
CS 241

Control Structures

Christopher A. Gantz

SPS Undergraduate Program
Regis University
cgantz@regis.edu

Lecture Topic #9

CS 241: Nesting and Extended IF Statements

Christopher A. Gantz
School of Professional Studies
Regis University
cgantz@regis.edu

Overview

- Reading
 - Nance textbook Pages 194-226
- Nested and Extended IF Statements
 - Nested IF statements
 - Extended IF statements
 - Sequential Conditional IF statements
 - CASE statements
 - Assertions

Nested IF Statements

- The IF... THEN ... ELSE statement implements a two way selection
- What if a many way selection is required?
- The many way selection can be implemented with the following three construct variations
 - Nested IF statements
 - Extended IF statements
 - Sequential conditional statements

Nested IF statement

- Multi/many way selection is implemented with IF ... THEN ... ELSE statements utilized in a nested structural pattern/configuration
 - IF <Boolean Expression 1> THEN
 - IF <Boolean Expression 2> THEN
 - <Statement>
 - ELSE
 - <Statement>
 - ELSE
 - <Statement>

Extended IF statement

- Multi/many way selection can also be implemented with another variational form of IF ... THEN ... ELSE statements utilized in a nested structural pattern/configuration
 - IF <Boolean Expression 1> THEN
 - <Statement> { Action 1 }
 - ELSE IF <Boolean Expression 2> THEN
 - <Statement> { Action 2 }
 - ELSE
 - <Statement> { Action n }

Extended IF statement (Cont.)

- See Example 5.12 on page 195 of Nance text
- Simple usage/best practice implementation rules for extended IF statement
 - The complex computational section should be implemented in after the THEN part of the construct
 - Nest the more simpler computation in the ELSE part of the construct
 - Extended IF form should be deployed when ever several possible actions can be considered sequentially

Matching ELSE Rule

- The rule of matching ELSE statements within a nested selection statement
- “ELSE clauses are matched/paired with the most recent THEN clause that has NOT yet been matched
- See pages 198 & 199 of Nance text
- Violation of this rule is a common cause of logical or semantic errors in program source code

Sequential conditional Statements

- IF <Boolean Expression 1> THEN
- <Statement 1> { Action 1 }
- IF <Boolean Expression 2> THEN
- <Statement 2> { Action 2 }
- IF <Boolean Expression 3> THEN
- <Statement 3> { Action 3 }

Sequential conditional Statements (Cont.)

- Caveat: Sequential conditional statements are less efficient because each IF ... THEN statement is executed each time through the program
- Sequential conditional statements are useful for specifying situations where multiple conditions may apply as opposed to a single condition
- Note: example 5.13 on page 197 of Nance text is a concrete example of a pre-condition

CASE statement

- An Alternative method for implementing a specific kind of multi/many was selection
- Form & Syntax
 - CASE <selector> OF
 - <label list 1>: <statement 1>
 - <label list 2>: <statement 2>
 - <label list n>: <statement n>
 - END { of CASE <selector> }

CASE statement (Cont.)

- Useful when several options depend on a specific value of a variable or expression
- Execution flow of a CASE Statement
 - Value of <selector> is determined
 - Value is matched with the exact label
 - Statement associated with label matched is executed
 - Control is passed to the statement directly after END

Selector/Label constraints/rules

- `<selector>` can be a value from any data type except a real data type
- `<selector>` can really only be an ordinal type
- Multiple labels with the same action statement can be separated by commas
- All possible values of the `<selector>` can only appear once within a CASE Statement label list
-

Selector/Label constraints/rules

- A program comment should note the end of a CASE statement
 - E.g. `END { of CASE <selector> }`
- A statement/action within a CASE statement can be a compound statement
- Note: Example 5.16 on page 211 is yet another example of a pre-condition

CASE statement OTHERWISE option

- Note: Only available on some Pascal versions
- Form & Syntax
 - CASE <selector> OF
 - <label list 1>: <statement 1>
 - <label list 2>: <statement 2>
 - <label list n>: <statement n>
 - OTHERWISE
 - <statement 1>
 - <statement n>
 - END { of CASE <selector> }

CASE statement OTHERWISE option

- Useful for
 - Specifying the same action for several values
 - Specifying a default action
 - Protects against out of range value's for the Case <selector>
 - CASE statements can be “sometimes” be utilized in place of an extended IF implementation construct

Assertions

- An assertion is simply a Boolean expression that defines an expected valid state space
- An assertion can also be implemented as simply a program comment
- Where this comment is a statement regarding what should be “true” about the state of a program a specific point during execution
 - E.g. { Assertion: NumberOf Students \neq 0 }
 - ClassAverage := SumOfScores /

Assertions (Cont.)

- Specifically named assertions
- Pre-Condition
 - Assertion specified before a statement or section of code
- Post-Condition
 - Assertion specified after a statement or section of code
- Invariant?
- See example on page 218 of Nance text