Mejora de un VI existente

Un problema común cuando hereda VIs de otros desarrolladores es que pueden haberse añadido características sin prestar atención al diseño, lo que dificulta cada vez más añadir nuevas características durante la vida del VI. Esto se conoce como decadencia del software. Una solución a la decadencia del software es refactorizar el software. Refactorización es el proceso de volver a diseñar software para hacerlo más legible y mantenible para que el coste del cambio no aumente con el tiempo. La refactorización cambia la estructura interna de un VI para hacerlo más legible y mantenible, sin cambiar su comportamiento perceptible.

En esta sección aprenderá métodos para refactorizar código heredado y experimentar con problemas típicos del código heredado.

Temas

A. Refactorización de código heredado

B. Problemas típicos de la refactorización

A. Refactorización de código heredado

Escriba aplicaciones de software grandes o a largo plazo teniendo presente la legibilidad, porque el coste de leer y modificar el software probablemente supere con creces el de ejecutarlo. A un desarrollador le cuesta más leer y comprender código mal diseñado que leer código que se creó para ser legible. En general, se asignan más recursos a leer y a modificar software que a la implementación inicial. Por lo tanto, los VIs que son fáciles de leer y de modificar son más valiosos que los que no lo son.

La creación de software bien diseñado facilita el desarrollo rápido y disminuye la posible decadencia. Si un sistema empieza a decaer, puede pasar enormes cantidades de tiempo localizando fallos de regresión, lo cual no es productivo. Los cambios también tardan más en implementarse porque resulta más difícil entender el sistema si está mal diseñado.

Observe el VI heredado que aparece en la figura 6-1.

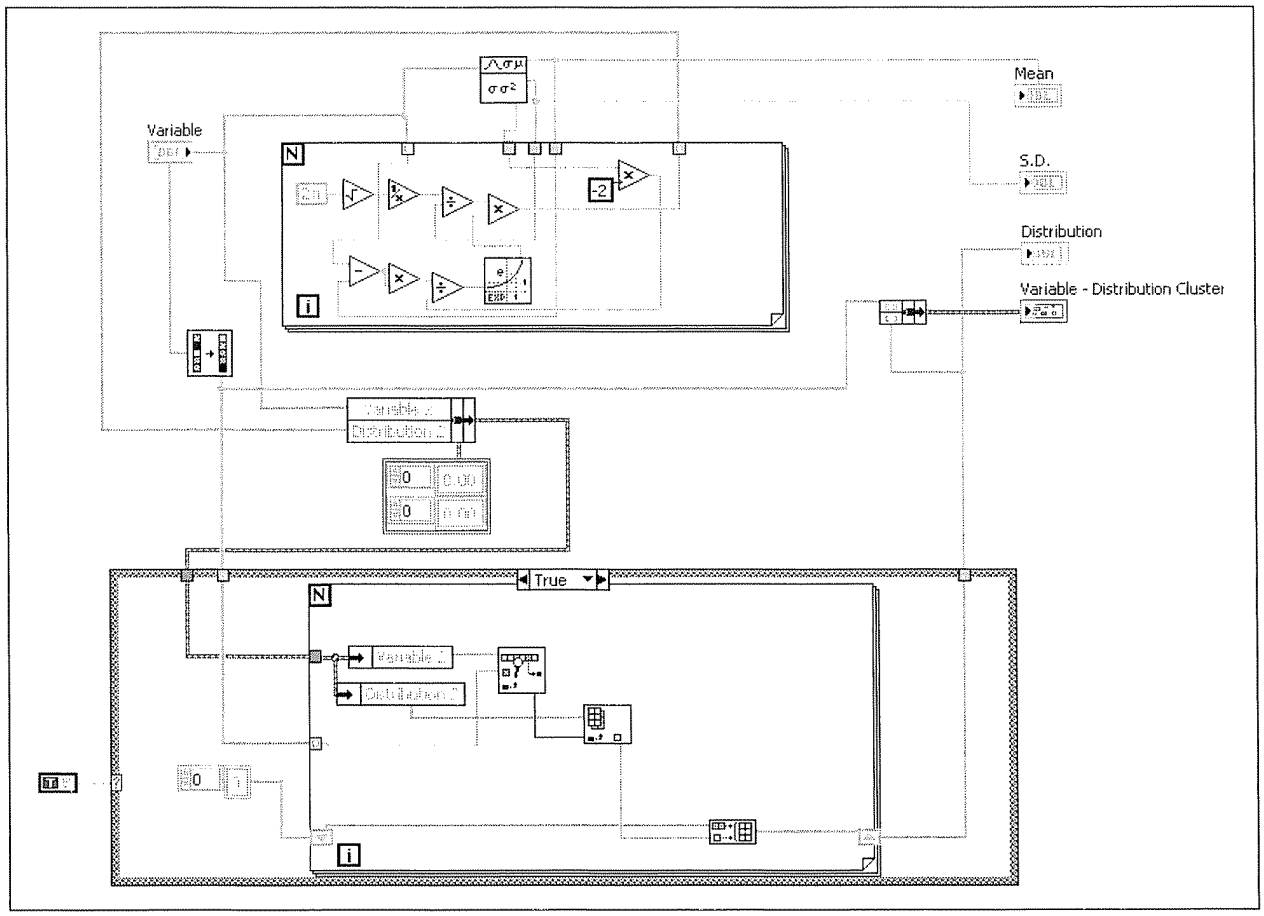


Figura 6-1. VI heredado

Puede refactorizar el código, como en la figura 6-2.

