17. Unusual Control Structures

Code Complete, cap. 17

Contents

17.1 Multiple Returns from a Routine

- Most languages support some means of exiting from a routine partway through the routine.
- The `return` and `exit` statements are control constructs that enable a program to exit from a routine at will.
- They cause the routine to terminate through the normal exit channel, returning control to the calling routine.
- The word `return` is used here as a generic term for return in C++ and Java, `Exit Sub` and `Exit Function` in Visual Basic, and similar constructs.

Here are guidelines for using the return statement:

Use a return when it enhances readability

In certain routines, once you know the answer, you want to return it to the calling routine immediately.
If the routine is defined in such a way that it doesn't require any further cleanup once it detects an error, not returning immediately means that you have to write more code.

Ejemplo: unidades checkFilas.py y checkColumnas.py

Use guard clauses (early returns or exits) to simplify complex error processing

Code that has to check for numerous error conditions before performing its nominal actions can result in deeply indented code and can obscure the nominal case, as shown here:

> Ejemplo: Visual Basic Code That Obscures the Nominal Case
>
> ```
> If file.validName() Then
>      If file.Open() Then
>           If encryptionKey.valid() Then
>                If file.Decrypt( encryptionKey ) Then
>                     This is the code for the nominal case:
>                     lots of code ...
>                End If
>           End If
>      End If
> End If
> ```
>
> Optimizado así mediante múltiples puntos de salida:
>
> ```
> ' set up, bailing out if errors are found
> If Not file.validName() Then Exit Sub
> If Not file.Open() Then Exit Sub
> If Not encryptionKey.valid() Then Exit Sub
> If Not file.Decrypt( encryptionKey ) Then Exit Sub
>
> ' lots of code
> ```

Returns

Fowler, Martin. Refactoring: Improving the Design of Existing Code, Reading, Mass.: Addison Wesley, 1999.

In the description of the refactoring called “Replace Nested Conditional with Guard Clauses,” Fowler suggests using multiple return statements from a routine to reduce nesting in a set of if statements.
Fowler argues that multiple returns are an appropriate means of achieving greater clarity, and that no harm arises from having multiple returns from a routine.