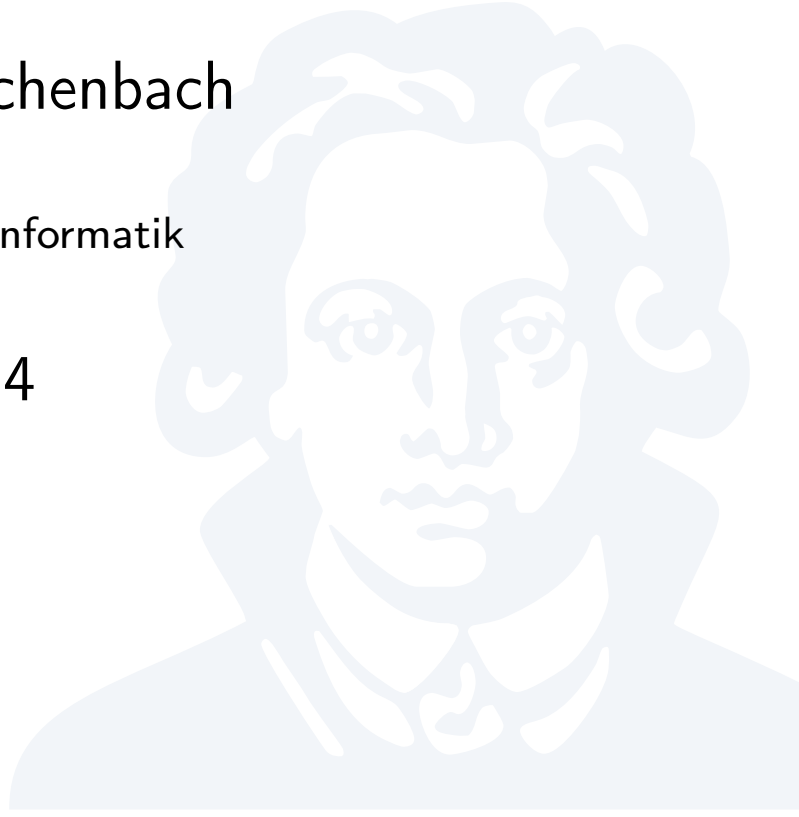
Foundations of Programming Languages

Iterative Control Structures

Prof. Dr. Christoph Reichenbach

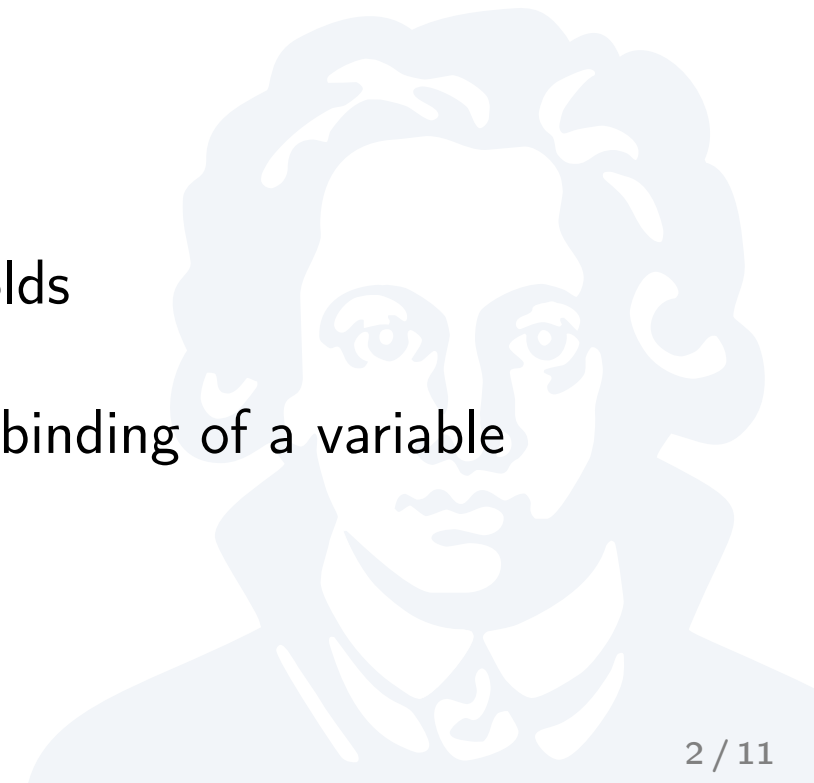
Fachbereich 12 / Institut für Informatik