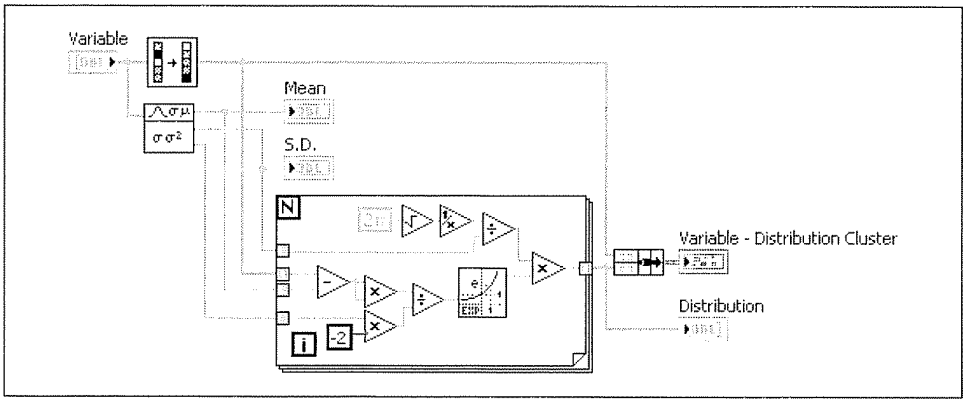


Figura 6-2. Código heredado refactorizado

El código refactorizado realiza la misma función que el código heredado, pero aquél es más legible. El código heredado viola muchas de las pautas del diagrama de bloques que ha aprendido.

Cuando hace un VI más fácil de entender y de mantener, adquiere más valor porque resulta más sencillo añadir funciones o depurar el VI. El proceso de refactorización no cambia el comportamiento perceptible. Cambiar el modo en que un VI interactúa con los clientes (usuarios u otros VIs) introduce riesgos que no existen cuando limita los cambios a los que ven sólo los desarrolladores. La ventaja de mantener los dos tipos de cambios por separado es que puede gestionar mejor los riesgos.

Refactorización frente a la optimización del rendimiento

Aunque puede realizar cambios que optimizan el rendimiento de un VI, no es lo mismo que refactorizar. La refactorización cambia concretamente la estructura interna de un VI para hacerla más legible, comprensible y mantenible. Una optimización del rendimiento no es refactorizar, ya que el objetivo de la optimización no es que el VI sea más fácil de entender y de modificar. De hecho, la optimización del rendimiento puede hacer que los VIs sean más difíciles de leer y entender, lo que puede ser una compensación aceptable. A veces debe sacrificar la legibilidad por el rendimiento mejorado, aunque la legibilidad normalmente tiene prioridad respecto a la velocidad de rendimiento.

Cuándo refactorizar

El momento adecuado de refactorizar es cuando está añadiendo una característica a un VI o depurándolo. Aunque podría verse tentado a rescribir el VI desde cero, existe un valor en un VI que funciona, aunque el diagrama de bloques no sea legible. Los buenos candidatos a rescribirse completamente son los VIs que no funcionan o VIs que satisfacen sólo una pequeña parte de sus necesidades. También puede rescribir VIs sencillos que no entiende bien. Piense en lo que funciona bien en un VI existente antes de refactorizarlo.

B. Problemas típicos de la refactorización

Cuando refactorice un VI, gestione el riesgo de introducir errores realizando pequeños cambios incrementales al VI y probando el VI después de cada cambio. El diagrama de flujo de la figura 6-3 indica el proceso para refactorizar un VI.

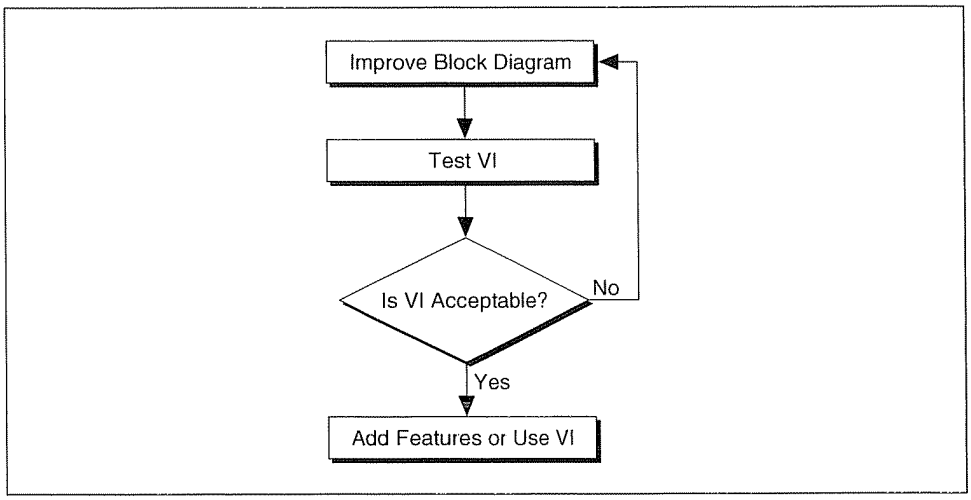


Figura 6-3. Diagrama de flujo de refactorización

Cuando refactorice para mejorar el diagrama de bloques, realice pequeños cambios de aspecto antes de abordar cuestiones más importantes. Por ejemplo, resulta más fácil buscar código duplicado si el diagrama de bloques está bien organizado y los terminales bien etiquetados.

Existen varios problemas que pueden complicar el trabajo con un VI heredado. Las siguientes secciones describen problemas típicos y las soluciones de refactorización que puede utilizar para hacer VIs heredados más legibles.

