§3.1–3.3: IF Statements and Boolean Expressions

devo

23 Sep 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

1) journals in folder
2) Two homeworks by lab section

Chapter 3: Program Structure

- Five basic program structure/flow abstractions:
  - Sequence (;)
  - Selection (IF-THEN-ELSE)
  - Repetition/loops (WHILE, REPEAT)
  - Composition (subroutines)
  - Parallelism
- This chapter mostly covers the first three program structure abstractions
  - Today we'll cover sequences and selection

Statement sequences

- A sequence of statements is executed in order:
  - Successive statements are not executed until the preceding statement is completed

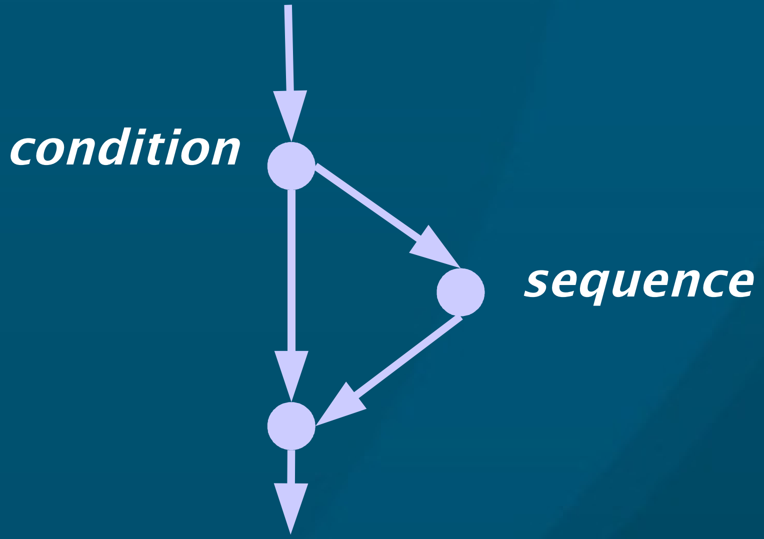
```
WriteString (“Running ReallySlowFunction... ”);
WriteLn;
ReallySlowFunction;
WriteString (“ ...done!”);
WriteLn;
```

- Semicolons separate statements
  - Not needed after last statement:

```
    WriteLn
END HelloWorld.
```

Simple selection: IF–THEN–END

```
IF condition
    THEN
        statement sequence
    END;
```



- Condition is a Boolean expression evaluating to either TRUE or FALSE
- Conditional execution: if condition evaluates to FALSE, then the statement sequence is skipped over and not executed

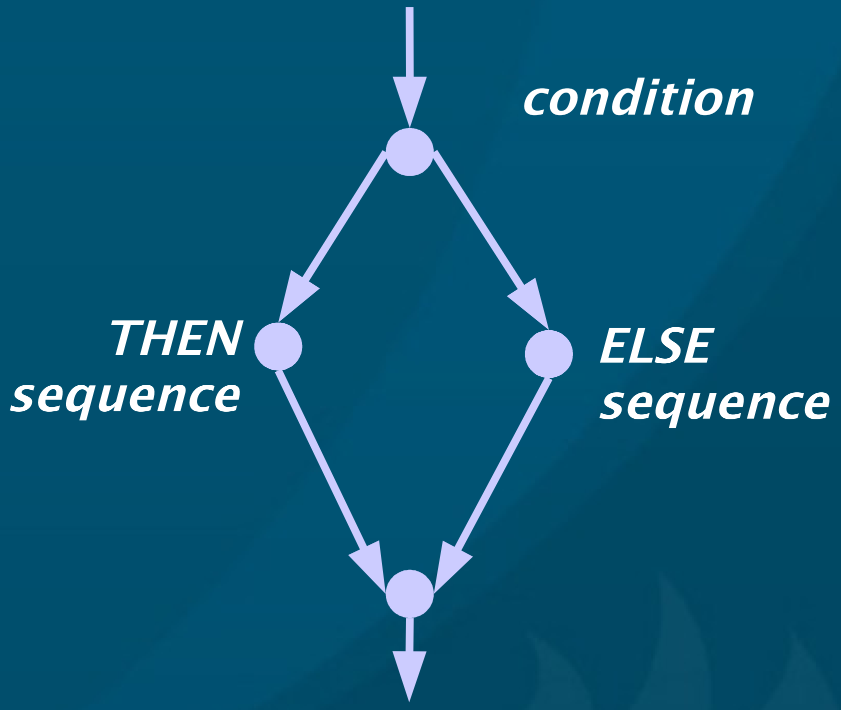
Example using IF-THEN-END

```
IF numApples > 12
    THEN
        WriteString(“Okay, that's waay too many apples!”);
        WriteLn
    END;
WriteString(“Let's eat some apples!”);
WriteLn
```

- Observe indentation, semicolon usage

Branching: IF–THEN–ELSE–END

```
IF *condition*
    THEN
        *statement sequence*
    ELSE
        *statement sequence*
    END
```



