**Chapter 15. Using Conditionals**

Contents

* if Statements
* case Statements

Un condicional es una declaración que controla la ejecución de otras declaraciones; la ejecución de las otras declaraciones está "condicionada" a declaraciones como if, else, case y switch.

# 15.1 if Statements

Dependiendo del idioma que esté usando, es posible que pueda usar varios tipos de declaraciones if.

● La más simple es la declaración simple si o si-entonces.

● El if-then-else es un poco más complejo y las cadenas de if-then-else-if son las más complejas.

**Declaraciones sencillas si-entonces**

Siga estas pautas al escribir declaraciones if:

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

○ Escriba su código para que la ruta normal a través del código sea clara.

○ Asegúrese de que los casos excepcionales no oculten la ruta normal de ejecución. Esto es importante tanto para la legibilidad como para el rendimiento.

Coloque el caso normal después del if en lugar del else.

○ Ponga el caso que normalmente espera procesar primero. Esto está en consonancia con el principio general de poner el código que resulta de una decisión lo más cerca posible de la decisión.

Example 15-1. Visual Basic Example of Code That Processes a Lot of Errors

Errores:

● los casos nominales y los casos de error están todos mezclados.

● Es difícil encontrar la ruta que normalmente se toma a través del código.

● Además, debido a que las condiciones de error a veces se procesan en la cláusula if en lugar de en la cláusula else, es difícil determinar con qué prueba if corresponde el caso normal.

### Para obtener una lista completa de los enfoques disponibles, consulte "Resumen de técnicas para reducir el anidamiento profundo" en Domar el anidamiento peligrosamente profundo.Follow the if clause with a meaningful statement.

Example: Java Example of a Null if Clause

if ( SomeTest )

;

else {

// do something

...

}

Cross-Reference

One key to constructing an effective if statement is writing the right boolean expression to control it.

Example 15-4. Java Example of a Converted Null

if ( ! SomeTest ) {

// do something

...

}

● Considere la cláusula else. Si cree que necesita una declaración if simple, considere si realmente no necesita una declaración if-then-else.

● Una opción es codificar la cláusula else (con una declaración nula si es necesario) para mostrar que se ha considerado el caso else.

● Utilice comentarios para explicar por qué la cláusula else no es necesaria.

Example 15-5. Java Example of a Helpful, Commented else Clause

// if color is valid

if ( COLOR\_MIN <= color && color <= COLOR\_MAX ) {

// do something

...

}

else {

// else color is invalid

// screen not written to –- safely ignore command

en Python debemos escribir pass

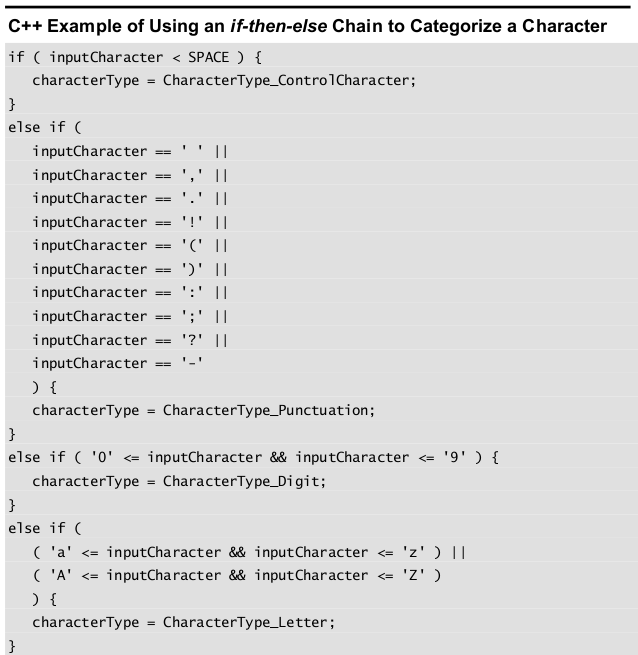
}

# 15.2 Chains of if-then-else Statements

## En los lenguajes que no admiten declaraciones de casos, o que las admiten solo parcialmente, a menudo se encontrará escribiendo cadenas de pruebas si-entonces-si no.