Diagrama de bloques desorganizado o mal diseñado

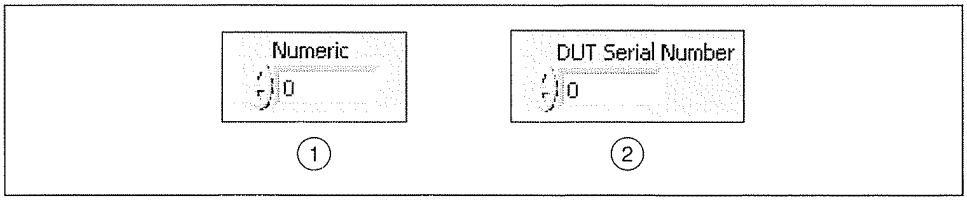
Mejore la legibilidad de un VI desorganizado reubicando objetos del diagrama de bloques. También puede crear subVIs para secciones del VI que estén desorganizadas. Coloque comentarios en zonas desorganizadas de un VI para mejorar su legibilidad.

Diagrama de bloques demasiado grande

Un VI que tiene un diagrama de bloques mayor que el tamaño de la pantalla resulta difícil de leer. Debe refactorizar el VI para reducirlo. El desplazamiento complica la lectura de un diagrama de bloques y la comprensión del código. Mejore un gran diagrama de bloques moviendo objetos. Otra técnica para reducir el espacio que ocupa un diagrama de bloques en la pantalla es crear subVIs para secciones de código del diagrama de bloques. Si no puede reducir el diagrama de bloques para que se ajuste a la pantalla, limite el desplazamiento a una dirección.

Objetos mal nombrados e iconos mal diseñados

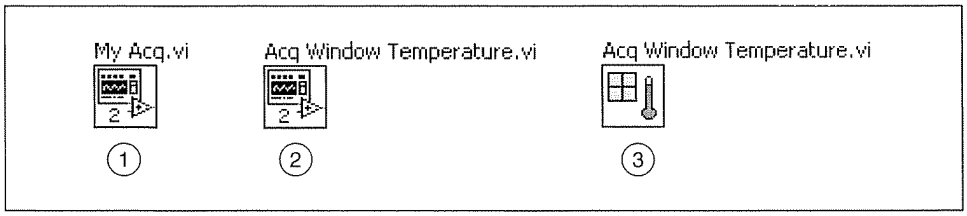
Los VIs heredados suelen contener controles e indicadores que no tienen nombres significativos. Por ejemplo, el nombre del Control 1 de la figura 6-4 no indica su finalidad. El Control 2 es el mismo control, pero con otro nombre para facilitar la lectura y comprensión del diagrama de bloques.



1 Control mal nombrado 2 Control nombrado significativamente

Figura 6-4. Elección de nombres de controles

Los nombres y los iconos del VI también son importantes para mejorar la legibilidad de un VI. Por ejemplo, el nombre `My Acqu.vi`, mostrado a la izquierda en la figura 6-5, no informa de la finalidad del VI. Puede dar al VI un nombre más significativo guardando una copia del VI con otro nombre y sustituyendo todas las instancias del VI por el VI renombrado. Un método más sencillo es abrir todos los llamadores del VI que desea renombrar y después guardar el VI con otro nombre. Si usa este método, LabVIEW volverá a enlazar automáticamente todos los llamadores abiertos del VI con el nuevo nombre. `Acq Window Temperature.vi` refleja un nombre más significativo del VI.



1 VI mal nombrado
2 VI nombrado significativamente
3 Nombre e icono significativos del VI

Figura 6-5. SubVI mal nombrado

El icono del VI también debe aclarar su finalidad. Los iconos predeterminados utilizados para el VI 1 y VI 2 en la figura 6-5 no representan su finalidad. Puede mejorar la legibilidad del VI mediante un icono significativo, como se muestra para el VI 3.

Al renombrar controles y VIs y crear iconos significativos para el VI, puede mejorar la legibilidad de un VI heredado.

Lógica innecesaria

Al leer el diagrama de bloques de la figura 6-6, observe que contiene lógica innecesaria. Si no se ejecuta una parte del diagrama de bloques, elimínela. Resulta difícil comprender código que se ejecuta, pero intentar entender el código que nunca se ejecuta es ineficaz y complica el diagrama de bloques.

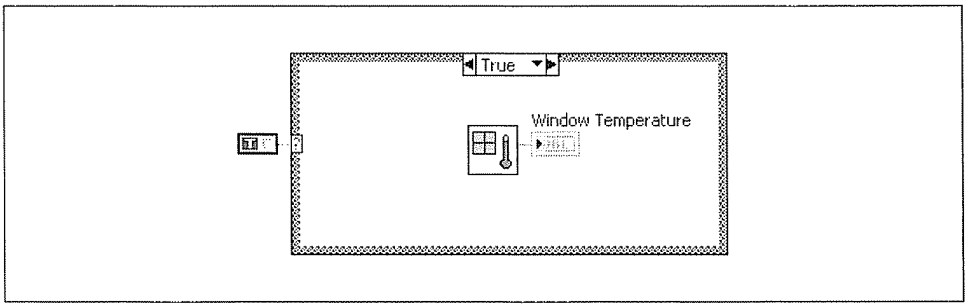


Figura 6-6. Lógica innecesaria

Lógica duplicada

Si un VI contiene lógica duplicada, siempre debe refactorizar el VI creando un subVI para la lógica duplicada. Esto puede mejorar la legibilidad y testabilidad del VI.

