Checklist

Programación de Computadoras

Creación de Datos

- ☐ ¿El programa valida las entradas?
- ☐ ¿El código inicializa las variables cerca de donde se usan?
- ☐ ¿Los contadores y acumuladores son inicializados apropiadamente y, si es necesario, reinicializados cada vez que son usados?
- ☐ ¿El código compila sin *warnings*?

Nomenclatura de Datos

Nombres

- ☐ ¿El nombre describe completa y exactamente lo que la variable, constante, tipo de dato o archivo representa?

Convenciones

- ☐ ¿La convención de nomenclatura distingue entre constantes con nombre y variables?

Problemas comunes: ¿Se evitan…

- ☐ … nombres que son diferentes en sólo uno o dos caracteres, que suenan similares o usan números?
- ☐ … nombres mal escritos intencionalmente para hacerlos más cortos o que comúnmente se escriben mal en español?
- ☐ … nombres en conflicto con palabras reservadas, nombres de rutinas o funciones de la biblioteca estándar, o nombres de datos predefinidos?

Consideraciones Generales del Uso de Datos

- ☐ ¿Las estructuras de control corresponden a las estructuras de datos?
- ☐ ¿Cada variable tiene uno y sólo un propósito?
- ☐ ¿Todas las variables declaradas se usan?

Datos Fundamentales

Números en general

- ☐ ¿El código evita números "mágicos"?
- ☐ ¿El código anticipa los errores de división por cero?

Enteros

- ☐ ¿Las expresiones que usan división entera se evalúan de la manera que deben (con operandos enteros)?
- ☐ ¿Las expresiones de enteros evitan problemas de desbordamiento?

Tipos Enumerados

- ☐ ¿Las comprobaciones de tipos enumerados revisan valores incorrectos, es decir, fuera de la enumeración?
- ☐ ¿La primera entrada en un tipo enumerado está reservado para "no válido" (0)?

Checklist

Programación de Computadoras

Constantes

- ☐ ¿El programa usa constantes con nombres en la declaración de datos?
- ☐ ¿Las constantes nombradas han sido usadas consistentemente – No usa constantes con nombre en una parte y literales en otra?

Condicionales

Sentencias *IF*

- ☐ ¿Las cláusulas *if* y *else* son usadas correctamente?
- ☐ ¿El caso normal sigue de *if* en vez de *else*?

Cadenas *IF-ELSE-IF*

- ☐ ¿Son considerados todos los casos?
- ☐ ¿Se usan las cadenas *if-else-if* sólo cuando es mejor su implementación que la sentencia *switch*?

Sentencias *SWITCH*

- ☐ ¿La cláusula *default* es usada para detectar y reportar casos inesperados?

Ciclos

- ☐ ¿El código de inicialización está inmediatamente antes del ciclo?
- ☐ ¿Los ciclos infinitos realmente lo son, es decir, no usan incorrectamente un límite que aparenta ser enorme?
- ☐ En el ciclo *for* de C, ¿la cabecera del ciclo está reservada para el código de control del ciclo y, por tanto, el cuerpo del ciclo es no vacío?
- ☐ ¿Las sentencias de control del ciclo están agrupadas, al principio o al final del ciclo?
- ☐ ¿El ciclo desempeña una y sólo una función – tal como lo hace una rutina bien definida?
- ☐ ¿El ciclo finaliza bajo todas las condiciones posibles?
- ☐ ¿La condición de terminación del ciclo es obvia?
- ☐ ¿En el ciclo *for*, el código interno evita el uso indebido del índice del ciclo?
- ☐ ¿Se usa una variable para guardar valores importantes del índice del ciclo, en vez de usar el índice fuera del ciclo?

Estructuras de Control Inusuales

return

- ☐ ¿Cada rutina usa el menor número de *returns* posibles?
- ☐ ¿Los *returns* aumentan la legibilidad, en vez de oscurecer el código?

Recursión

- ☐ ¿El código en la rutina recursiva incluye el caso base para terminar la recursión?
- ☐ ¿La implementación recursiva de la rutina es mejor que la iterativa?

Checklist

Programación de Computadoras

Edición de Estructuras de Control

- ☐ ¿Las expresiones usan *TRUE* y *FALSE* (o sus equivalentes) en vez de 1 y 0?
- ☐ ¿Los valores booleanos son comparados implícitamente?
- ☐ ¿Las expresiones booleanas preguntan por el caso verdadero?
- ☐ ¿Los números, caracteres y punteros son comparados a 0, '\0' o *NULL* explícitamente?

Apariencia

General

- ☐ ¿El formato está hecho principalmente para aclarar la estructura lógica del código?
- ☐ ¿El formato puede ser usado consistentemente, resulta en código fácil de mantener y mejora la legibilidad?

Formato:

```
void  main()
{    // <= La llave de inicio del programa principal se escribe en una línea separada

       const unsigned LIMITE = 40u;
       unsigned edad;
       …
       if(edad == LIMITE){   //<= La llave de apertura se escriben al final y en la misma línea donde inicia la estructura

              …Hacer algo          // Use sangrías para indicar distintos niveles de indentación (sangrado)

       }   // <= La llave de cierre va en una línea aparte al mismo nivel del inicio de la estructura
                        …
}   // <= La llave final del programa principal se escribe en una línea separada al mismo nivel de la de inicio
```

Estructuras de control

- ☐ ¿El código evita indentaciones dobles en pares *{ }*?
- ☐ ¿Los bloques secuenciales son separados entre sí por líneas en blanco?
- ☐ ¿Las expresiones complicadas son formateadas (modificadas) para hacerlas más legibles?
- ☐ ¿Los bloques de sentencias simples (estructuras de control) son formateadas consistentemente?

Sentencias Individuales

- ☐ ¿Las sentencias incompletas terminan de una forma que es obviamente incorrecta, y las líneas de continuación tienen una indentación razonable?
- ☐ ¿Cada línea contiene a lo más una sentencia?

Comentarios

- ☐ ¿Los comentarios están indentados al mismo nivel del código que comentan?
- ☐ ¿El estilo de los comentarios es fácil de mantener (modificar)?

Archivos, Módulos y Programas

- ☐ ¿Cada archivo contiene código para uno y sólo un módulo?
- ☐ ¿Las rutinas dentro de un archivo son claramente separadas, por ejemplo con líneas en blanco?

Checklist

Programación de Computadoras

Código Auto–Documentado

Diseño

- ☐ ¿El código se lee de corrido de arriba hacia abajo, es decir, sin interrupciones (**goto**)?
- ☐ ¿La ruta principal a través del código es clara?
- ☐ ¿Los detalles de implementación están lo más ocultos posible?

Buena Técnica para Comentar

General

- ☐ ¿El listado fuente contiene la mayor parte de la información acerca del programa?
- ☐ ¿Puede alguien tomar el código y empezar a entenderlo inmediatamente?
- ☐ ¿Los comentarios explican la intención del código o resumen lo que el código hace en vez de sólo repetirlo?
- ☐ ¿El código demasiado "ingenioso" ha sido reescrito en vez de comentado?
- ☐ ¿Los comentarios están actualizados?

Sentencias y Párrafos

- ☐ ¿El código evita comentarios de fin de línea?
- ☐ ¿Los comentarios se enfocan más en el *por qué* que en el *cómo*?
- ☐ ¿Han sido evitadas las abreviaciones?

Estructuras de Control

- ☐ ¿Los finales de estructuras de control largas o complejas están comentadas?

Archivos, Módulos y Programas

- ☐ ¿Se describe el propósito de cada archivo?