Control Structures

Control Structure: A structure that allows the programmer to determine whether or not specific statements are executed.

Two types of control structure:

1. decision structures
2. loops

Two types of decision structures:

1. IF statement
2. SELECT CASE statement

Blocked IF statement

IF a condition occurs THEN I need to perform certain tasks

```
IF Age = 18 THEN
    PRINT "Be sure to register to vote."
END IF
```

Here the condition is "Age = 18"
The PRINT statement only gets executed if age is 18.

Blocked IF syntax

```
IF   condition  THEN
  Statement block
END IF
```

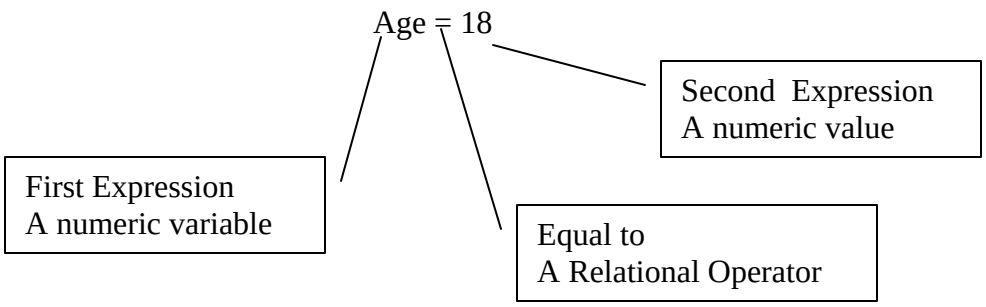
Any number of statements can be included in the body of the blocked IF – the statement block

Note that the statement block is indented

The *condition* must be evaluated as true or false. This is done by the use of Boolean or logical operators within the *condition*.

The execution of the blocked IF is controlled by a Boolean or logical expression.

The blocked IF used a relational operator to compare two expressions, determining whether the first expression is greater than, equal to or less than the second expression.



The expressions can be either numeric or character. However, both the first and second expression must be of the same type. You can not compare *"Mike" > 5*

```
IF "Mike" > 5   THEN
      PRINT "Wrong name"
END IF
```

Relational Operators

Operator	Meaning	Example
<	Less than	2 < 67
<=	Less than or equal to	" A " <= "G"
>	Greater than	23 > 5
>=	Greater than or equal to	"Jonathan" >= "Jon"
=	Equal to	21 + 3 = 24
<>	Not equal to	"jon" <> "Jon"

Computers assign an internal ordering to the set of characters they are able to recognize. This ordering is referred to as then computer's collating sequence. This allows the computer to compare character strings.

Most computers use ASCII (American Standard Code for Information Interchange) see page 129

Capital A has a value of 65
Lower case a has a value of 97

When a computer compares two character strings it compares each character from left to right.

Example	Evaluation
“John” >= “john”	False: J = 74 while j = 106
“John” >= “John”	True
“Jon” < “Jonathan”	True (see below)

When two strings of unequal length are compared, and all the letters of the shorter string match the corresponding letters of the longer string, the shorter string is considered to be less than the longer string.

Leading and trailing blanks are significant, because a blank has a smaller ASCII value (32) than any letter or digit.

Example	Evaluation
“ John” < “John”	True: blank < J

```
CLS
INPUT “Enter the number of CDs: “,  Number
Cost = Number * 10

IF Number >= 6 THEN
      Cost = Cost * .9
END IF
F$ = “\                                                \   $$##.##”
PRINT USING F$; “The cost of the CDs is”; Cost

END
```

Single-alternative decision structure: A decision structure in which an action is taken if the specified condition is true. Otherwise, control continues to the next statement.

Double-alternative decision structure: A decision structure in which one action is taken if the specified condition is true and another action if it is false.

```
IF Speed <= 65 THEN
   PRINT “You are going”; Speed; “miles an hour.”
ELSE
   PRINT “Pull over!”
   Tickets = Tickets + 1
END IF
```

The statement(s) following the THEN is executed if the condition is true; otherwise, (if the condition is false) the condition following the ELSE is executed.

The ELSEIF Clause

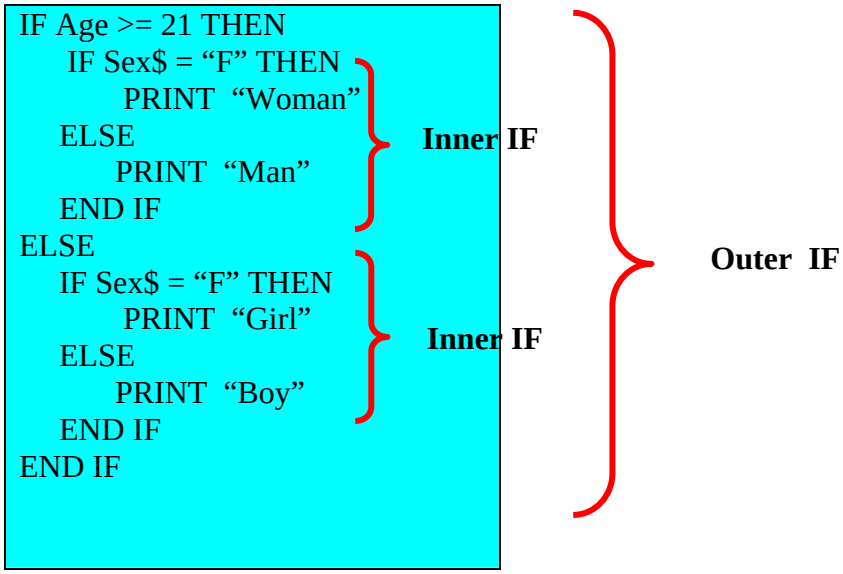
Inserting one or more ELSEIF clauses into a blocked IF statement allows the statement to check for one of several conditions.

```
IF Score > 50000   THEN
   PRINT “Congratulations!  You earned the highest rank of warrior.”
ELSEIF  Score > 35000 THEN
    PRINT “You earned the rank of commander.”
ELSEIF Score > 20000  THEN
    PRINT “You earned the rank of captain.”
ELSE
     PRINT “You earned the rank of cadet.”
END `IF
```

