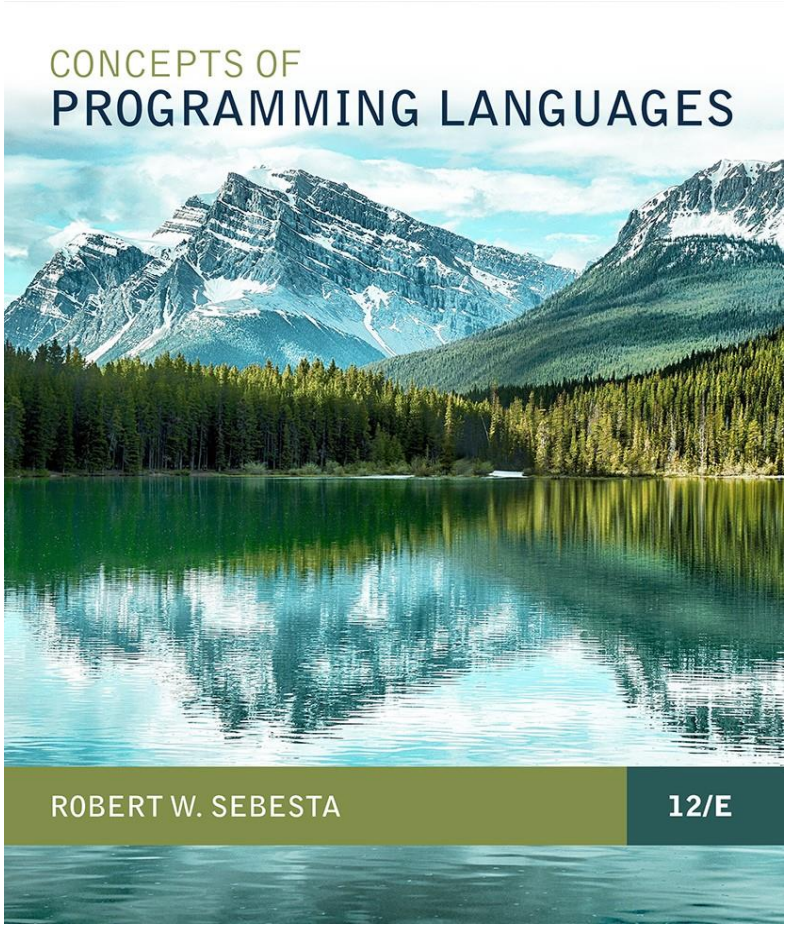
CAPÍTULO 8: STATEMENT-LEVEL CONTROL ESTRUCTURES
ESTRUCTURAS DE CONTROL A NIVEL DE SENTENCIA

Estructura de los lenguajes

Dr. Christian von Lücken

Ajustado al libro “Concepts of Programming Languages”, Robert Sebesta, 12/E. Pearson. 2018.
ISBN 0-321-49362-1



Capítulo 8 Tópicos

- Introducción
- Sentencias de Selección*
- Sentencias de Iteración*
- Salto no condicional
- Comandos de guarda
- Conclusiones

Introducción

- Una *estructura de control* es una sentencia de control y las sentencias cuya ejecución controla
- En FORTRAN I las sentencias de control estaban basadas directamente en el hardware de IBM 704

Sentencias de Control: Evolución

- Mucha investigación y argumento en los 60s trato sobre esa cuestión
  - Un resultado importante: se probo que todos los algoritmos representados por diagramas de flujo pueden ser codificados usando solo una selección de dos vías y los ciclos lógicos de *pretest*

Sentencias de Selección

- Una sentencia de selección provee los medios para seleccionar entre uno o más caminos de ejecución

- Dos categorías generales:
  - Selectores de dos vías
  - Selectores de múltiples vías

Sentencias de Selección de dos vías

- Forma general:

```
if control_expression
    then clause
    else clause
```

- Cuestiones de diseño:
  - Cuál es la forma y el tipo de la expresión que controla la selección?
  - Cómo se especifican las clausulas **`then`** y **`else`**?
  - Cómo debería especificarse el significado de selectores anidados?