Falta de programación de flujo de datos

Si hay estructuras Sequence y variables locales en el diagrama de bloques, el VI probablemente no utilizará el flujo de datos para determinar el flujo de programación.

Debe sustituir la mayoría de las estructuras Sequence por el patrón de diseño de la máquina de estados. Elimine las variables locales y cablee los controles e indicadores directamente.

Algoritmos complicados

Los algoritmos complicados dificultan la lectura de un VI. Los algoritmos complicados pueden ser más difíciles de refactorizar porque existe mayor probabilidad de que los cambios introduzcan errores. Cuando refactorice un algoritmo complicado, realice pequeños cambios para probar el código frecuentemente. En algunos casos puede refactorizar un algoritmo complicado utilizando funciones integradas en LabVIEW. Por ejemplo, el VI de la figura 6-7 comprueba un nombre de usuario y una contraseña en una base de datos.

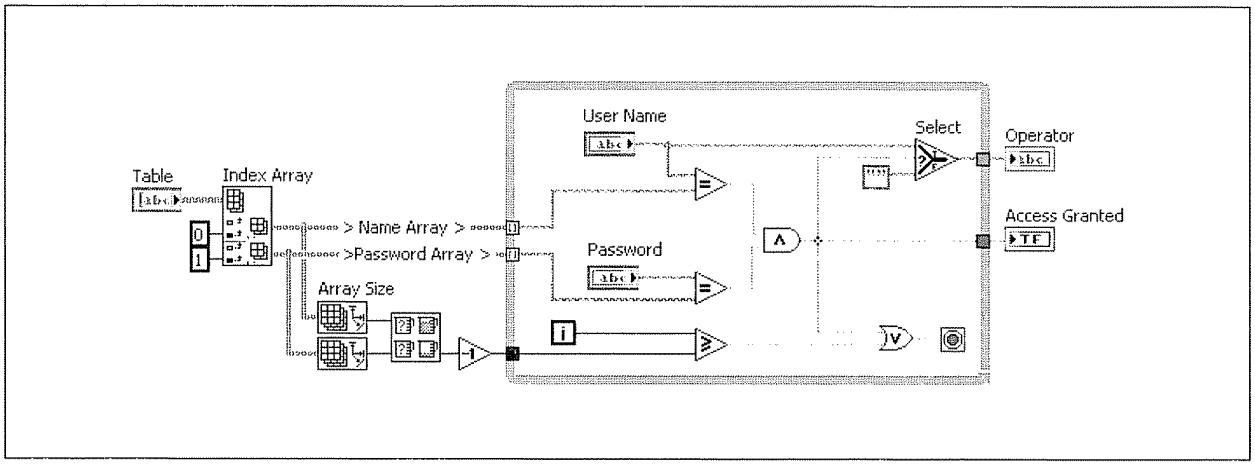


Figura 6-7. VI de un algoritmo complicado

Puede refactorizar este VI usando las funciones integradas para buscar cadenas de caracteres, como en la figura 6-8.

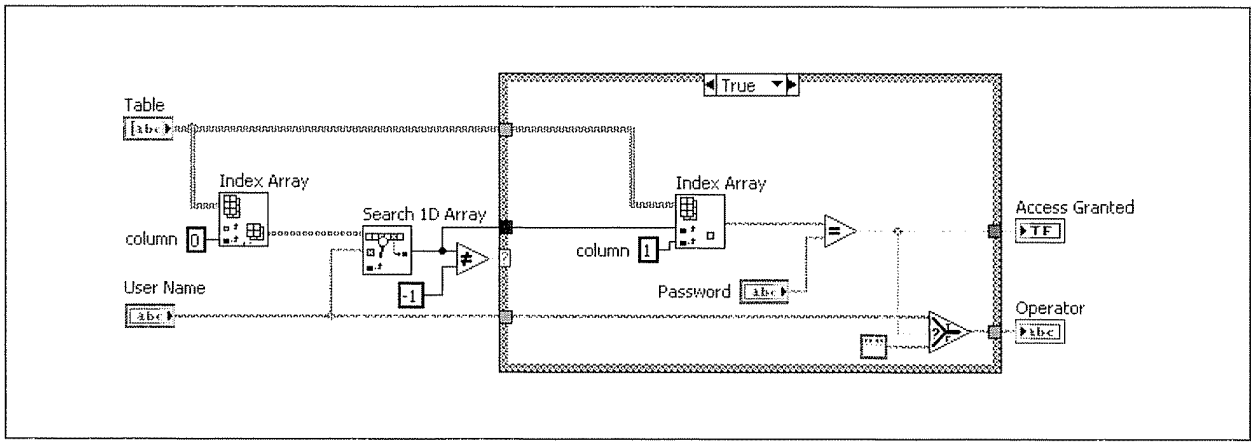


Figura 6-8. VI refactorizado

Ejercicio 6-1 Concepto: problemas típicos

Objetivo

Mejorar un VI existente que esté mal diseñado.

Descripción

Recibe un VI que se utiliza como subVI en un proyecto mayor.
Debe mejorar la legibilidad y la sencillez de uso del VI.

Evaluación del VI

1. Abra el `Determine Warning Bad One.vi` en el directorio `<Exercises>\LabVIEW Basics II\Determine Warnings`. La figura 6-9 muestra el diagrama de bloques de este VI.