- Only one of the two statement sequences is executed

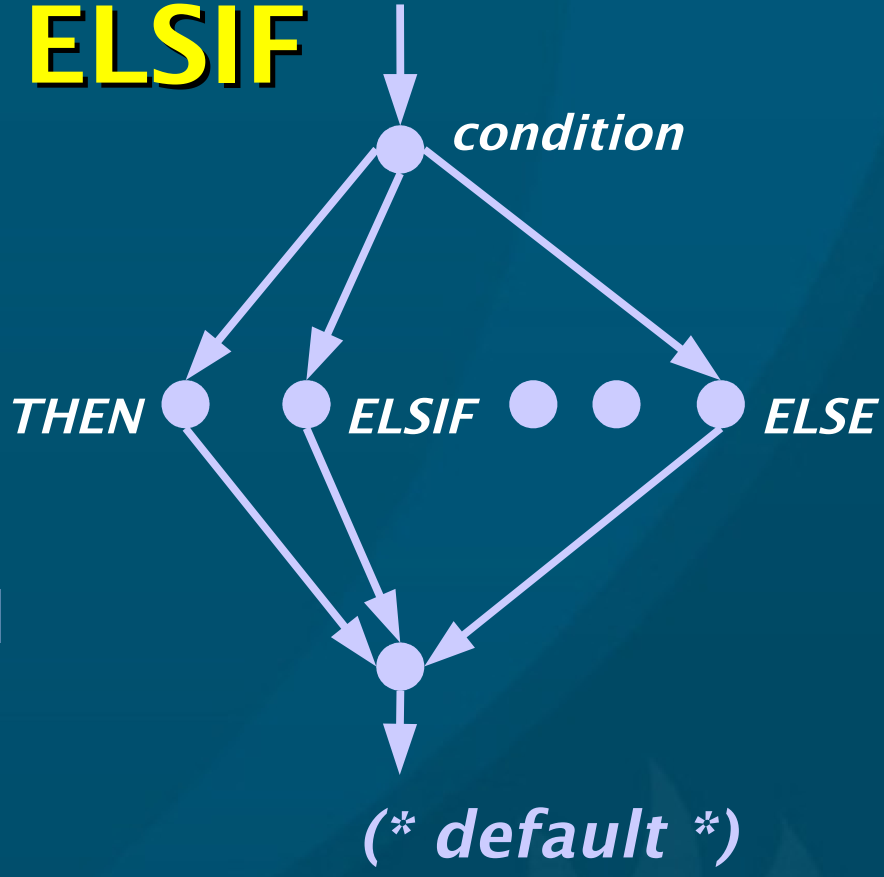
Example using IF-THEN-ELSE-END

```
IF numFriends > 0
    THEN
        applesPerFriend := numApples / numFriends;
    ELSE
        WriteString("Awww, you need some friends!");
    END;
```

- Would the division work if numFriends = 0?
- Will this code generate an error if numFriends=0?

Complex Branching: ELSIF

```
IF userInput = 'y'
    THEN
        Eat (myApple);
    ELSIF userInput = 'n' THEN
        Eat (myPear);
    ELSE
        WriteString ("Please answer 'y' or 'n'.");
    END;
```



- Can have as many ELSIFs as you like
- First ELSIF whose condition is TRUE is executed

A note on indenting preferences

- For purposes of this class, please stick to the book's indenting convention
- But be aware that different people have different personal preferences
- Sean's personal indenting style:

```
IF condition THEN
    statement sequence
ELSIF condition THEN
    statement sequence
ELSE
    statement sequence
END;
```

Boolean expressions

- The conditions in IF statements are Boolean expressions: evaluate to either TRUE or FALSE
- Relational operators:
  - = (equal),
  - < (less than), > (greater than),
  - <= (less than or equal),
  - >= (greater than or equal),
  - <> or # (not equal)
- Boolean operators (connectives):
  - AND (&), OR, NOT (~)

Precedence rules

- In order from highest precedence (first in evaluation) to lowest (last in evaluation):
  - NOT (~)
  - *, /, DIV, MOD, REM, AND (&)
  - +, –, OR
  - =, <, >, <=, >=, <> (#)
- Within the same level, evaluation is done left to right:
  - (2 < 3) & (3 * 2 + 4 = 18) OR NOT (5 # 6)
    - TRUE & (6 + 4 = 18) OR NOT TRUE
    - TRUE & (10 = 18) OR FALSE
    - TRUE & FALSE OR FALSE
    - FALSE OR FALSE
    - FALSE

Shortcut operators

- The boolean operators AND and OR are shortcut operators:
  - The second argument is not even evaluated if not necessary:

```
enoughApplesToGoAround :=
    (numFriends > 0) AND (numApples / numFriends > 2);
```

  - If numFriends is 0, this does not generate a divide-by-zero error

Review of today (3.1–3.3)

- Statement sequences; use of semicolon (;)
- Forms of the IF statement:
  - IF – THEN – END
  - IF – THEN – ELSE – END
  - IF – THEN – ELSIF – THEN – ELSE – END
- Boolean expressions:
  - =, <, >, <=, >=, <> (#)
  - AND (&), OR, NOT (~)
  - Precedence
  - Shortcut semantics

TODO items

- Lab2 due next MTW: §3.14 # (38 / 45)
  - Choose either #38 or #45
  - Short writeup okay
- Quiz ch3 on Mon: may include material not yet covered in lecture – read the text!
- Reading: through §3.8 for Mon