Selection Structures

Every procedure in an application is written using one or more of three basic ***control structures: sequence, selection, and repetition.*** They are called control structures because they control the order in which a procedure's instructions are processed. The procedures in the previous lectures used the sequence structure only. When one of the procedures was invoked during run time, the computer processed its instructions in the order they appeared in the procedure in other words, sequentially. Every procedure you write will contain the sequence structure. Many times, however, a procedure will need to use the selection structure.

Selection Structure

This structure tells the computer that it needs to make a decision before it can select the next instruction to process. The decision is based on a condition specified at the beginning of the selection structure, and the next instruction to process is based on the result of that decision. Figure 2-24 shows examples of two different types of selection structures: single-alternative and dual-alternative. The examples are written in pseudocode, with each selection structure's condition shaded.

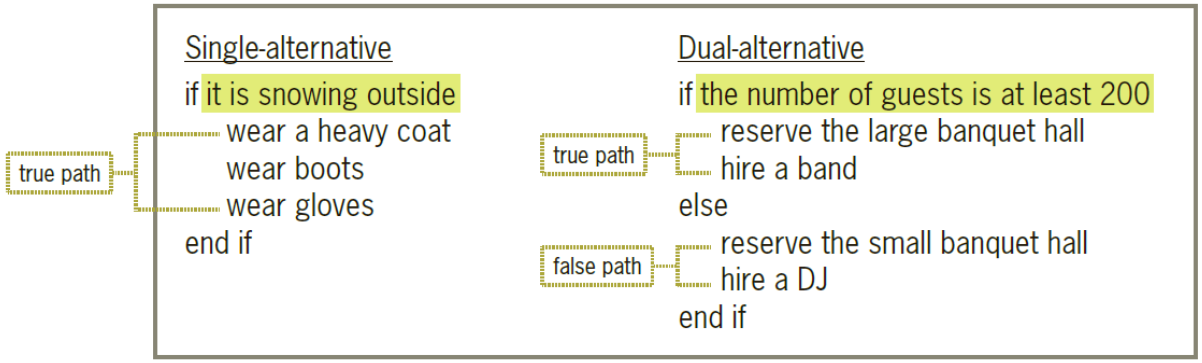

Figure 2-24 Examples of selection structures written in pseudocode

The condition in a selection structure must be phrased so that it evaluates to an answer of either true or false. ***A single-alternative selection structure*** requires one or more actions to be taken only when its condition evaluates to true. ***A dual-alternative selection structure***, on the other hand, requires one set of instructions to be followed only when the condition is true

and a different set of instructions only when it is false. The instructions to follow when the condition evaluates to true are called the true path. As Figure 2-24 indicates, the true path begins with the instruction immediately below the if and ends with either the else (if there is one) or the end if. The instructions to follow when the condition evaluates to false are called the false path. The false path begins with the instruction immediately below the else and ends with the end if. Figure 2-25 shows the flowcharts for the single-alternative and dual alternative selection structures from Figure 2-24. The diamond in a flowchart is called the decision symbol because it is used to represent the ***selection structure's condition (decision).*** The condition in each flowchart is shaded in the figure.

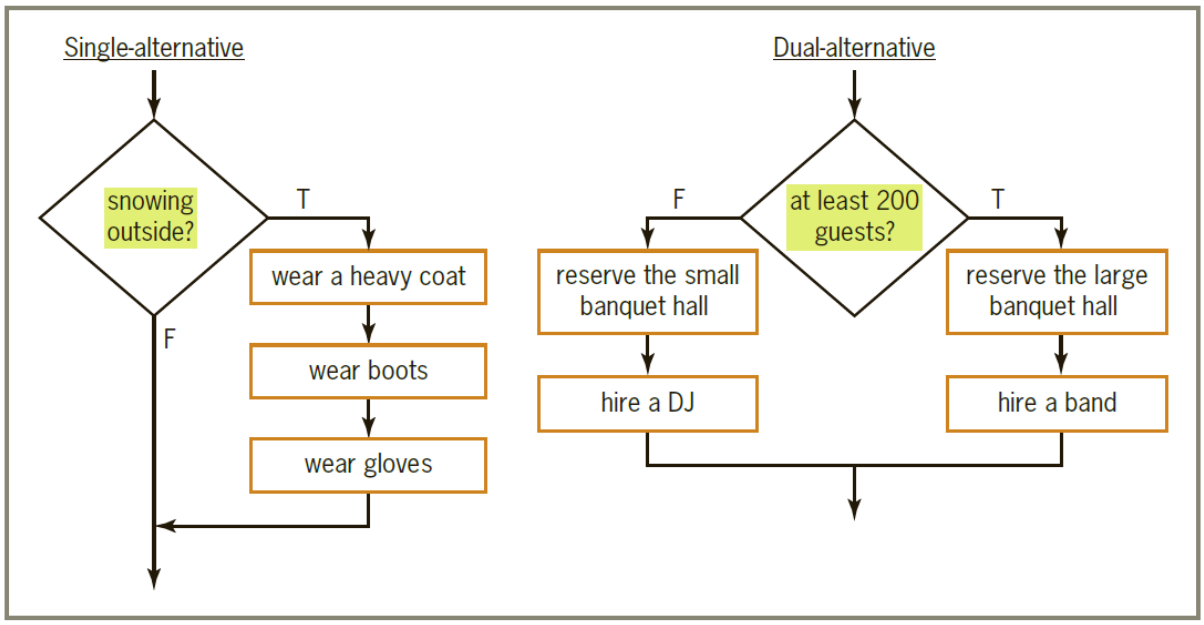

Figure 2-25 Examples of selection structures drawn in flowchart form

Notice that the conditions in both decision symbols evaluate to either true or false only. The flowline marked with a T points to the next instruction to be processed when the condition evaluates to true. Similarly, the flowline marked with an F points to the next instruction to be processed when the condition evaluates to false.