2. Use la siguiente lista para evaluar el VI. Marque todos los problemas que presente.

   - ❑ Diagrama de bloques desorganizado
   - ❑ Diagrama de bloques demasiado grande
   - ❑ Objetos mal nombrados e iconos mal diseñados
   - ❑ Lógica innecesaria
   - ❑ Lógica duplicada
   - ❑ Falta de programación de flujo de datos
   - ❑ Algoritmos complicados

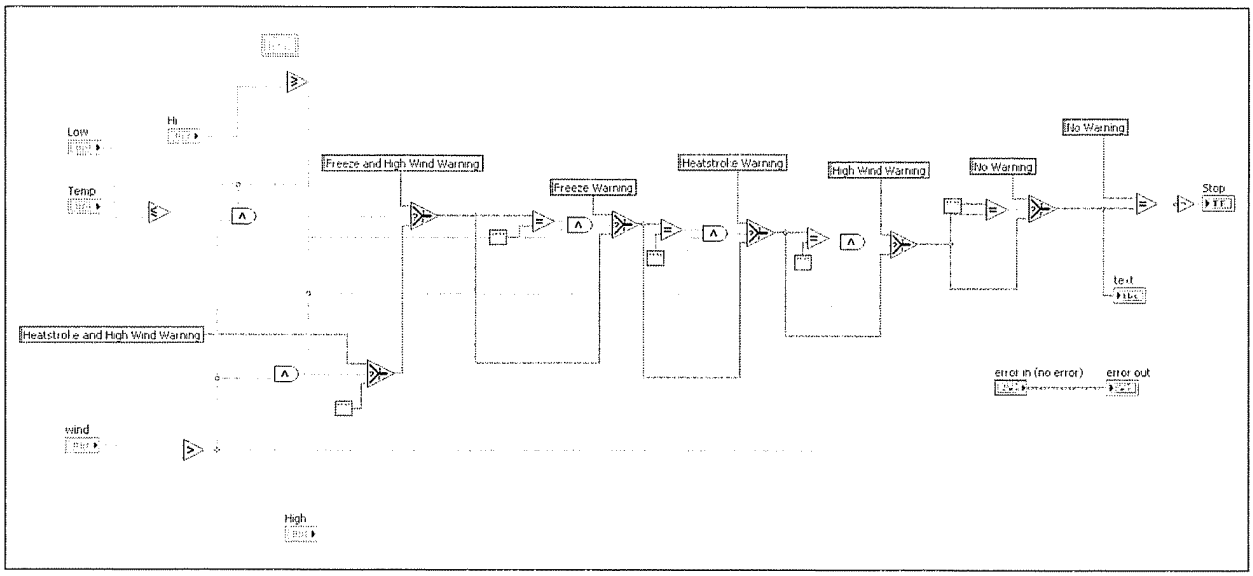


Figura 6-9. Diagrama de bloques mal diseñado

Mejora del VI

Mejore el VI por etapas. Empiece por la primera marca: el diagrama de bloques está demasiado desorganizado.

1. Siga estos consejos para ayudarle a organizar el diagrama de bloques:

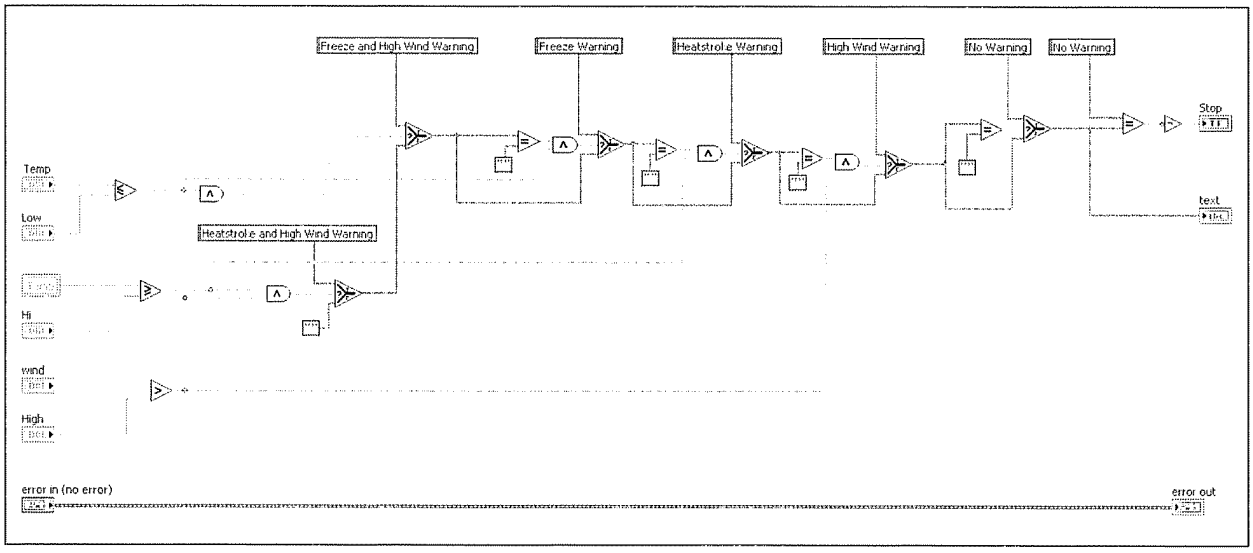


Figura 6-10. Diagrama de bloques reorganizado

- ❑ Mueva todos los controles a la izquierda del diagrama de bloques.
- ❑ Mueva todos los indicadores a la derecha del diagrama de bloques.

- ❑ Use los botones de la barra de herramientas **Align Objects** y **Distribute Objects** para organizar los controles e indicadores.
- ❑ Reorganice los cables para que no se solapen.
- ❑ Reorganice los cables para que no discurran cables de entrada de derecha a izquierda.
- ❑ Reduzca el número de curvas en los cables.
- ❑ Los cables no deben discurrir bajo objetos.

