Code Complete 스터디
17 특이한 제어 구조들

DB검색플랫폼팀
조영일

17.1 여러 곳에서 리턴하는 루틴

- 리턴
  - C++, Java: return
  - VB: Exit Sub, Exit Function

- 필요하다면 여러 위치에서 리턴해도 됨

17.1 여러 곳에서 리턴하는 루틴

- 보다 읽기 쉬운 코드를 만들기 위해

```
Comparison Compare(int vale1, int value2) {
        if (value1 < value2) {
                return Comparison_LessThan;
        } else if (value1 > value2) {
                return Comparison_GreaterThan;
        }
        return Comparison_Equal;
}
```

17.1 여러 곳에서 리턴하는 루틴

- 복잡한 오류 처리를 단순화하기 위해

```
If Cond1 Then
    If Cond2 Then
        If Cond3 Then
            정상적인 경우에 실행되는 코드
        End If
    End If
End If
```

```
If Not Cond1 Then Exit Sub
If Not Cond2 Then Exit Sub
If Not Cond3 Then Exit Sub
정상적인 경우에 실행되는 코드
```

17.1 여러 곳에서 리턴하는 루틴

- 코드 하단을 읽고 있을 때 위쪽 어딘가에서 리턴되는지 모른다면 루틴을 이해하기 어려움
- return은 읽기 쉬운 코드가 만들어지는 경우에만 신중하게 사용하자!

17.2 재귀적 용법

- Recursion
  - 루틴이 자신을 호출하는 것
  - 문제를 작은 단위로 나눠서 해결할 때

```
void QuickSort(int firstIndex, int lastIndex, String [] names) {
        if (lastIndex > firstIndex) {
                int midPoint = Partition(firstIndex, lastIndex, names);
                QuickSort(firstIndex, midPoint - 1, names);
                QuickSort(midPoint, lastIndex, names);
        }
}
```

17.2 재귀적 용법

- 재귀 호출 사용 팁
  - 재귀 호출이 중단되는지 확인하라
  - 무한 재귀 호출을 막기 위해서 안전한 카운터를 사용하라
    - 포인터나 레퍼런스로 파라미터 전달
  - 한 루틴으로 재귀 호출을 제한하라
    - 순환호출은 위험함
  - 스택을 감시하라
    - 스택의 크기에 따라 카운터 한계값을 결정
    - 지역 변수 대신에 new로 동적 할당이 안전함
  - 팩토리얼이나 피보나치 수열을 계산하기 위해 사용하지 말 것
    - iteration이 더 바람직한 경우가 있음

17.3 goto 문

- goto 반대
  - Dijkstra,1968
    - 코드의 품질이 프로그래머가 사용한 goto 문 수에 반비례함
  - 논리적인 구조를 방해하고 컴파일러의 최적화에서 손해
  - 코드가 느려지고 크기가 줄어들지도 않음
  - Java같은 현대 언어에서는 아예 지원하지 않음

17.3 goto 문

- goto 찬성
  - 특정한 환경에서 신중하게 사용하자
  - 적절한 위치에 사용하면 코드 중복을 제거
  - 한 곳에서 몰아서 리소스 해제하기 좋음
  - Knuth, 1974
    - goto를 사용해서 유리한 케이스
    - goto를 사용해서 불리한 케이스
  - C++, VB, Ada와 같은 정성들여 만들어진 언어에서도 지원하고 있음

17.3 goto 문

- 오류 처리와 goto 문

```
' This routine purges a group of files.
Sub PurgeFiles( ByRef errorState As Error_Code )
   Dim fileIndex As Integer
   Dim fileToPurge As Data_File
   Dim fileList As File_List
   Dim numFilesToPurge As Integer

   MakePurgeFileList( fileList, numFilesToPurge )

   errorState = FileStatus_Success
   fileIndex = 0
   While ( fileIndex < numFilesToPurge )
      fileIndex = fileIndex + 1
      If Not ( FindFile( fileList( fileIndex ), fileToPurge ) ) Then
         errorState = FileStatus_FileFindError
         GoTo END_PROC
      End If

      If Not OpenFile( fileToPurge ) Then
         errorState = FileStatus_FileOpenError
         GoTo END_PROC
      End If

      If Not OverwriteFile( fileToPurge ) Then
         errorState = FileStatus_FileOverwriteError
         GoTo END_PROC
      End If

      if Erase( fileToPurge ) Then
         errorState = FileStatus_FileEraseError
         GoTo END_PROC
      End If

   Wend

END_PROC:
   DeletePurgeFileList( fileList, numFilesToPurge )
End Sub
```

Here's a GoTo.