29. Oktober 2014



Iterative Statements

- Better known as *loops*
- Execute same statement multiple times
- **Components**:
  - *body*: statement to repeat
  - Other components vary
- Variants:
  - *Logically controlled*:
    Execute *body* until predicate holds
  - *Variable controlled*:
    Execute *body* for each possible binding of a variable



Logically Controlled Loops

- **Components**:
  - *predicate (boolean expression)*
  - *body*
- **Variants:**
  - *Pre-test loop*: Check before executing
  - *Post-test loop*: Execute, then check

Pascal: **post-test**

```
repeat (* smallest x!>10 *)
    x := x + 1;
    fact := fact * x;
until fact > 10;
```

C: **pre-test**

```
while (b != 0) {
    int c = b;
    b = a % b;  // modulo
    a = b;
} // a <- gcd(a, b)
```

Semantics of Logically Controlled Loops

$$
\begin{array}{rcl}
\mathit{stmt} & ::= & \varepsilon \\
& | & \langle \mathit{stmt} \rangle \text{`;'} \langle \mathit{stmt} \rangle \\
& | & \text{`}\texttt{while}\text{'} \langle \mathit{expr} \rangle \text{`}\texttt{do}\text{'} \langle \mathit{stmt} \rangle \\
& | & \text{`}\texttt{repeat}\text{'} \langle \mathit{stmt} \rangle \text{`}\texttt{until}\text{'} \langle \mathit{expr} \rangle
\end{array}
$$

$$
\frac{\langle s_1 | \sigma \rangle \longrightarrow^{\star} \langle \varepsilon | \sigma' \rangle}{\langle s_1 \ ; \ s_2 | \sigma \rangle \longrightarrow \langle s_2 | \sigma' \rangle} \ (\mathit{seq})
$$

$$
\frac{\langle e | \sigma \rangle \longrightarrow^{\star} \langle \texttt{true} | \sigma' \rangle}{\langle \texttt{while } e \texttt{ do } s | \sigma \rangle \longrightarrow \langle s \ ; \ \texttt{while } e \texttt{ do } s | \sigma' \rangle} \ (\mathit{pre\text{-}true})
$$

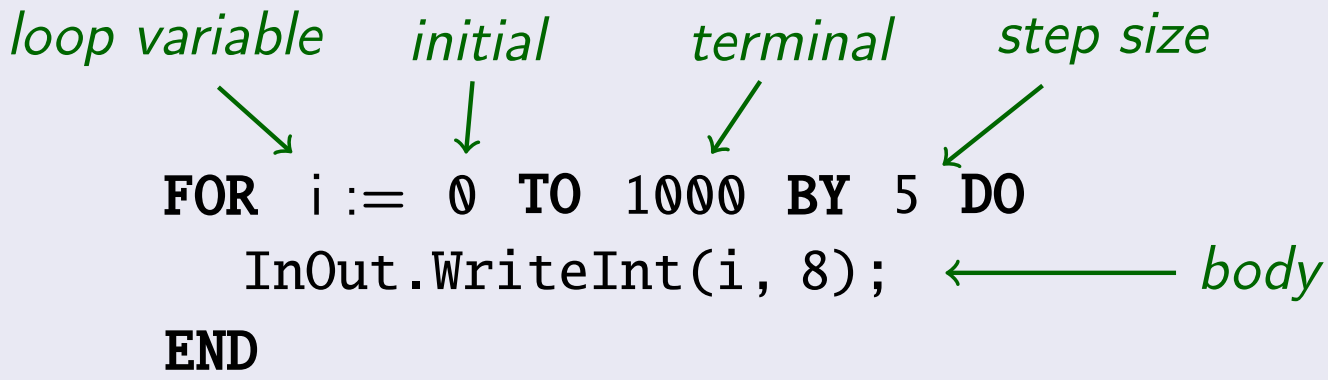
$$
\frac{\langle e | \sigma \rangle \longrightarrow^{\star} \langle \texttt{false} | \sigma' \rangle}{\langle \texttt{while } e \texttt{ do } s | \sigma \rangle \longrightarrow \langle \varepsilon | \sigma' \rangle} \ (\mathit{pre\text{-}false})
$$

$$
\frac{}{\langle \texttt{repeat } s \texttt{ until } e | \sigma \rangle \longrightarrow \langle s \ ; \ \texttt{while } \textbf{not } e \texttt{ do } s | \sigma \rangle} \ (\mathit{post})
$$

Variable-Controlled Loops

- **Components**:
  - *body*
  - *loop variable*
  - *loop parameters* (what to iterate over)
- **Variants**:
  - Integer range:
    - *initial*, *terminal*, (optional: *step size*)
    - Count up or down?
  - Data structure range
  - Variable bindings: can be reassigned?

