Checklist de las prácticas de construcción utilizadas

Inicializar cada variable cuando se declara.

Idealmente, declarar y definir cada variable cerca de donde se usa.

Inicializar los datos de los miembros de una clase en su constructor.

Manejo del “Scope”

Uso de calificadores de cálculo en nombres de variables

Usar opuestos comunes en nombre de variables

Nombramiento de índice de bucle

Permite asignar nombres más largos a las variables de bucles

Usar cada variable para un solo propósito

Problema-Orientación(Nombres mnemotécnicos)

Nombramiento de variables de estado

Nombramiento de variables temporales

Nombramiento de variables booleanas

Nombramiento de constantes

Diferenciar entre nombres de variables y nombres de rutinas

Identificar constantes nombradas

Identificar parámetros de solo entrada (input-only) en lenguajes que no los aplican

Dar formato a los nombres para mejorar la legibilidad

Documentar nombres extremadamente cortos con tablas de traducción en el código

Anticipar errores de división por cero

Hacer que las conversiones de tipo sean obvias

Revisar divisiones enteras

Revisar desbordamientos de enteros

Evitar sumas y restas en números que tengan magnitudes muy distintas.

Lenguajes y bibliotecas de soporte para número grandes

Revisar el lenguaje y bibliotecas de soporte para tipos de datos específicos

Errores off-by-one de cadenas

Código Unicode

Internacionalización

Evitar cadenas y caracteres mágicos

Usar constantes nombradas en declaraciones de datos

Evitar las literales o números, incluso las "seguras"

Asegurar de que todos los índices del arreglo estén dentro de los límites del arreglo

Crear tipos con nombres orientados funcionalmente

Definir tipos sustitutos para la portabilidad (INT- int)

Considerar la posibilidad de crear una clase en lugar de utilizar un typedef

Evitar comparaciones de igualdad

Documentar al usar variables boolenas

Sentencias secuenciales más fáciles de ordenar son aquellas en las que el orden cuenta

Organizar el código para que las dependencias sean obvias.

Nombrar las rutinas para que las dependencias sean obvias.

Usar parámetros de rutina para hacer que las dependencias sean obvias.

Escribir primero la ruta nominal a través del código; luego escribe los casos inusuales

Asegurarse de bifurcar correctamente en igualdad. Usar “>=” en lugar de “>” o “<=” en lugar de “<”

Poner el caso normal después del if, en lugar de después del else.

Seguir la cláusula if con una declaración significativa.

Poner los casos más comunes primero

Ordenar los casos por orden alfabético

Poner el caso normal primero

Ordenar caso por frecuencia.

Usar parámetros con nombres significativos

Poner las variables de estado o error al final

No utilizar parámetros de rutina como variables de trabajo

Para nombrar una función, usar una descripción del valor devuelto

Para nombrar un procedimiento, utilizar un verbo fuerte seguido de un objeto

Poner los contadores que controlan los ciclos del bucle, al principio o al final del bucle.

Cuándo usar un ciclo while con la condición al principio

Cuándo usar un ciclo while con la condición al final

Cuándo usar un ciclo loop-with-exit

Cohesión Funcional

Cohesión Secuencial

Cohesión Comunicativa

Cohesión Temporal

Cohesión Procedural

Cohesión lógica

Cohesión Coincidente

Preferir la herencia a la verificación extensiva de tipos.

Si varias clases comparten datos comunes, pero no comportamiento, crear un objeto común que esas clases puedan contener

Si varias clases comparten un comportamiento común, pero no datos, derívalas de una clase base común que defina las rutinas comunes

Si varias clases comparten datos y comportamientos comunes, heredar de una clase base común que define las rutinas y los datos comunes

Heredar cuando desea que la clase base controle la interfase; contener cuando desea controlar su interfase

Hacer cumplir la propiedad singleton mediante el uso de un constructor privado

Comentarios