## Simplify complicated tests with boolean function calls.

Example 15-7. C++ Example of an if-then-else Chain That Uses Boolean Function Calls



C++ Example of an if-then-else Chain That Uses Boolean Function Calls

if ( IsControl( inputCharacter ) ) {

characterType = CharacterType\_ControlCharacter;

}

else if ( IsPunctuation( inputCharacter ) ) {

characterType = CharacterType\_Punctuation;

}

else if ( IsDigit( inputCharacter ) ) {

characterType = CharacterType\_Digit;

}

else if ( IsLetter( inputCharacter ) ) {

characterType = CharacterType\_Letter;

}

## Put the most common cases first.

Poniendo los casos más comunes primero:

● minimiza la cantidad de código de manejo de casos de excepción que alguien tiene que leer para encontrar los casos habituales.

● Mejora la eficiencia porque minimiza la cantidad de pruebas que realiza el código para encontrar los casos más comunes.

C++ Example of Testing the Most Common Case First

if ( IsLetter( inputCharacter ) ) {

characterType = CharacterType\_Letter;

}

else if ( IsPunctuation( inputCharacter ) ) {

characterType = CharacterType\_Punctuation;

}

else if ( IsDigit( inputCharacter ) ) {

characterType = CharacterType\_Digit;

}

else if ( IsControl( inputCharacter ) ) {

characterType = CharacterType\_ControlCharacter;

}

**Asegúrese de que todos los casos estén cubiertos.**

Codifique una cláusula else final con un mensaje de error o una afirmación para detectar los casos que no planeó. Este mensaje de error está dirigido a usted y no al usuario, por lo que debe redactarlo correctamente.

Example 15-9. C++ Example of Using the Default Case to Trap Errors

if ( IsLetter( inputCharacter ) ) {

characterType = CharacterType\_Letter;

}

else if ( IsPunctuation( inputCharacter ) ) {

characterType = CharacterType\_Punctuation;

}

else if ( IsDigit( inputCharacter ) ) {

characterType = CharacterType\_Digit;

}

else if ( IsControl( inputCharacter ) ) {

characterType = CharacterType\_ControlCharacter;

}

else {

DisplayInternalError( "Unexpected type of character detected." );

}

## Replace if-then-else chains with other constructs if your language supports them.

## 15.2 Case Statements

## Choosing the Most Effective Ordering of Cases

**Ordene los casos alfabéticamente o numéricamente**

● Si los casos son igualmente importantes, ponerlos en orden A-B-C mejora la legibilidad.

● Un caso específico es fácil de identificar en el grupo.

**Pon el caso normal primero**

● Si tiene un caso normal y varias excepciones, coloque el caso normal primero.

● Indicar con comentarios que es el caso normal y que los demás son inusuales.

### Order cases by frequency

Coloque los casos ejecutados con más frecuencia en primer lugar y los casos ejecutados con menos frecuencia al final.

Este enfoque tiene dos ventajas:

● Los lectores humanos pueden encontrar fácilmente los casos más comunes. Es probable que los lectores que examinen la lista en busca de un caso específico estén interesados en uno de los casos más comunes.

● Poner los comunes en la parte superior del código hace que la búsqueda sea más rápida.

## Tips for Using case Statements

### Keep the actions of each case simple

Mantenga breve el código asociado con cada caso.

El código corto que sigue a cada caso ayuda a aclarar la estructura de la declaración del caso.

Si las acciones realizadas para un caso son complicadas, escriba una rutina y llame a la rutina desde el caso en lugar de poner el código en el propio caso.

### Don’t make up phony variables in order to be able to use the case statement

Se debe usar una declaración de caso para datos simples que se pueden categorizar fácilmente. Si sus datos no son simples, use cadenas de if-then-elses en su lugar.