Selección de dos vías: Ejemplos

- FORTRAN: **`IF`** (boolean_expr) statement
- Problema: puede seleccionar sólo una única sentencia; para seleccionar más de uno, debe ser utilizado un **`GOTO`**, cómo en el siguiente ejemplo

```
IF (.NOT. condition) GOTO 20
      ...
20 CONTINUE
```

- Lógica negativa es malo para la legibilidad
- Este problema fue resuelto en FORTRAN 77
- La mayoría de los lenguajes posteriores permiten combinaciones para la sentencia de selección en su selector de una vía

Selección de dos vías: Ejemplos

- ALGOL 60:

```
if (boolean_expr)
    then statement  (then clause)
    else statement  (else clause)
```

- Las sentencias pueden ser simples o compuestas

Selectores anidados

- Ejemplo de Java

```
if (sum == 0)
   if (count == 0)
       result = 0;
else result = 1;
```

- Cuál `if` contiene el `else`?
- La regla semántica estática de Java: `else` *matchea* con el `if` más cercano

Selectores anidados (continuación)

- Para forzar una semántica alternativa, pueden ser utilizadas sentencias agrupadas:

```
if (sum == 0) {
  if (count == 0)
      result = 0;
}
else result = 1;
```

- La solución anterior es utilizada en C, C++, y C#
- Otros lenguajes requieren que todas las clausulas then y else sean agrupadas

Sentencias de Selección Multiple

- Permite seleccionar una de varios grupos de sentencias
- Cuestiones de diseño:
  1. Cuál es la forma y el tipo de la expresión de control?
  2. Cómo especificar los segmentos seleccionables?
  3. El flujo de ejecución en la estructura esta restringida para incluir un único segmento de selección?
  4. Que se hace con respecto a los valores no representados?

Selección Multiple: Ejemplos

- Los primeros selectores múltiples:
  - FORTRAN arithmetic `IF` (selector three-way)

```
IF (arithmetic expression) N1, N2, N3
```

  - Los segmentos requieren `GOTO`s
  - No encapsulado (los segmentos seleccionables pueden estar en cualquier lugar)

Selección Multiple: Ejemplos

- Selectores múltiples modernos
  - Sentencia `switch` del C

```
switch (expression) {
        case const_expr_1: stmt_1;
        …
        case const_expr_n: stmt_n;
        [default: stmt_n+1]
}
```

Selección Multiple-Way: Ejemplos

- Opciones de diseño para la sentencia **`switch`** de C
  1. Las expresiones de control pueden ser sólo de tipo entero
  2. Los segmentos seleccionables pueden ser secuencias de sentencias, bloques o sentencias compuestas
  3. Cualquier número de segmentos puede ser ejeutado en una ejecución de la construcción (no existe un salto implicito al fin del segmento seleccionable)
  4. **`default`** se utiliza para valores no representados (si no hay **`default`**, la sentencia entera no hace nada)

Selección Multiple-Way: Ejemplos

- La sentencia `case` de Ada

```
case expression is
    when choice list => stmt_sequence;
    …
    when choice list => stmt_sequence;
    when others => stmt_sequence;]
end case;
```

- Más confiable que el `switch` de C (una vez que una secuencia de sentencias se termina, el control se pasa a la primera sentencia luego del `case`

Selección Multiple: Ejemplos usando **`if`**

- Los selectores múltiples puede aparecer como extenciones directas de los selectores de dos vías, usando clausulas else-if, por ejemplo en Ada:

```
if ...
  then ...
elsif ...
  then ...
elsif ...
  then ...
  else ...
end if
```

Sentencias de Iteración

- La ejecución repetida de una sentencia o sentencia agrupada se logra bien sea por medio de la iteración o la recursión
- Las cuestiones generales de diseño para las sentencias de control de iteración son:
  1. Cómo se controla la iteración?
  2. Dónde se ubica el mecanismo de control en la iteración?