2. Una vez que el diagrama de bloques esté mejor organizado, renombre los controles e indicadores usando nombres que sean más descriptivos.
   - El objetivo de este VI es determinar si la temperatura y la velocidad del viento actuales están en un nivel que requiera generar una advertencia. El VI también ilumina un LED si hay una alarma.
   - Los nombres de entradas sugeridos son `Current Temperature`, `Low Temp`, `High Temp`, `Current Wind Speed` y `High Wind Speed`.
   - Los nombres de salidas sugeridos son `Warning Text` y `Warning?`.
3. Elimine la lógica innecesaria del diagrama de bloques.

   La figura 6-11 muestra una función Equal? seguida de una función Not.

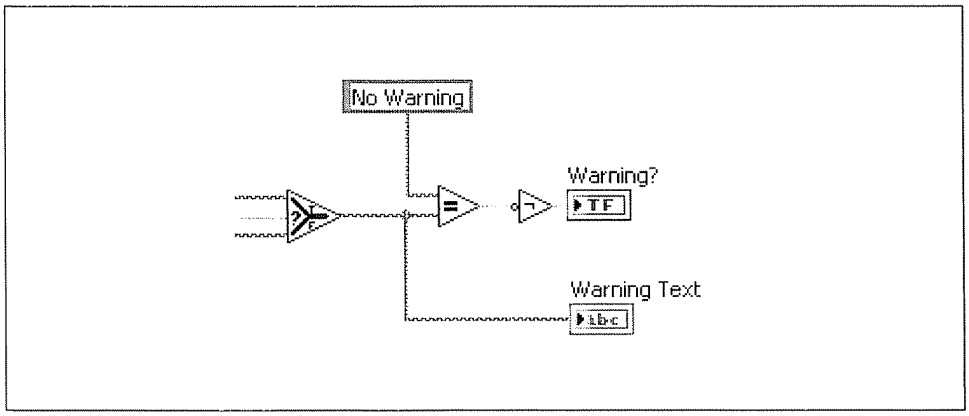


Figura 6-11. Lógica innecesaria

Nota Si el subVI está atenuado, haga clic derecho en el icono del subVI en el diagrama de bloques y seleccione **Relink to SubVI** en el menú contextual.

- ❑ Elimine la lógica duplicada en otras ubicaciones y sustitúyala por el nuevo subVI.
- ❑ Pruebe el VI modificado.

2. Elimine las variables locales innecesarias y cablee al control o indicador apropiado.

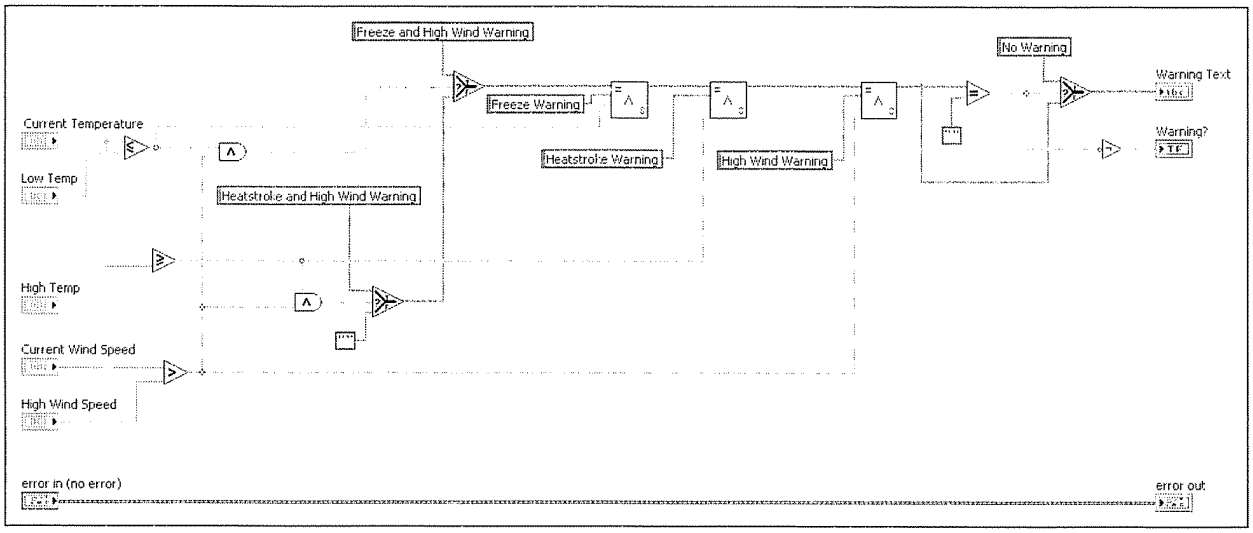


Figura 6-16. Lógica duplicada situada en un subVI y variables locales eliminadas

3. Guarde el VI como `Determine Warnings Good One.vi`.

Reto: simplificación de algoritmo

Si le sobra tiempo en este ejercicio, intente determinar un modo para simplificar el algoritmo y volver a escribir el código para facilitar las posteriores modificaciones.

Un ejemplo de solución se muestra en la figura 6-17 usando una máquina de estado. Los estados que contiene son: Heatstroke, Freeze, High Wind y Generate Warning. Puede explorar esta solución en el `Determine Warnings State Machine.vi` situado en el directorio `<Exercises>\LabVIEW Basics I\Determine Warnings`. El proyecto del curso también usa esta solución.

