

---

CS 241  
Control Structures

Christopher A. Gantz

SPS Undergraduate Program  
Regis University  
cgantz@regis.edu

---

# Lecture Topic #7

## CS 241: Selection Statements

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

# Overview

---

- Reading
  - Nance textbook Pages 165-193
- Selection Statements
  - Boolean Expressions
  - Relational & logical Operators
  - IF ... THEN statements
  - IF ... THEN ... ELSE statements
  - Robust Program Development

# Selection Statement Overview

---

- Selection statements allow Pascal programmers to make decision about what sections of code are executed at run-time
- Selection Statement are a specific type of sequential execution control flow structures
- Control flow is simply the concept of having the ability to control the execution of sequential program statements

# Selection Statement Overview (Cont.)

---

- In order for a language construct to make a decision a special type of expressions must be defined
- These decision related expressions are referred to as Boolean expressions
- A Boolean expression is composed of boolean operands and a boolean operator

# Boolean Expressions

---

- Boolean operands are typically variables of data type boolean
  - E.g. VAR
  - Bit : boolean; { <variable name> : boolean; }
- The Boolean data type consists of two values
  - True
  - false
- The Boolean data type is an ordinal data type

# Boolean Expressions (Cont.)

---

- An ordinal data type is any type that is defined as a 1 to 1 mapping to a finite subset of integers
- The ordinal data type mapping for the Boolean data type values are:
  - true = 1
  - false = 0

# Relational Operators

---

- The set of Pascal operators utilized for comparison of the primary data types (i.e. Integer, real and char) are referred to as relational operators
-

# Relational Operators (Cont.)

---

- The Pascal relational operators are:
  - = is equal to
  - < is less than
  - > is greater than
  - <= is less than or equal to
  - >= is greater than or equal to
  - <> is NOT equal to

# Simple Boolean Expressions

---

- The Pascal simple boolean expression is that which consists of any combination of the same data type constants or variables that are compared by a singled relational operator
- A simple boolean expression will always evaluate to either a “true” or “false” boolean value
- Note only operands of the same data type can be compared to one another

# Simple Boolean Expressions (Cont.)

---

- Caveat: reals and integers can be compare however the underlying data type of the integer is interpreted as a real
- Examples
  - $3.8 > 3.5$      true
  - $2.9 \geq 3.0$      false
- The precedence of relational operators in relation to the arithmetic operators is:

# Simple Boolean Expressions (Cont.)

---

- The precedence of relational operators in relation to the arithmetic operators is:

– +, /, MOD, DIV	1 <sup>st</sup>
– +, -	2nd
– =, <, >, <=, >=, <>	3rd

-

# Logical Operators

---

- The logical operators of Pascal are:
  - AND
  - OR
  - NOT
- The precedence (i.e. order of evaluation) of the logical operators is:
  - NOT                    1st
  - AND                    2nd
  - OR                     3rd

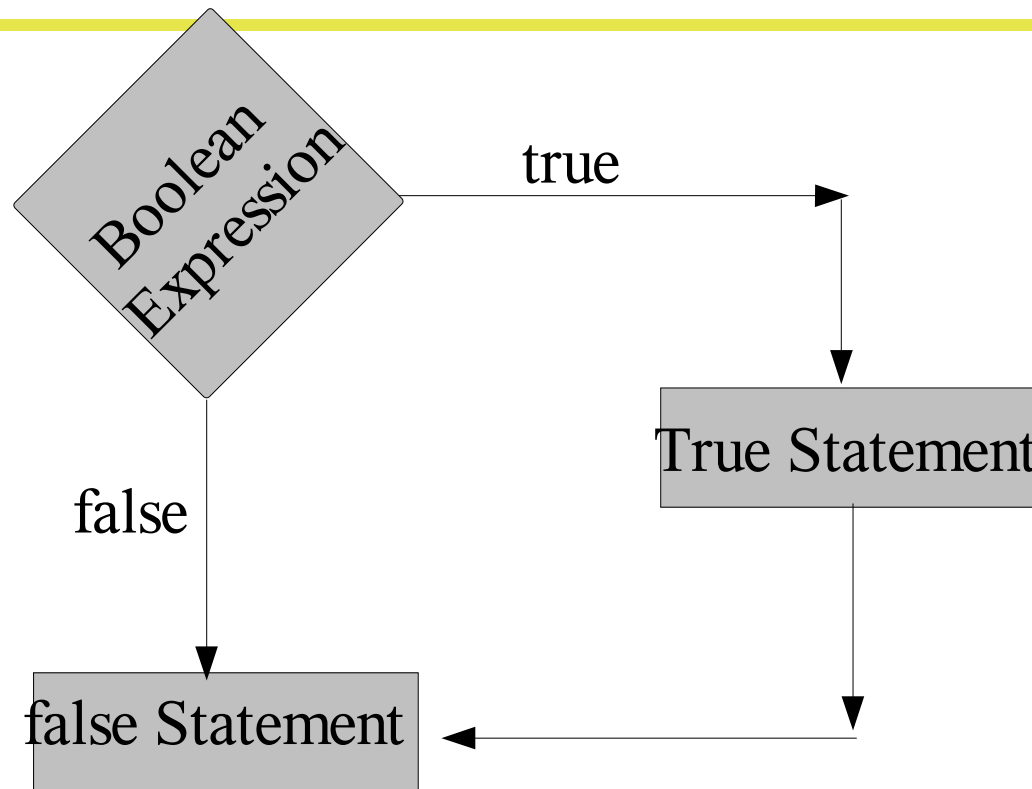
# IF ... THEN Statements

---

- IF <Boolean Expression> THEN
- <Statement>
- One way branch sequential flow construct
- Contains two sequential statement execution branches
  - True branch