Modula-2



Variable-Controlled Loops: Examples

Fortran

```
   do i=1,10,2
      i = 7 ! error: no write permission    ...
   end do
i ← 1, 3, 5, 7, 9
```

Ada

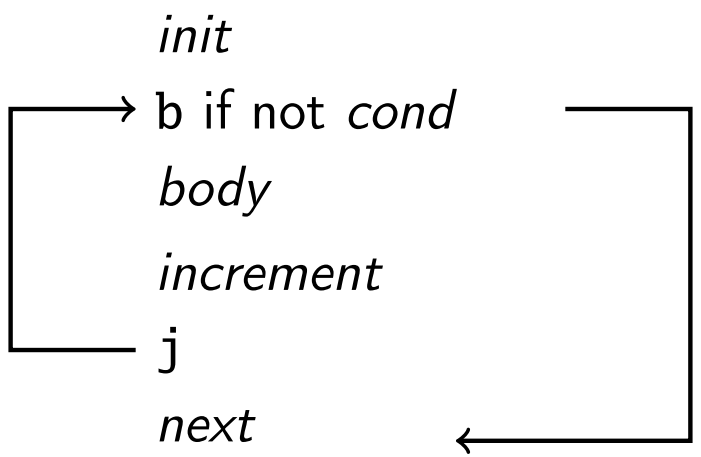
```
   for i in reverse 1..10 loop
      i := 7; -- error: no write permission    ...
   end loop;
i ← 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

C family

```
   for (int i = 0; i < 10; i += 2) {
      i = 7; // permitted
   }
```

The C family `for` loop

```
main()
{
  for (init; cond; increment)
    body
  next
```



Iteration over Data Structures

- Loop variable bound to elements of data structure:

Python

```
for i in set:
    print i
```

Ruby

```
set.each do |i|
    puts i
end
```

Java

```
for (int i : set) {
    System.out.println(i);
}
```

Iteration order defined in data types

User-Located Loop Control

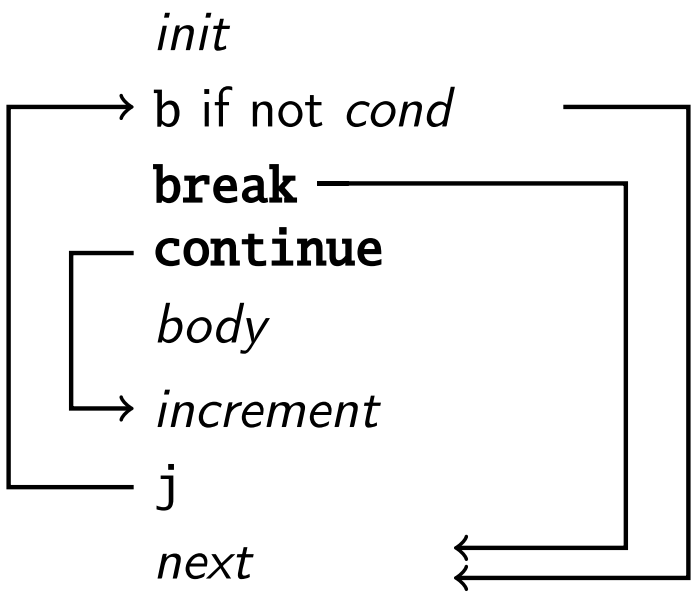
- Explicitly control loop behaviour:
  - terminate loop (C family: **break**)
  - skip forward (C family: **continue**)
- Affect *innermost loop*
- Access to outer loops via naming (e.g., in Java)

Java

```
for (int i : set) {
   if (i = x) {
      break; // found x, we're done
   } else if (i < 0) {
      continue; // don't print negative numbers
   }
   System.out.println(i);
}
```

User-Located Loop Control, C family

```
main()
{
  for (init; cond; increment)
    body
  next
```



Summary

- Iterative statements:
  - Repeated execution of *loop body*
  - *Logically controlled*: repeat while/until predicate P
  - *Variable controlled*: repeat for each value in set
    - Over integer ranges
    - Over container types
- User-located loop control:
  - Skip forward to next loop interation
  - Break out of loop