Legibilidad del código fuente: facilitando el mantenimiento de las aplicaciones

Coloquios Abiertos

J. Baltasar García Perez-Schofield
Escuela Superior de Ingeniería Informática, Edif. Politécnico, s/n, 32004 Ourense
e.mail jbgarcia@uvigo.es

Abstract. *Una parte importante del éxito de un software consiste en que sea fácilmente mantenible. Está demostrado que, un programa que es utilizado precisa siempre de mantenimiento, y si esta aplicación soporta con cierta facilidad la extensión de su funcionalidad, el éxito desde el punto de vista económico está asegurado. Existen varios factores que inciden directamente sobre la capacidad de un software de ser mantenido, pero uno de ellos, quizás el importante, es que este software pueda ser leído fácilmente por otros programadores.*

Legibilidad

Se ha demostrado que el ser humano no lee una a una las letras de una palabra cuando está intentando leer un artículo en un periódico, un libro ... etc. Así, lo que realmente hace el cerebro humano es un reconocimiento de formas, de manera que una palabra de tres letras, con la primera alargada hacia abajo, la segunda redonda y la tercera como medio cuadrado será interpretada como la preposición "por". Así, el reconocimiento de formas, y colores, hacen más sencilla y agradable la lectura.

Concretamente, en el ámbito de la programación, ésto explica el desarrollo de editores con coloreado de sintaxis y otros formateadores de texto. De ahí se deduce que escribir código con una indentación adecuada (es decir, con una determinada "forma", en el sentido gráfico), facilita su comprensión, al quedar claros los anidamientos de bucles, niveles ... etc.

En la exposición se proponen diferentes técnicas, que, asociadas a la indentación, pueden ayudar a escribir código más legible. En concreto, se enumeran técnicas de escritura de comentarios, técnicas de formato de definición de variables, técnicas de división del código en líneas, y finalmente, escritura correcta de condiciones y bucles.

La última parte de la exposición, como ejemplo de lo que no se debe hacer, se dedica al concurso de C-ofuscado, que trata precisamente de programas que parten de códigos fuente ilegibles.

Bibliografía

- Eckel (2000). *Thinking in C++.* Prentice Hall
- Caro, Ramos, Barceló (2002). *Introducción a la programación con orientación a objetos.* Prentice-Hall
- Documentación de Microsoft: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/stg/stg/coding_style_conventions.asp
- Documentación de Sun: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html