# IF ... THEN Statement Flow Diagram

---

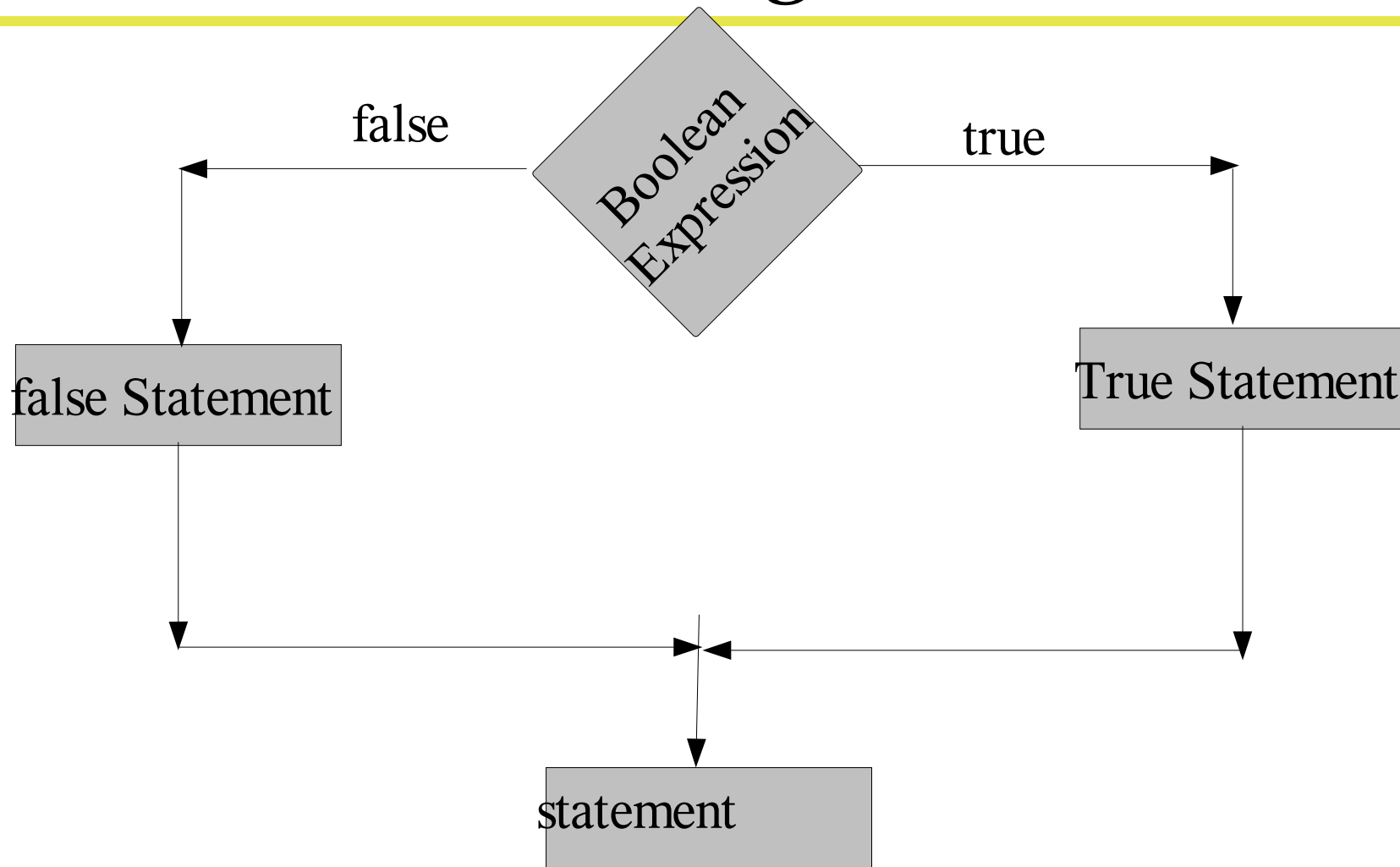


# IF ... THEN ELSE Statements

---

- IF <Boolean Expression> THEN
- <True Statement Branch>
- ELSE
- <False Statement Branch>
- One way branch sequential flow construct
- Contains two sequential statement execution branches
  - True branch
  - False branch

# IF ... THEN... ELSE Statement Flow Diagram



# Selection Statement Examples

---

- See Example 5.10 on page 189 of Nance textbook

# Developing Robust Programs

---

- The primary selection statements along w/ appropriate boolean expressions can be utilized to construct robust programs
- Robust programs are those programs which incorporate assertions/guards that are implemented with a combination of boolean expressions and IF ... THEN statements
-

# Developing Robust Programs (Cont.)

---

- These defined assertions/guards facilitate the protection against crashes from bad data input or generation
- These assertions depending on their placement within the source code implement pre-conditions, post-conditions and invariant
- See initial example definition and usage on page 191 of Nance textbook