Ch7. Expressions & Assignment Statements

- Arithmetic Expressions
- Overloaded Operators
- Type Conversions
- Relational and Boolean Expressions
- Short-Circuit Evaluation
- Assignment Statements
- Mixed-Mode Assignment

This lecture covers review questions 7-16 of Chapter 7

Dr. Amirhossein Chinai, ECE, UPRM Spring 2010

***Some slides are adapted from the Sebesta's textbook

Relational Expressions

- Relational Expressions
  - Use relational operators and operands of various types
  - Evaluate to some Boolean representation
  - Operator symbols used vary somewhat among languages (!=, /=, -=, .NE., <>, #)

- JavaScript and PHP have two additional relational operator, === and !==
  - Similar to their cousins, == and !=, except that they do not coerce their operands

Boolean Expressions

- Boolean Expressions
  - Operands are Boolean and the result is Boolean
  - Example operators

<table>
<thead>
<tr>
<th>FORTRAN 77</th>
<th>FORTRAN 90</th>
<th>C</th>
<th>Ada</th>
</tr>
</thead>
<tbody>
<tr>
<td>.AND.</td>
<td>and</td>
<td>&amp;&amp;</td>
<td>and</td>
</tr>
<tr>
<td>.OR.</td>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.NOT.</td>
<td>not</td>
<td>!</td>
<td>not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xor</td>
<td></td>
</tr>
</tbody>
</table>

- C89 has no Boolean type—it uses int type with 0 for false and nonzero for true
- One odd characteristic of C's expressions: a < b < c is a legal expression, but the result is not what you might expect:
  - Left operator is evaluated, producing 0 or 1
  - The evaluation result is then compared with the third operand (i.e., c)
Short Circuit Evaluation

• An expression in which the result is determined without evaluating all of the operands and/or operators
• Example: \((13 \ast a) \ast (b/13 - 1)\)
  If \(a\) is zero, there is no need to evaluate \((b/13 - 1)\)
• Problem with non-short-circuit evaluation
  
  ```
  index = 1;
  while (index <= length) \&\& (LIST[index] \neq value)
    index++;
  ```
  When \(index=\text{length}\), \(\text{LIST}[\text{index}]\) will cause an indexing problem (assuming \(\text{LIST}\) has \(\text{length} - 1\) elements)

Assignment Statements

• The general syntax
  
  `<target_var> <assign_operator> <expression>`

• The assignment operator
  
  - \(\text{FORTRAN, BASIC, the C-based languages}\)
  - \(\text{:= ALGOLs, Pascal, Ada}\)
• = can be bad when it is overloaded for the relational operator for equality
  
  - (that's why the C-based languages use \(--\) as the relational operator)
Assignment Statements: **Compound Operators**

- A shorthand method of specifying a commonly needed form of assignment
- Introduced in **ALGOL**
- Example
  
  \[ a = a + b \]
  
is written as
  
  \[ a += b \]

**Unary assignment operators in C-based**

- languages combine increment and decrement operations with assignment
- Examples
  
  \[ sum = ++count \]
  \[ sum = count++ \]
  \[ -count++ \] (count incremented then negated)

**Assignment as an Expression**

- In **C, C++, and Java**, the assignment statement produces a result and can be used as operands
- An example:
  
  ```c
  while ((ch = getchar()) != EOF){ ... } 
  ch = getchar() is carried out; the result (assigned to ch) is used as a conditional value for the while statement
  ```

- List Assignments
  
  - E.g. in **Perl** and **Ruby**
    
    \((\$first, \$second, \$third) = (20, 30, 40)\);

**Mixed-Mode Assignment**

- Assignment statements can also be mixed-mode
- In **Fortran, C**, and **C++**, any numeric type value can be assigned to any numeric type variable
- In **Java**, only widening assignment coercions are done
- In **Ada**, there is no assignment coercion
Summary

- Expressions
- Operator precedence and associativity
- Operator overloading
- Mixed-type expressions
- Various forms of assignment

Levels of Control Flow

- Within expressions (Chapter 7)
- Among program units (Chapter 9)
- Among program statements (this chapter)

Control Statements: Evolution

- **FORTRAN I** control statements were based directly on IBM 704 hardware
- 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?
- A selection statement is a control statement that 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 languages such as Ada, Java, Ruby, and C#, 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 Fortran 95, Ada, and Ruby, clauses are statement sequences.
- Python uses indentation to define clauses.
  ```python
  if x > y :
      x = y
      print "case 1"
  ```
Nesting Selectors

- **Java** example
  ```java
  if (sum == 0)
    if (count == 0)
      result = 0;
    else result = 1;
  else result = 1;
  ```
- Which *if* gets the *else*?
- **Java’s static semantics rule:**
  Else matches with the nearest unmatched *if*

Nesting Selectors (cont’d)

- To force an alternative semantics, compound statements may be used:
  ```java
  if (sum == 0) {
    if (count == 0)
      result = 0;
    else result = 1;
  }
  else result = 1;
  ```
- The above solution is used in C, C++, and C#
- **Perl** requires that all then and else clauses to be compound

Nesting Selectors (cont’d)

- Statement sequences as clauses: **Ruby**
  ```ruby
  if sum == 0 then
    if count == 0 then
      result = 0
    else
      result = 1
    end
  end
  ```

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++, and Java**
  ```
  switch (expression) {
    case const_expr_1: stmt_1;
    ...
    case const_expr_n: stmt_n;
    [default: stmt_n+1]
  }
  ```

Multiple-Way Selection: Examples

- **C#**
  - Differs from C in that it has a **static semantics** rule that disallows the implicit execution of more than one segment
  - Each selectable segment must end with an unconditional branch (goto or break)
  - Also, in C# the control expression and the case constants can be strings

Multiple-Way Selection: Examples

- **Ada**
  ```
  case expression is
    when choice list => stmt_sequence;
    ...
    when choice list => stmt_sequence;
    when others => stmt_sequence;
  end case;
  ```

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, compound statements, or blocks
  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 for unrepresented values)

- **More reliable than C’s switch** (once a stmt_sequence execution is completed, control is passed to the first statement after the case statement)**
Multiple-Way Selection: **Examples**

- **Ada design choices:**
  1. Expression can be any ordinal type
  2. Segments can be single or compound
  3. Only one segment can be executed per execution of the construct

- **Constant List Forms:**
  1. A list of constants
  2. Can include:
     - Subranges
     - Boolean OR operators (|

Ruby has two forms of case statements

1. One form is a compound assignment
   ```ruby
   leap = case
     when year % 400 == 0 then true
     when year % 100 == 0 then false
     else year % 4 == 0
   end
   ```

2. The other uses a case value and when values
   ```ruby
   case in_val
     when -1 then neg_count++
     when 0 then zero_count++
     when 1 then pos_count++
     else puts "Error - in_val is out of range"
   end
   ```

Multiple-Way Selection Using **if**

- Multiple Selectors can appear as direct extensions to two-way selectors, using else-if clauses, for example in **Python**:
  ```python
  if count < 10 :
    bag1 = True
  elif count < 100 :
    bag2 = True
  elif count < 1000 :
    bag3 = True
  ```

The **Python** example can be written as a **Ruby** case

```ruby
case
  when count < 10 then bag1 = true
  when count < 100 then bag2 = true
  when count < 1000 then bag3 = true
end
```