Tips de estilo para la programación

Gabriel Saldaña

1. Rutinas (funciones)

La razón más importante de crear una rutina es para mejorar la manejabilidad intelectual del programa. Se pueden crear rutinas por muchas razones, pero las mejores son: mejorar la legibilidad del código, mejorar el mantenimiento, mejorar la consistencia.

Las rutinas deben hacer una sola cosa, y eso debe estar descrito y entendido desde su nombre.

El nombre de una rutina indica la calidad de su construcción. Si el nombre es malo pero preciso, la rutina esta mal diseñada. Si el nombre es malo e impreciso, no esta diciendo que hace el programa. De cualquier forma, un mal nombre significa que el programa debe ser modificado.

Los nombres de las rutinas deben empezar con un verbo.

Las rutinas de un programa deben mantener un orden común en los parámetros.

Los parámetros en una rutina no deben ser mayores de 7.

Las rutinas deben siempre de devolver un sólo valor.

1.1. Checklist

- el nombre de la rutina empieza con un verbo?
- El nombre de la rutina es claro y conciso?
- El nombre de la rutina describe todo lo que hace esa rutina?
- los parametros de la rutina estan en orden logico?
- la rutina tiene menos de 7 parametros?
- se estan usando todos los parametros de entrada la rutina?
- se estan usando todos los parametros de salida la rutina?
- la rutina regresa siempre un valor en todos los casos?

2. Datos en general

Es común encontrar errores en la inicialización de datos.
Minimiza el ámbito de cada variable. Evita las variables globales.
Mantén las instrucciones que usan una misma variable lo mas cerca posible.
Usa una variable para un sólo propósito.

2.1. Checklist

- Cada rutina checa la validez de los datos de entrada?
- Se estan declarando las variables cerca de donde se usan?
- Se estan inicializando las variables al momento de declararlas si es posible?
- Los contadores e indices se inician adecuadamente y reinicializados al terminar?
- el ambito de las variables es corto?
- se estan usando todas las variables declaradas?
- se esta usando una variable para un solo proposito?

3. Variables

Buenos nombres de variables son elementales para la legibilidad de un programa.

Los nombres deben ser tan específicos como sea posible. Nombres vagos o generales que pueden usarse para más de un propósito, usualmente son malos nombres.

Convenciones de nombramiento ayudan a distinguir entre datos locales, de clase o globales. Distinguen entre los nombres de tipo, constantes, enumeraciones y variables.

Independientemente del proyecto en el que se esté trabajando, se debe buscar apegarse a un estandard de nombramiento de variables.

El código es leído mas veces de las que es escrito. Asegúrate que los nombres que elijas ayuden a la lectura del código por encima de la rapidez de escritura.

3.1. Checklist

- El nombre de la variable describe su proposito y uso?
- Los indices en un ciclo tienen nombres claros?
- las variables de valor booleano se puede reconocer por el nombre?

4. Condicionales

Para los casos de if-else simples, pon atención en el órden de los enunciados. El caso más común debe ir primero.

Para cachar errores, usa la cláusula "default.[en] un switch, o el último else en una cadena de if-else.

4.1. Checklist

- Se puede entender el paso del programa claramente?
- Se esta usando el caso más común al principio?
- el caso mas comun esta en el if y no en el else?
- se estan cubriendo todos los casos posibles?
- en el caso de switch, los casos estan ordenados en base a su uso?
- la clausula default esta siendo usada para cachar errores?

5. Ciclos

Los ciclos pueden ser complicados. Mantenerlos simples ayudan a la legibilidad del código.

Para mantener los ciclos simples algunas técnicas son: minimizar los loops anidados, mantener claro cuando se entra y se sale de un ciclo.

Los índices en un ciclo se puede abusar para muchos usos. Nómbralos correctamente y úsalos para un sólo propósito.

Revisa bien un ciclo para verificar que opera normalmente bajo todos los casos y termina correctamente en todas las condiciones posibles.

Mantén las expresiones booleanas simples y fáciles de leer para mejorar la calidad en tu código.

5.1. Checklist

- Se esta usando un while en lugar de un for adecuadamente?
- El codigo de inicializacion esta directamente arriba del ciclo?
- el ciclo es corto y entendible?
- El ciclo tiene gran contenido que podria estar mejor en rutinas separadas?
- el ciclo realiza solamente una tarea bien definida?
- el ciclo termina siempre en todos los casos?
- es clara la condicion de salida de un ciclo?

6. Estrategias de optimización

El rendimiento es sólo uno de los aspectos en la calidad de un programa. Código optimizado es sólo un aspecto del rendimiento. La arquitectura del programa, diseño detallado, estructuras de datos y la selección de algoritmos son los que más peso tienen en la rapidez y tamaño de ejecución de un programa.

Se debe poder medir cuantitativamente las áreas de un programa para poder maximizar su eficiencia. Sin mediciones, podemos hacer cambios que no sólo no afecten en nada, sino al contrario, degradar el rendimiento del programa sin darnos cuenta.

La mayoría de los programas pasan mucho tiempo en una pequeña parte del código. Si no se mide, no podremos determinar que parte es.

La mejor manera de prepararse para optimizar el código en un programa es escribiendo código fácil de entender y de modificar.

6.1. Checklist

- usa enteros en lugar de flotantes
- inicializa datos con anticipacion
- deja de evaluar cuando ya encontraste la respuesta
- ordena los cases y los if-else en orden de frecuencia
- minimiza el trabajo realizado dentro de un ciclo
- pon el ciclo mas activo en la parte de adentro de un loop anidado
- realiza cache de datos usados frecuentemente