### Use the default clause only to detect legitimate defaults

A veces, es posible que solo le quede un caso y decida codificar ese caso como la cláusula predeterminada. Aunque a veces es tentador, eso es tonto. Pierde la documentación automática proporcionada por las etiquetas de declaración de casos y pierde la capacidad de detectar errores con la cláusula predeterminada.

### Use the default clause to detect errors

Si la cláusula predeterminada en una declaración de caso no se usa para otro procesamiento y no se supone que ocurra, coloque un mensaje de diagnóstico en ella.

Mensajes como este son útiles tanto en depuración como en código de producción.

#### Java Example of Using the Default Case to Detect Errors—Good Practice

switch ( commandShortcutLetter ) {

case 'a':

PrintAnnualReport();

break;

case 'p':

// no action required, but case was considered

break;

case 'q':

PrintQuarterlyReport();

break;

case 's':

PrintSummaryReport();

break;

default:

DisplayInternalError( "Internal Error 905: Call customer support." );

}

### In C++ and Java, avoid dropping through the end of a case statement

Los lenguajes similares a C (C, C ++ y Java) no salen automáticamente de cada caso.

En su lugar, debe codificar el final de cada caso explícitamente. Si no codifica el final de un caso, el programa pasa al final y ejecuta el código para el siguiente caso.

# Comparing Switches and if-then Statements

Del libro Beginning Java Programming

Al igual que los bucles for y while son estructuras similares, los conmutadores y las declaraciones if-then también son fáciles de comparar.

Cuando usa un interruptor, lo lee de la misma manera que una declaración si-entonces: si el

valor coincide con el caso, luego haga algo. Entonces, ¿cómo sabes cuándo usar cada uno?

Al igual que con las otras estructuras de control, habrá situaciones en las que cualquiera sea apropiado y usted

Puede elegir según sus propias preferencias. A medida que continúe codificando, su experiencia le dirá si un problema se resolvería mejor con un interruptor o no.

En general, puede considerar los siguientes criterios:

➤➤ Si tiene una sola variable que puede tomar varios valores, un interruptor puede ser adecuado.

➤➤ Si tiene que considerar varias variables o condiciones, probablemente necesitará una declaración si-entonces.

➤➤ Si el valor que está considerando puede tener un número finito de valores, considere usar un interruptor.

➤➤ Si la variable puede tomar cualquier valor dentro de un rango continuo de números, considere un if-then

declaración.

----------------------------------------------------------------------------------------------------------------------------

LISTA DE VERIFICACIÓN: Uso de condicionales

Declaraciones si-entonces

❏ ¿Está clara la ruta nominal a través del código?

❏ ¿Las pruebas si-entonces se ramifican correctamente en igualdad?

❏ ¿Está presente y documentada la cláusula else?

❏ ¿Es correcta la cláusula else?

❏ ¿Se utilizan correctamente las cláusulas if y else, no se invierten?

❏ ¿El caso normal sigue al if en lugar del else?

Cadenas if-then-else-if

❏ ¿Las pruebas complicadas están encapsuladas en llamadas a funciones booleanas?

❏ ¿Se examinan primero los casos más comunes?

❏ ¿Están cubiertos todos los casos?

❏ ¿Es la cadena if-then-else-if la mejor implementación, mejor que una declaración de caso?

-------------------------------------------------- -------------------------------------------------- ------------------------

PUNTOS CLAVE

● Para declaraciones simples if-else, preste atención al orden de las cláusulas if y else, especialmente si procesan muchos errores. Asegúrese de que el caso nominal esté claro.

● Para cadenas if-then-else y declaraciones de casos, elija un orden que maximice la legibilidad.

● Para atrapar errores, use la cláusula predeterminada en una declaración de caso o la última else en una cadena de declaraciones if-then-else.

● No todas las construcciones de control son iguales. Elija la construcción de control que sea más apropiada para cada sección de código.