error	Grado de vulnerabilidad	Grado de fortaleza
Muy alta	Muy vulnerable	Inconsistente, frágil
Alta	Vulnerable	Débil
Baja	Poco vulnerable	Fuerte, sólido
Nula	Invulnerable	Muy fuerte, totalmente sólido

Si partimos de que hemos de alcanzar el programa válido nos situamos en un programa con probabilidad de errores baja o alta, pero no muy alta ni nula. A partir de aquí, el programador ha de decidir si una vez alcanzado ese nivel continúa depurando para hacer al programa más fuerte de lo que es, o bien si establece una gestión de errores con el fin de que en caso de darse circunstancias no previstas se puedan obtener resultados parciales o hacer una salida controlada. Cuanta más depuración más fuerte es el programa, pero también mayor esfuerzo y tiempo requiere. El programador puede adoptar pues una actitud de lucha tenaz y búsqueda de cualquier posible error para su eliminación, a lo que llamamos enfoque de intransigencia, o bien dar por bueno que se produzcan ciertos errores para supuestos poco usuales que serían gestionados, a lo que llamamos enfoque de permisividad. Si un programador se obstina en mantener una total intransigencia frente a los errores, necesitará más tiempo y esfuerzo para construir sus programas, que no necesariamente van a ser más efectivos. Crear algoritmos en previsión de situaciones que muy extrañamente pueden ocurrir puede tener más costos que beneficios.

En resumen: la postura más lógica es la de tener cierta permisividad hacia los errores, buscando el equilibrio entre esfuerzo y grado de perfección del programa.

Próxima entrega: CU00244A

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