- *If..Then And If..Then..Else Statements*

Visual Basic provides the If…Then and If…Then…Else statement for coding single-alternative and dual alternative selection structures respectively. The statement's syntax is shown in Figure 2-26.

If...Then Statement

Syntax

If *condition* **Then**
    statement block to be processed when the condition is true
End If

If...Then...Else Statement

Syntax

If *condition* **Then**
    statement block to be processed when the condition is true
Else
    statement block to be processed when the condition is false]
End If

Note: The *condition* must be a Boolean expression and can contain variables, literals, properties, methods, keywords, arithmetic operators, comparison operators, and logical operators.

Figure 2-26 Syntax of the If…Then and If…Then…Else statement

The condition in an If statement must be a Boolean expression, which is an expression that results in a Boolean value (True or False). In addition to providing the condition, the programmer must provide the statements to be processed in the true path and (optionally) in the false path. The set of statements contained in each path is referred to as a statement block. An If statement's condition can contain variables, properties, methods, keywords, arithmetic operators, comparison operators, and logical operators. You will learn about comparison operators and logical operators in this lecture. We will begin with comparison operators.

Comparison Operators

Figure 2-27 lists the most commonly used comparison operators in Visual Basic. Comparison operators (also referred to as relational operators) are used to compare two values. When making comparisons, keep in mind that equal to (=) is the opposite of not equal to (< >), greater than (>) is the opposite of less than or equal to (< =), and less than (<) is the opposite of greater than or equal to (> =). Expressions containing a comparison

operator always evaluate to a Boolean value: either True or False. Also included in Figure 2-27 are examples of using comparison operators in an If statement's condition.

Comparison Operators

Operator	Operation
=	equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
<>	not equal to

Examples

```
If State = "IL" Then
If Hours > 40 Then
If Max >= 75.65  Then
If OnHand < Ordered Then
If Total <= 999.99 Then
If Continue <> "N" Then
If IsInsured = True Then
```

Note: In the last example shown above, the condition compares a Boolean variable's value to the Boolean value True. You can omit the = `True` and write the condition as simply `If IsInsured Then`.

Figure 2-27 List and examples of commonly used comparison operators

Unlike arithmetic operators, comparison operators in Visual Basic do not have an order of precedence. When an expression contains more than one comparison operator, the computer evaluates the comparison operators from left to right in the expression. Comparison operators, however, are evaluated after any arithmetic operators, as both examples in Figure 2-28 indicate.

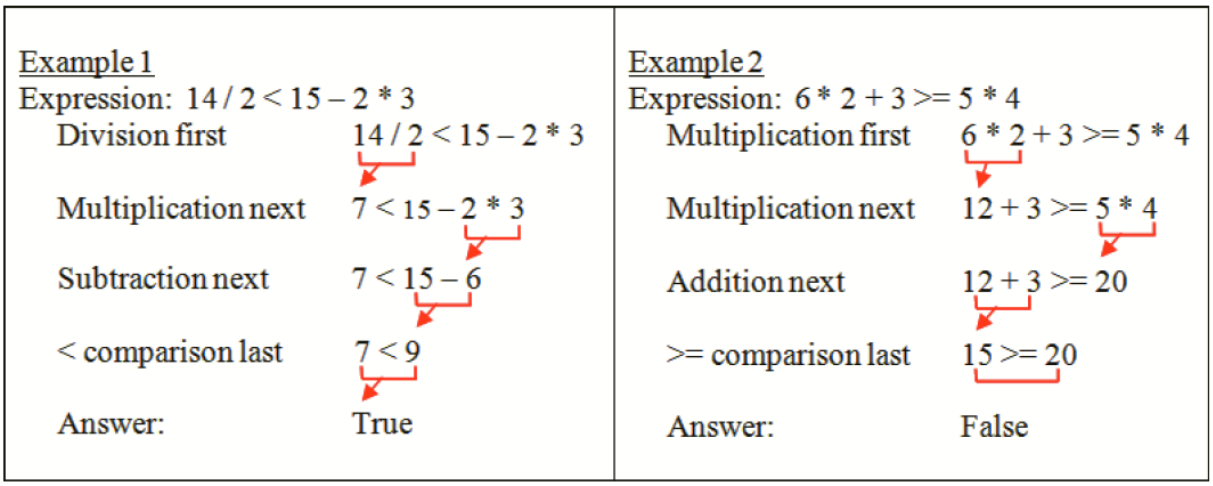

Figure 2-28 Evaluation steps for expressions containing arithmetic and comparison operators