Hochschule für Technik Stuttgart

Concepts of Programming Languages

5th Week

Control Structures and Statements

Prof. Dr. Peter Heusch

Control Structures

- A control structure is a control statement and the statements whose execution it controls
- Allows non-linear flow of code execution

Types of Statements

- Selection Statements:
    - If
    - Switch/Case (multiway selectors)
- Iterative Statements
    - While
    - For
- Unconditional Branching
    - Goto
    - Break/Continue
- Guarded Commands

Goto - a BASIC Sample

```
10 let guess = random(100)
20 print "Please input a number between 1
and 100"
30 read a
40 if a = guess then goto 70
50 if a < guess then print "Too small";
   goto 30
60 if a > guess then print "Too large";
   goto 30
70 print "Congratulations, you got it“
```

The First Developments

- FORTRAN I control statements were based directly on IBM 704 hardware: language design by hardware design
- Research in the 1960s found (Böhm&Jacopini `66):
  - Sequence, selection, repetition are needed for Turing completenes
  - GOTO with selection condition is sufficient (but error-prone)
  - All computable functions can be coded with only two-way selection and pretest logical loops: **If/Else** and **While**

Types of Statements

- Selection Statements:
    - If
    - Switch/Case
- Iterative Statements
    - While
    - For
- Unconditional Branching
    - Goto
    - Break/Continue
- Guarded Commands

Selection Statements

- Selection statements allow the conditional execution of other statements depending on certain values
- Selection statements can be subdivided into two groups:
  - Two-way selectors
  - Multiple-way selectors
- Especially the multiple-way selectors show a lot of variation across programming languages

Two-Way Selection

- General form:

```
  if control_expression
  then statement
  else statement
```

- Design Issues:
  - What is the form and type of the control expression?
  - How are the then and else clauses specified?
  - How should the meaning of nested selectors be specified?

Case Study: FORTRAN

- Before the appearance of Fortran 77, `if` could only control a single elementary statement:

  ```
  if (x .ne. 0) y = 2/x
  ```

- Moreover there was no else statement
- If a single statement was not enough, a `goto` had to be used
- In order to implement an `if-then-else` programmers had to use one `if` and two `gotos`

If-Then-Else in FORTRAN

```
10 if ( x .ne. 0 ) goto 20
   print *, X, ' is zero'
   goto 30
20 X = -X
30 print *, 'On we go‘
```

- Or, using arithmetic if (`if(expr) neg,zero,pos`):

```
if ( x ) 20, 30, 20
20 X = -X
   goto 40
30 print *, X, ' is zero'
40 print *, 'On we go'
```

Case Study: COBOL

Cobol allows the definition of multivalued Boolean expressions:

```
01 Punktzahl PIC 99 VALUE 0.
    88 Mangelhaft VALUE 0 THRU 50.
    88 Ausreichend VALUE 51 THRU 68.
    88 Befriedigend VALUE 69 THRU 80.
    88 Gut VALUE 81 THRU 91.
    88 SehrGut VALUE 92 THRU 99.
If Mangelhaft THEN ....
```

Case Study: C, C++, Java, Pascal

- In C and C++ the condition can be anything that is an expression:
    - Integer values: 0 or not 0
    - Floating point values: 0.0 or not 0.0
    - Pointer values: NULL or not NULL
- In Java & Pascal conditions have to be Boolean expressions

Two-Way Selection

- General form:

```
if control_expression
then statement
else statement
```

- Design Issues:
  - What is the form and type of the control expression?
  - How are the then and else clauses specified?
  - How should the meaning of nested selectors be specified?

Nesting Selectors

- Java example

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

- Which `if` gets the `else`?
- Java's static semantics rule: `else` matches with the nearest unmatched `if`
- Python's static semantics rule: `else` matches with the `if` at same level of indentation

The Dangling Else

- To force an alternative semantics, compound statements (blocks) may be used:

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

- The above solution is used in C, C++, and C#
- Perl requires all `then` and `else` clauses to be compound

Types of Statements

- Selection Statements:
  - If
  - Switch/Case (multiway selectors)
- Iterative Statements
  - While
  - For
- Unconditional Branching
  - Goto
  - Break/Continue
- Guarded Commands

Multiway-Selectors

- Allow the selection of one of any number of statements or statement groups
- Design Issues:
  - What is the form and type of the control expression?
  - How are the selectable segments specified?
  - Is execution flow through the structure restricted to include just a single selectable segment?
  - What is done about unrepresented expression values?

Multiway-Selector Examples

- FORTRAN: The arithmetic IF
- C & Successors: The switch/case statement:

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

- In C, a case „falls through“ to the next case unless an explicit break is given

Multiple Entry Points

Design question for all control structures:

- Should a control structure have multiple entries?
- „Entry (point)“ means the place in the code where execution begins
- FORTRAN, PL/I and C have the keyword „entry“ allowing multiple entries for the same procedure
- This feature was „almost never“ implemented because the result ist a software engineering nightmare

More Multiway Selector Examples

- Pascal's case/of resembles, C's switch/case but with some differences:
    - A case can select multiple values:

      ```
      case expression of:
      when a...b: begin...end
      when c,d,e: begin..end
      ```

    - If a case does not match any value (and no other clause is given), this is considered an error
- Cases not fall through to the next one, this enhances readability

Multiway-Selector with if/elif/else

- Python does not have any explicit multiway-selector, but uses if/elif/else:

```
if expr 1:
   stmt block 1
elif expr n:
   stmt block n
...
else:
   stmt block n+1
```

- Advantage over C, C++: expressions can be any values, not only constants

Types of Statements

- Selection Statements:
    - If
    - Switch/Case
- Iterative Statements
    - While
    - For
- Unconditional Branching
    - Goto
    - Break/Continue
- Guarded Commands
-

Iterative Statements

- The repeated execution of a statement or compound statement can be accomplished:
    - By iteration: while, for, etc.
    - By recursion
- General design issues for iteration control statements:
    1. How is iteration controlled? Logic or counting?
    2. Where is the control mechanism in the loop? Pretest or posttest?

Counting Loops

- A counting iterative statement has a **loop variable**, a means of **specifying the initial and terminal**, and **stepsize values**
- Design Issues:
  - What are the type and scope of the loop variable?
  - Access to the loop variable after loop termination?
  - Can the loop body change the loop variable?
  - Should the loop parameters be evaluated only once, or once for every iteration?

Counting Loops: FORTRAN 90

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

END DO
```

- Stepsize can be any value but zero
- Parameters can be expressions
- Design choices:
  1. Loop variable must be INTEGER
  2. Loop variable always has its last value
  3. The loop variable cannot be changed in the loop, but the parameters can; because they are evaluated only once, it does not affect loop control

Counting Loops: Pascal

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

- Design choices:
    - Loop variable must be an ordinal type
    - After normal termination, loop variable is undefined
    - The loop variable cannot be changed in the loop; the loop parameters can be changed, but they are evaluated just once, so it does not affect loop control

Counting Loops: Ada

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

- A discrete range is a sub-range of an integer or enumeration type
- Scope of the loop variable is the range of the loop
- Loop variable is implicitly undeclared after loop termination

Counting Loops: C

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

- The expressions can be whole statements, or even comma separated statement sequences
- The value of a multiple-statement expression is the value of the last statement in the expression
- There is no explicit loop variable
- Everything can be changed in the loop
- The first expression is evaluated once, but the other two are evaluated with each iteration

Types of Statements

- Selection Statements:
    - If
    - Switch/Case
- Iterative Statements
    - While
    - For
- Unconditional Branching
    - Goto
    - Break/Continue
- Guarded Commands