



*COSC252: Programming Languages:*

*Basic Semantics: Environments,  
Names, Literals, Scope, ...*

Jeremy Bolton, PhD

Asst Teaching Professor

GEORGETOWN  
UNIVERSITY

# *Chapter 8 Topics*

- Introduction
- Selection Statements
- Iterative Statements
- Unconditional Branching
- Guarded Commands
- Conclusions

# *Control Statements: Evolution*

- Much research and argument in the 1960s about the issue
  - One important result: It was proven that all algorithms represented by flowcharts can be coded with only two-way selection and pretest logical loops

# *Control Structure*

- *A control structure* is a control statement and the statements whose execution it controls
- Design question
  - Should a control structure have multiple entries?

# *Selection Statements*

- *A selection statement* provides the means of choosing between two or more paths of execution
- Two general categories:
  - Two-way selectors
  - Multiple-way selectors

# *Two-Way Selection Statements*

- General form:

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

- 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?

# *The Control Expression*

- If the then reserved word or some other syntactic marker is not used to introduce the then clause, the control expression is placed in parentheses
- In C89, C99, Python, and C++, the control expression can be arithmetic
- In most other languages, the control expression must be Boolean

# Clause Form

- In many contemporary languages, the then and else clauses can be single statements or compound statements
- In Perl, all clauses must be delimited by braces (they must be compound)
- In Python and Ruby, clauses are statement sequences
- Python uses indentation to define clauses

```
if x > y :  
    x = y  
    print "x was greater than y"
```



# *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 previous `if`

## *Nesting Selectors (continued)*

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

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

- The above solution is used in C, C++, and C#

## *Nesting Selectors (continued)*

- Python

```
if sum == 0 :  
    if count == 0 :  
        result = 0  
else :  
    result = 1
```

# *Multiple-Way Selection Statements*

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

# *Multiple-Way Selection: Examples*

- C, C++, Java, and JavaScript

```
switch (expression) {  
    case const_expr1: stmt1;  
    ...  
    case const_exprn: stmtn;  
    [default: stmtn+1]  
}
```

# *Multiple-Way Selection: Examples*

- Design choices for C's **switch** statement
  1. Control expression can be only an integer type
  2. Selectable segments can be statement sequences, blocks, or compound statements
  3. Any number of segments can be executed in one execution of the construct (*there is no implicit branch at the end of selectable segments*)
  4. **default** clause is for unrepresented values (if there is no **default**, the whole statement does nothing)

# *Implementing Multiple Selectors*

- Approaches:
  - Multiple conditional branches
  - Store case values in a table and use a linear search of the table
  - When there are more than ten cases, a hash table of case values can be used
  - If the number of cases is small and more than half of the whole range of case values are represented, an array whose indices are the case values and whose values are the case labels can be used

## *Multiple-Way Selection Using **if***

- Multiple Selectors can appear as direct extensions to two-way selectors, using else-if clauses, for example in Python:

```
if count < 10 :  
    bag1 = True  
elif count < 100 :  
    bag2 = True  
elif count < 1000 :  
    bag3 = True
```



## *Multiple-Way Selection Using **if***

- The Python example can be written as a Ruby `case`

`case`

```
  when count < 10 then bag1 = true
```

```
  when count < 100 then bag2 = true
```

```
  when count < 1000 then bag3 = true
```

`end`

# *Iterative Statements*

- The repeated execution of a statement or compound statement is accomplished either by iteration or recursion
- General design issues for iteration control statements:
  1. How is iteration controlled?
  2. Where is the control mechanism in the loop?

# Counter-Controlled Loops

- A counting iterative statement has a loop variable, and a means of specifying the *initial* and *terminal*, and *stepsize* values
- Design Issues:
  1. What are the type and scope of the loop variable?
  2. Should it be legal for the loop variable or loop parameters to be changed in the loop body, and if so, does the change affect loop control?
  3. Should the loop parameters be evaluated only once, or once for every iteration?

# Counter-Controlled Loops: Examples

- C-based languages

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

- The expressions can be whole statements, or even statement sequences, with the statements separated by commas

- The value of a multiple-statement expression is the value of the last statement in the expression
- If the second expression is absent, it is an infinite loop

- Design choices:

- 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
- It is legal to branch into the body of a for loop in C

# Counter-Controlled Loops: Examples

- Python

- `for loop_variable in object:`

- loop body

- `[else:`

- else clause]

- The object is often a range, which is either a list of values in brackets (`[2, 4, 6]`), or a call to the range function (`range(5)`), which returns `0, 1, 2, 3, 4`
  - The loop variable takes on the values specified in the given range, one for each iteration
  - The else clause, which is optional, is executed if the loop terminates normally

# *Logically-Controlled Loops*

- Repetition control is based on a Boolean expression
- Design issues:
  - Pretest or posttest?
  - Should the logically controlled loop be a special case of the counting loop statement or a separate statement?

## *Logically-Controlled Loops: Examples*

- C and C++ have both pretest and posttest forms, in which the control expression can be arithmetic.
- Java is like C and C++, except the control expression must be Boolean (and the body can only be entered at the beginning -- Java has no `goto`)

## *User-Located Loop Control Mechanisms*

- C , C++, Python, Ruby, and C# have unconditional unlabeled exits (**break**)
- Java and Perl have unconditional labeled exits (**break** in Java)
- C, C++, and Python have an unlabeled control statement, **continue**, that skips the remainder of the current iteration, but does not exit the loop
- Java and Perl have labeled versions of **continue**



# *Unconditional Branching*

- Transfers execution control to a specified place in the program
- Represented one of the most heated debates in 1960's and 1970's
- Major concern: Readability
- Some languages do not support `goto` statement (e.g., Java)
- C# offers `goto` statement (can be used in `switch` statements)
- Loop exit statements are restricted and somewhat camouflaged `goto`'s