Ciclos controlados por contadores

- Una sentencia iterativa con contador tiene una variable de ciclo o lazo
- Cuestiones de diseño:
  1. Cuál es el tipo y alcance de la variable de lazo?
  2. Cuál es el valor de la variable de lazo al finalizar el mismo?
  3. Es legítimo para la variable de lazo o parámetros de lazo que sean cambiados en el cuerpo del ciclo, y si es así altera el control?
  4. Deberían ser los parámetros de lazo evaluados, una vez o una vez para cada iteración

Sentencias de Iteración: Ejemplos

- Sintáxis de FORTRAN 90

```
DO label var = start, finish [, stepsize]
```

- Stepsize puede ser cualquier valor menos cero
- Los parámetros pueden ser expresiones
- Opciones de diseño:
  1. La variable de lazo debe ser **`INTEGER`**
  2. La variable de lazo siempre tiene su último valor
  3. La variable de lazo no puede cambiar su valor en el loop, pero si pueden los parámetros, como estos son evaluados una vez esto no afecta el control del loop
  4. Los parámetros de loop son evaluados sólo una vez, y este valor es usado para calcular el contador de iteraciones

Sentencias de Iteración: Ejemplos

- FORTRAN 95 : una segunda forma:

```
[name:] DO variable = initial, terminal [,stepsize]
                  …
END DO [name]
```

  - La variable de lazo debe ser **`INTEGER`**

```
Do count = 1, 10
 …
End do
```

Sentencias de Iteración

- Sentencia `for` de Pascal

```
for variable := initial (to|downto) final do
    statement
```

- Opciones de diseño:
  1. La variable de lazo debe ser un tipo ordinal
  2. Luego de la terminación normal la variable de lazo es indefinida
  3. La variable de lazo no puede ser cambiada en el lazo; los parámetros del loop pueden cambiar, pero son evaluados una sola vez, por lo que no afecta el control

Sentencias de Iteración: Ejemplos

- Ada

```
for var in [reverse] discrete_range loop
...
end loop
```

- Un rango discreto es un sub-rango de un tipo entero o enumeración
- El alcance de la variable de lazo es el lazo
- La variable de lazo esta implicitamente no declarada luego de la terminación del loop

Sentencias de Iteración: Ejemplos

- Sentencia `for` de C

```
for ([expr_1] ; [expr_2] ; [expr_3]) statement
```

- Las expresiones pueden ser sentencias completas, o inclusive secuencias separadas con comas
  - El valor de una expresión con múltiples sentencias es el calor de la última sentencia en la expresión
- No existe una variable de loop explícita
- Todo puede cambiar en el loop
- La primera expresión se evalúa una vez, pero las otras en cada iteración

Iteraciones controladas lógicamente

- El control de repetición se basa en un Boolean
- Cuestiones de diseño:
  - Pre-test o post-test?
  - Debería el loop controlado de manera lógica ser una caso especial del controlado por contados? Expresión más que un contador
- Forma general:

```
while (ctrl_expr)
  loop body
```

```
do
  loop body
while (ctrl_expr)
```

Iteraciones controladas lógicamente: Ejemplos

- Pascal separa las sentencias lógicamente controladas en pre-test y post-test (`while-do` y `repeat-until`)
- C y C++ también tienen ambos, pero la expresión de control para la versión post-test se trata como en el caso de pre-test (`while-do` y `do- while`)
- Java es como C, excepto que la expresión de control debe ser Boolean (y el cuerpo puede ser accedido sólo al comienzo, Java no tiene **`goto`**

Sentencias de Iteración: Logically-Controlled Loops: Ejemplos

- Ada tiene una versión pretest, pero no post-test
- FORTRAN 77 y 90 ninguno
- Perl tiene dos pre-test logical loops, `while` y `until`, pero no post-test

Sentencias de Iteración: User-Located Loop Control Mechanisms

- Algunas veces es conveniente para los programadores decidir la ubicación del control de lazo (distinto la parte superior/inferior del loop)
- Diseño simpre para loops simples (ej., `break`)
- Cuestiones de diseño para loops anidados
  1. Debería el condicional ser parte de la exit?
  2. Debería control ser transferible fuera de más de un loop?