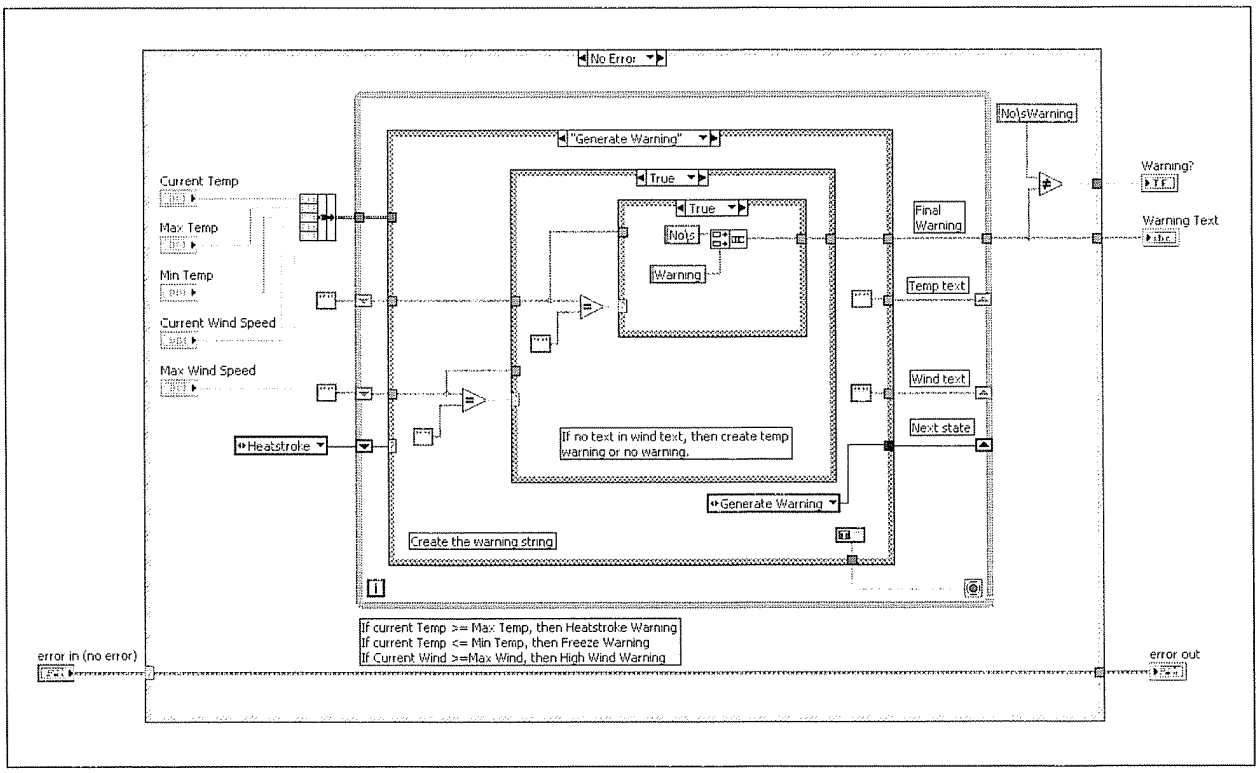


Figura 6-17. Algoritmo simplificado legible y mantenible

Fin del ejercicio 6-1

- Puede replicar esto con una función Not Equal?, completando la misma lógica con menos funciones, como se muestra en la figura 6-12.

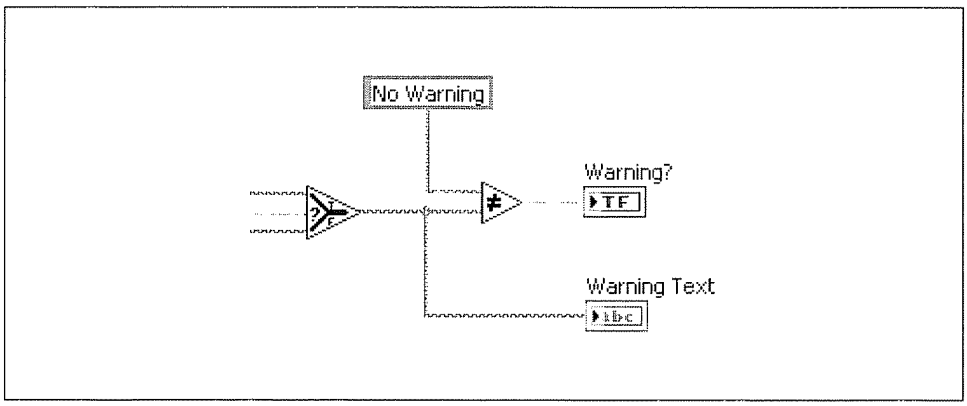


Figura 6-12. Lógica innecesaria simplificada

- Puede reducir más la lógica innecesaria utilizando la entrada booleana de la función Select, como se ve en la figura 6-13.

