Principios de la Buena Programación
Programación Avanzada

Prof. Miguel A. Pizaña

17 de Septiembre de 2007

Lista basada en: Kernighan y Plauger
The Elements of Programming Style
Computing McGraw-Hill, 2ª edición, (1988)

Objetivos: • Robustez • Claridad • Extensibilidad • Reusabilidad • Eficiencia

1. Principios Generales
   - *a*) Es mejor ser claro que astuto.
   - *b*) Parentizar para evitar ambigüedades.
   - *c*) Escoger nombres de variables y funciones claros.
   - *d*) Usar cada variable para un solo propósito.
   - *e*) Inicializar variables tan cerca como sea posible de su definición y/o uso.
   - *f*) Reemplazar código repetitivo con llamadas a funciones.
   - *g*) Usar funciones de la biblioteca standard.
   - *h*) Utilizar identación para separar bloques de código lineal conceptualmente distintos.
   - *i*) Hacer que el programa se lea de arriba a abajo, por ejemplo: `main()` siempre va primero.
   - *j*) Reemplazar código malo (no parcharlo).
   - *k*) Escribir y poner a prueba el programa de manera incremental.
   - *l*) Evitar el uso de "números mágicos".
2. Estructuras de control
   - *a*) Usar identación en las estructuras de control.
   - *b*) Usar `else-if` en lugar de `then-if`.
   - *c*) Evitar `else`'s vacíos.
   - *d*) Usar `switch` siempre que sea posible.
   - *e*) Evitar `goto`.

3. Modularización

   a) Usar módulos y subrutinas.

   b) Definir claramente las atribuciones de los módulos.

   c) Dejar que los módulos escondan cosas.

   d) Usar archivos cabecera.

   e) Evitar el uso de variables globales en los módulos cuando sea posible.

4. Entrada/Salida

   a) Checar la validez de las entradas.

   b) Verificar que los rangos de las variables no violen las capacidades del programa.

   c) Identificar entradas inválidas y recuperarse.

   d) Verificar que el fin de archivo se encuentre cuando se espera.

   e) Autodefinir la entrada, usar valores por omisión.

   f) Aislar rutinas de entrada/salida en módulos separados:
   nunca usar `getch()` ni `scanf()` directamente.

   g) Jamás usar `gets()`.

5. Errores Comunes

   a) Verificar el programa en los valores frontera.

   b) Verificar errores de ± 1.

   c) Programar defensivamente.

   d) Escribir código de recuperación para apuntadores nulos.

   e) Punto flotante: la igualdad.

6. Documentación

   a) No muchos ni pocos comentarios.

   b) Que el código y los comentarios coincidan.

   c) Comentar funciones y módulos.

   d) Documentar las estructuras de datos.

   e) Comentar los parámetros de las funciones cuando sea necesario.

7. Eficiencia

   a) Primero correcto, luego rápido.

   b) Primero claro, luego rápido.

   c) Mantenerlo correcto cuando se optimiza.

   d) El código simple es en general más rápido.

   e) No sacrificar claridad por una pequeña ganancia en eficiencia.