Here's a GoTo.

Here's a GoTo.

Here's a GoTo.

Here's the GoTo *label.*

```
' This routine purges a group of files.
Sub PurgeFiles( ByRef errorState As Error_Code )
   Dim fileIndex As Integer
   Dim fileToPurge As Data_File
   Dim fileList As File_List
   Dim numFilesToPurge As Integer

   MakePurgeFileList( fileList, numFilesToPurge )

   errorState = FileStatus_Success
   fileIndex = 0
   While ( fileIndex < numFilesToPurge And errorState = FileStatus_Success )

      fileIndex = fileIndex + 1

      If FindFile( fileList( fileIndex ), fileToPurge ) Then
         If OpenFile( fileToPurge ) Then
            If OverwriteFile( fileToPurge ) Then
               If Not Erase( fileToPurge ) Then
                  errorState = FileStatus_FileEraseError
               End If
            Else ' couldn't overwrite file
               errorState = FileStatus_FileOverwriteError
            End If
         Else ' couldn't open file
            errorState = FileStatus_FileOpenError
         End If
      Else ' couldn't find file
         errorState = FileStatus_FileFindError
      End If
   Wend
   DeletePurgeFileList( fileList, numFilesToPurge )
End Sub
```

The While test has been changed to add a test for errorState.

This line is 13 lines away from the If *statement that invokes it.*

```
' This routine purges a group of files.
Sub PurgeFiles( ByRef errorState As Error_Code )
   Dim fileIndex As Integer
   Dim fileToPurge As Data_File
   Dim fileList As File_List
   Dim numFilesToPurge As Integer

   MakePurgeFileList( fileList, numFilesToPurge )

   errorState = FileStatus_Success
   fileIndex = 0
   While ( fileIndex < numFilesToPurge ) And ( errorState = FileStatus_Success )

      fileIndex = fileIndex + 1

      If Not FindFile( fileList( fileIndex ), fileToPurge ) Then
         errorState = FileStatus_FileFindError
      End If

      If ( errorState = FileStatus_Success ) Then
         If Not OpenFile( fileToPurge ) Then
            errorState = FileStatus_FileOpenError
         End If
      End If

      If ( errorState = FileStatus_Success ) Then
         If Not OverwriteFile( fileToPurge ) Then
            errorState = FileStatus_FileOverwriteError
         End If
      End If

      If ( errorState = FileStatus_Success ) Then
         If Not Erase( fileToPurge ) Then
            errorState = FileStatus_FileEraseError
         End If
      End If
   Wend
   DeletePurgeFileList( fileList, numFilesToPurge )
End Sub
```

The While test has been changed to add a test for errorState.

The status variable is tested.

The status variable is tested.

The status variable is tested.

```
' This routine purges a group of files. Exceptions are passed to the caller.
Sub PurgeFiles()
   Dim fileIndex As Integer
   Dim fileToPurge As Data_File
   Dim fileList As File_List
   Dim numFilesToPurge As Integer
   MakePurgeFileList( fileList, numFilesToPurge )
   Try
      fileIndex = 0
      While ( fileIndex < numFilesToPurge )
         fileIndex = fileIndex + 1
         FindFile( fileList( fileIndex ), fileToPurge )
         OpenFile( fileToPurge )
         OverwriteFile( fileToPurge )
         Erase( fileToPurge )
      Wend
   Finally
      DeletePurgeFileList( fileList, numFilesToPurge )
   End Try
End Sub
```

17.3 goto 문

- goto 문과 else 절에서 공유하는 코드
  - 생략

17.3 goto 문

- goto 문의 사용법 요약
  - "장화를 신고 있다면 흙탕물을 피하기 위해 돌아갈 필요는 없다."
  - 그러나...
    - 구조적인 제어 구현을 지원하지 않는 언어에서는 그걸 흉내내기 위해서 goto 문을 엄격하게 사용하라
    - goto 문과 동일한 구현이 가능하면 goto는 사용하지 마라
    - 효율성을 높이기 위해 사용되었다면 성능을 측정하고 필요하면 재작성하라
    - 구조적인 구현을 흉내내는 게 아니라면, 한 루틴에 한 번만, 앞으로 전진만...
    - 모든 레이블이 사용되었는지, 접근이 안 되는 코드가 있는지 확인하라

17.4 특이한 제어 구조에 대한 전망

- 한 번 쯤은...
  - 마음껏 goto 문 사용하기
  - goto 문의 목적지를 동적으로 계산하여...
  - 다른 루틴의 중간으로 goto
  - 줄 번호나 레이블에서 루틴 시작하기
  - 중략...