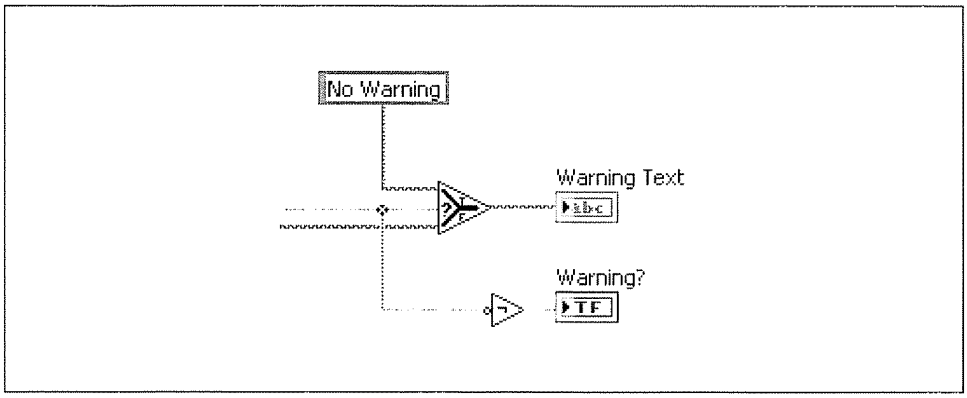


Figura 6-13. Lógica innecesaria más simplificada

Consulte la figura 6-14 para obtener ayuda acerca del cableado de esta función duplicada que ocurre cerca del final del VI.

- Elimine la función Equal?.
- Elimine el cable de entrada a la función Not.
- Cablee la entrada de la función Not al cable de entrada de la función Select.
- Pruebe el VI modificado para asegurarse de que la funcionalidad no ha cambiado.

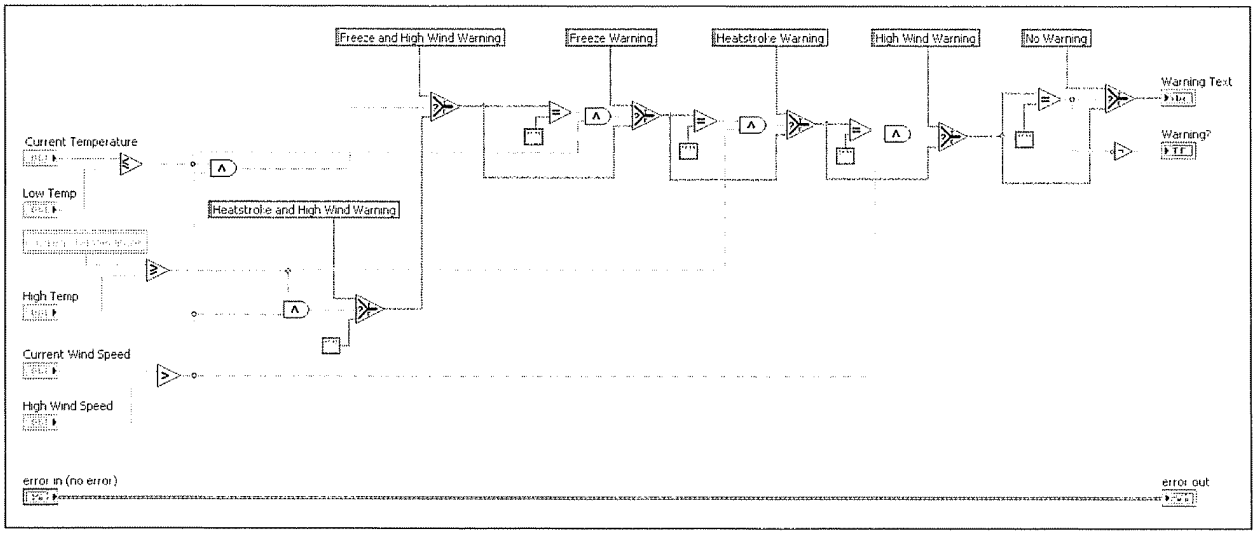


Figura 6-14. Controles bien nombrados y lógica innecesaria eliminada

4. Guarde el VI como `Determine Warnings Good One.vi`.

Opcional

1. Sustituya la lógica duplicada del diagrama de bloques por subVIs. La figura 6-15 muestra un ejemplo del algoritmo del VI que se repite. Puede sustituir este algoritmo por un subVI.

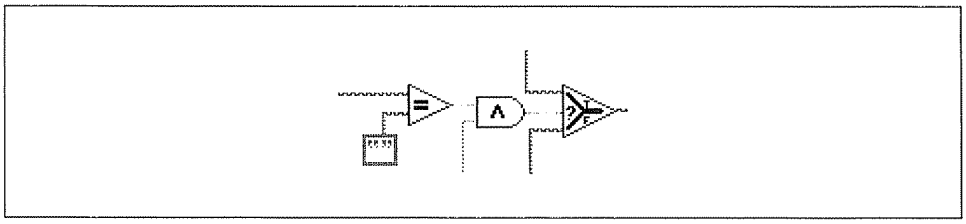


Figura 6-15. Algoritmo repetido

- ❑ Seleccione el algoritmo repetido dibujando un cuadro de selección alrededor de los objetos.
- ❑ Seleccione **Edit»Create SubVI**.
- ❑ Haga doble clic en el nuevo subVI para abrirlo.
- ❑ Modifique el nuevo subVI si es necesario. No olvide: crear un icono apropiado, volver a crear el panel de conectores y renombrar los controles e indicadores.
- ❑ Guarde el subVI.
- ❑ Cierre el subVI.

Ayuda para el trabajo

Use la siguiente lista de comprobación de refactorización para ayudarle a determinar si debe refactorizar un VI. Si responde sí a alguno de los elementos de la lista de comprobación, consulte las pautas de la sección *Cuándo refactorizar* de esta lección para refactorizar el VI.

- ❑ Diagrama de bloques desorganizado
- ❑ Diagrama de bloques demasiado grande
- ❑ Objetos mal nombrados e iconos mal diseñados
- ❑ Lógica innecesaria
- ❑ Lógica duplicada
- ❑ Falta de programación de flujo de datos
- ❑ Algoritmos complicados