Guía de estilo en Programación

(en C)

CC31A

Johan Fabry (jfabry@dcc).
Inspirado en parte en documentos de Wolfgang De Meuter y Chris Dutchyn.

Una parte muy importante en la vida de un computín es trabajar con código ya hecho: piense en mantenimiento y evolución del código existente. Por eso es sumamente importante que ustedes escriban código que se puede entender algunos meses, años o décadas en el futuro. Queremos enseñarles ya en este curso las cosas básicas sobre escribir código 'lindo'. Esa guía esta hecha para tener menos de una página de reglas, así no se olvidan ninguna regla.

El código entregado para las tareas de CC31A imperativamente tiene que seguir estas reglas. No respetar estas reglas implica bajar sus notas.

I Nombres de cosas:

Para entender el código, el lector se deja guiar por los nombres de las cosas: nombres de variables, nombre de funciones, nombre de tipos.

1. El nombre de una cosa tiene que dar una idea de que se trata
2. Nombres tienen que ser usados de manera consistente (ej: no en ingles y en español)
3. No se ponen números mágicos dentro del código. Usen un #define para darle un nombre y usen este. Así también el numero se pueda cambiar fácilmente después.

II Funciones

1. Funciones deben tener un tamaño chico. En general si una función es de mas de una pantalla, hace demasiadas cosas y no se puede entender.
2. Use tabulación para indicar el flujo de control en sus funciones

III Comentarios:

1. Cada función necesita un comentario que dice su intención: expone lo que debería hacer la función. Tiene que mencionar cada argumento y el valor que resulta.
2. Cada definición de union y struct necesita un comentario que dice lo que representa este nuevo tipo, que significan sus partes y aclara valores especiales o específicos para estas partes (si hay).
3. Dentro de cada función si hay una parte algoritmicamente compleja un comentario debe aclarar lo que hace.

IV .c versus .h

Archivos **.h** sirven para declarar, no para definir!

1. Tipos de funciones, declaración de structs y enums y typedefs en un **.h**
2. #defines en un **.h**
3. Implementación de funciones en un **.c**