Sentencias de Iteración: User-Located Loop Control Mechanisms `break` y `continue`

- C , C++, y Java: **`break`**
- Incondicional; para cualquier loop o **`switch`**; sólo un nivel
- Java y C# tienen la opción de etiquetar los `break` : el control se transfiere a la etiqueta
- Una alternativa: sentencia **`continue`**; salta el resto de la iteración pero no sale del loop

Sentencias de Iteración: Iteration Based on Data Structures

- Número de elementos en una estructura de datos que controla la iteración
- El mecanismo de control es la llamada a una función iteradora que retorna el siguiente elementos en un orden dado, hasta que no quedan elementos
- El **`for`** de C puede ser utilizado para contruir un iterador definido por el usuario:

```
for (p=root; p==NULL; traverse(p)){
}
```

Sentencias de Iteración: Iteration Based on Data Structures (continued)

- El foreach de C# itera en los elementos de arrays y otras colecciones:

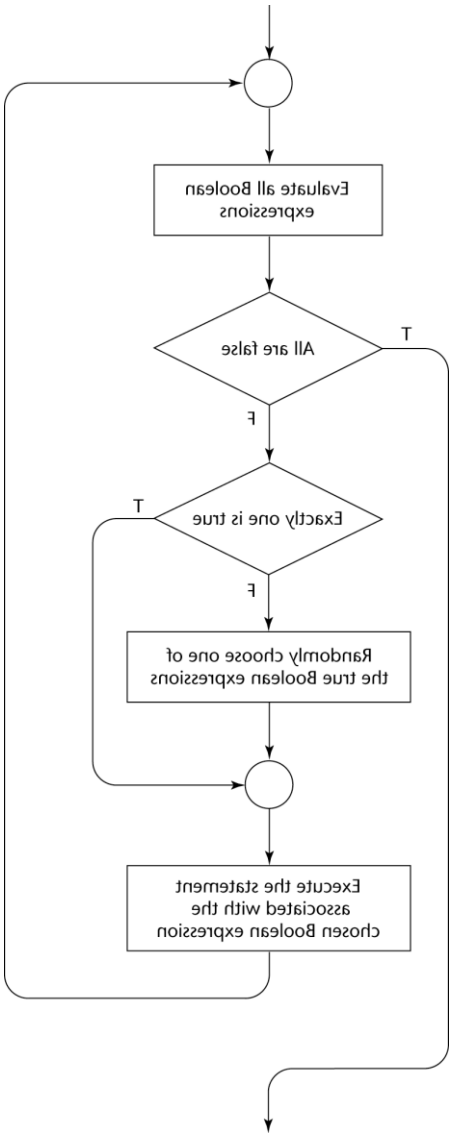
```
Strings[] = strList = {“Bob”, “Carol”, “Ted”};
foreach (Strings name in strList)
    Console.WriteLine (“Name: {0}”, name);
```

- La notación {0} indica la posición en la cadena a ser mostrada

Salto no condicional

- Transfiere el control de ejecución en un logar especificado en el programa
- Represento uno de los más acalorados debates en los 1960's y 1970's
- Un mecanismo bien conocido: `goto`
- Preocupación principal: Legibilidad
- Algunos lenguajes no soportan `goto` (ej. Module-2 y Java)
- C# ofrece la sentencia `goto` (puede ser utilizado en sentencias `switch`)

Loop Guarded Command: Illustrated



Conclusión

- Variedad de estructuras de control a nivel de sentencia
- Elección de sentencias de control más halla de la selección y los loops de pretest es una cuestion de compromiso entre el tamaño del lenguaje y la facilidad de escritura
- Los Lenguajes de programación funcional y lógicos tienen estructuras de control diferentes