Score	**Results**
33000	*IF Score > 50000 THEN* Evaluates as false therefore *PRINT “Congratulations! You earned the highest rank of warrior.”* Is ignored and control passes to the next line *ELSEIF Score > 35000 THEN* Evaluates as false therefore *PRINT “You earned the rank of commander.”* Is ignored and control passes to the next line *ELSEIF Score > 20000 THEN* Evaluates as true therefore *PRINT “You earned the rank of captain.”* Is executed and control passes to the line after the ENDIF

Nested block IF statements

It is possible to nest block IF statements by placing one inside the other.



Notice each IF must have its own END IF

Indenting block statements makes their logic easier to follow.

The SELECT CASE Statement

The SELECT CASE statement allows an action to be selected from a list of alternative.

```
INPUT “Enter your grade level (9-12)”, Class
SELECT CASE Class
    CASE  9
       PRINT “Freshman”
    CASE  10
       PRINT “Sophomore”
    CASE  11
       PRINT “Junior”
    CASE  12
       PRINT “Senior”
     CASE ELSE
       PRINT “Invalid grade level”
END SELECT
```

If the user inputs 10 of class the program finds the case where class is equal to 10 and then executes the statements in that block. Here it prints to the screen the word *Sophomore*.

If the value of Class is invalid (not a 9, 10, 11, or 12) the statement(s) in the CASE ELSE clause is executed.

The CASE ELSE is optional, but is useful for checking for invalid input.

SELECT CASE and Relational Operators

```
SELECT CASE Score
    CASE IS > 50000
        PRINT "Congratulations!  You earned the highest rank of warrior."
    CASE 50000 TO 35001
        PRINT "You earned the rank of commander."
    CASE 35000 TO 20001
        PRINT "You earned the rank of captain."
    CASE IS <= 20000
       PRINT "You earned the rank of cadet."
END SELECT
```

If you do not insert IS when using a relational operator, QBasic will insert it automatically.

When checking for ranges, the key word TO is used.

SELECT CASE is perfect for creating Menus.

A **MENU** is a list of the functions that a program can perform.

Logical Operators

There are four logical or Boolean operators: AND, OR, NOT, XOR

From time to time you will need to test more than one set of variables. You can combine more than one relational test into a compound relational test by using the following logical operators:

AND OR XOR NOT

The AND truth table (both sides must be true)

True	**AND**	**True**	**=**	**True**
True	**AND**	**False**	**=**	**False**

False	**AND**	**True**	**=**	**False**
Flase	**AND**	**False**	**=**	**False**

The OR truth table (one side must be true)

True	**OR**	**True**	**=**	**True**
True	**OR**	**False**	**=**	**True**
False	**OR**	**True**	**=**	**True**
False	**OR**	**False**	**=**	**False**

The XOR truth table (one or the other must be true, but not both)

True	**XOR**	**True**	**=**	**False**
True	**XOR**	**False**	**=**	**True**
False	**XOR**	**True**	**=**	**True**
False	**XOR**	**False**	**=**	**False**

The NOT truth table (causes an opposite relation.)

NOT True	**=**	**False**
NOT False	**=**	**True**

Examples:

A *must be less than* ***B****, and* ***C*** *must be greater than* ***D****, for the* ***CalcIt*** *routine to execute.*

```
IF ( (A < B)  AND (C> D) )  THEN PRINT CalcIt
```

The ***sales*** *must be more than* ***5000*** *or the* ***hrsWorked*** *must be more than* ***81*** *before the* ***OverPay*** *routine executes.*

```
IF ( (sales > 5000)  OR (hrsWorked > 81) )  THEN PRINT OverPay
```

The variable called ***bit2*** *must be equal to* ***0*** *or* ***bit3*** *must not be equal to* ***1*** *before* ***Error*** *is printed. If both are true, however, the test fails (because* ***XOR*** *is used), the* ***PRINT*** *is ignored, and the next instruction in sequence executes.*

```
IF ( (bit2 = 0)  XOR (bit3 = 1) )  THEN PRINT "Error"
```

If the ***sales*** *are not less than* ***2500****, the* ***bonus*** *is initialized*

```
IF ( NOT(sales < 2500)  )  THEN bonus = 500
```

The Complete Order of Operators

Order	Operator
1	Parentheses
2	Exponentiation ^
3	Negation (the unary -)
4	Multiplication, division, integer division, MOD (*, /, \, MOD)
5	Addition, Subtraction (+, -)
6	Relational operators (=, <, >, <=, >=, <>)
7	NOT
8	AND
9	OR
10	XOR

See Menu Page 142 in Textbook