# COMPUTER PROGRAMMING

## CONTROLLING EXECUTION WITH CONTROL STRUCTURES

**DR. USMAN AKMAL**

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE

## CONTENTS

## CONTROL STRUCTURES

A control structure allows the programmer to determine whether or not specific statements are executed.

QBasic has two control types:

- Decision structures
- Loops

## DECISION STRUCTURES

Decision structures are used to make comparisons in order to decide if certain statements and actions are to be executed or taken.

One form is the IF Statement Block which is a single-alternative decision. It either does something or it does nothing at all.

## IF STATEMENT BOLOCK

The form of the IF Statement Bolock is as follows:

```
IF (expression) THEN
     ------------
     ------------
     ------------
     ------------
END IF
```

The central statements will only be executed if (expression) is true; otherwise, execution moves on to the next executable statement.

## IF ELSE STATEMENT

Another form of a decision structure is the IF ELSE STATEMENT, which is referred to as a double-alternative decision structure.
The form of the IF ELSE statement is as follows:

```
IF (expression 1) THEN
    ------------
ELSE
    ------------
END IF
```

First action is taken if the expression 1 is true and another action is taken if the expression is false.

## IF ELSE STATEMENT [EXAMPLE]

```
CLS
DIM number AS INTEGER
INPUT “Enter any Number = “, number
    IF number >= 100 THEN
        PRINT "This is a high value."
    ELSE
        PRINT "This is a low value."
    END IF
```

## IF ELSEIF ELSE STATEMENT

If you want to check for one of several conditions, you can add ELSEIF clauses into the body of the IF ELSE STATEMENT. This type of decision structure will always perform some action. The form is as follows:

```
IF expression1 THEN
     stmtT1
ELSEIF expression2 THEN
     stmtT2
ELSEIF expression3 THEN
     stmtT3
ELSE
     stmtF
END IF
```

## IF ELSEIF ELSE STATEMENT [EXAMPLE]

```
CLS
DIM score AS INTEGER
INPUT “Enter Student Marks = “, score
     IF score >= 90 THEN
          PRINT "Grade = A"
     ELSEIF score >= 80 THEN
          PRINT "Grade = B"
     ELSEIF score >= 70 THEN
          PRINT "Grade = C"
     ELSEIF score >= 60 THEN
          PRINT "Grade = D"
     ELSE
          PRINT "Grade = F"
     END IF
```

## NESTED IF STATEMENTS

You can also check for several conditions by using nested IF statements, which are IF statements used in the body of IF statements. The form for nested IFs is as follows:

```
IF expression1 THEN
      IF expression1A THEN
            stmtT1A
      ELSE
            stmtF1A
      END IF
ELSE
      IF expression1B THEN
            stmtT1B
      ELSE
            stmtF1B
      END IF
END IF
```

## SELECT CASE STATEMENT

- A SELECT CASE statement is another control structure which allows an action to be selected from a list of alternatives.
- The SELECT CASE statement uses various "cases", individually named CASE, which include one or more statements to be executed if the specified value of the expression equals the value of the "case".
- There is also a CASE ELSE clause which is optional but is useful for validating user input.

## SELECT CASE STATEMENT

- The SELECT CASE statement is particularly efficient when menus are included in a program.
- A menu is a list of options that is displayed to the user with each option having an action to take place if it is selected.
- When the user makes a choice, the choice can be evaluated easily with a SELECT CASE statement.

## SELECT CASE STATEMENT

The form of a SELECT CASE statement is as follows:

```
SELECT CASE testExpression
    CASE expression1
        stmt1
    CASE expression2
        stmt2
    CASE expression3
        stmt3
    CASE ELSE
        stmt(N)
END SELECT
```

# SELECT CASE STATEMENT [EXAMPLE]

```
CLS
PRINT “1. Area of Rectangle” ‘Option 1
PRINT “2. Area of Circle” ‘Option 2
PRINT “3. Area of Triangle” ‘Option 3

INPUT "Enter Option number(1 - 3): ", OptNum

SELECT CASE OptNum
      CASE 1
            PRINT “Area of Rectangle"
      CASE 2
            PRINT “Area of Circle"
      CASE 3
            PRINT “Area of Triangle"
      CASE ELSE
            PRINT "You did not specify an index number
from 1 - 3!"
END SELECT
```

## SELECT CASE STATEMENT [EXAMPLE]

When checking for a range of values while using a SELECT CASE statement, you must use the keyword IS when using a relational operator to make a comparison, and you must use the keyword TO for checking the ranges.

```
CLS
INPUT "Enter the Test Marks: ", TestMarks%

SELECT CASE TestMarks%
      CASE IS >= 90
            PRINT "Your grade is an A!"
      CASE 80 TO 89
            PRINT "Your grade is a B!"
      CASE 70 TO 79
            PRINT "Your grade is a C!"
      CASE 60 TO 69
            PRINT "Your grade is a D!"
      CASE IS <= 59
            PRINT "Your grade is a F!"
```

## CONTROL STRUCTURES [LOOPS]

- **WHILE...WEND LOOP**
- **DO LOOP**
- **FOR...NEXT LOOP**

## LOOPS

Loops allow a specified group of statements to be executed a certain number of times. Because the exact same code is being executed a certain number of times, we call this "looping" or "iteration" in programming.

QBasic offers two type of looping statements:

- WHILE...WEND
- DO...LOOP
- FOR...NEXT

## WHILE...WEND LOOP

The WHILE...WEND command continues a loop until a specified expression is false.

```
WHILE (expression/condition)
    ------------
    ------------
    ------------
    ------------
WEND
```

## WHILE...WEND LOOP [EXAMPLE]

```
CLS
x = 10
WHILE x < 15
   PRINT x
   x = x + 1
WEND
```

## DO...LOOP

DO...LOOP is same as WHILE...WEND, except it has following two advantages.

i. Loop until an expression is true
ii. Loop at least one time regardless of whether the expression is true or not.

DO…LOOP continues "while" the expression is true or "until" the expression is true, using the WHILE and UNTIL statements, respectively.

## DO…LOOP STRUCTURE USING WHILE KEYWORD

```
DO WHILE (expression/condition)
     -----------
     -----------
LOOP


DO
     -----------
     -----------
LOOP WHILE (expression/condition)
```

## DO...LOOP STRUCTURE USING WHILE KEYWORD

[EXAMPE 1]

```
x = 10
DO WHILE x < 15
    PRINT x
    x = x + 1
LOOP
```

[EXAMPE 2]

```
x = 10
DO
    PRINT x
    x = x + 1
LOOP WHILE x < 15
```

## DO...LOOP STRUCTURE USING UNTIL KEYWORD

```
DO UNTIL (expression/condition)
     ------------
     ------------
LOOP


DO
     ------------
     ------------
LOOP UNTIL (expression/condition)
```

## DO...LOOP STRUCTURE USING UNTIL KEYWORD

[EXAMPE 1]

```
x = 10
DO UNTIL x = 15
    PRINT x
    x = x + 1
LOOP
```

[EXAMPE 2]

```
x = 10
DO
    PRINT x
    x = x + 1
LOOP UNTIL x = 15
```

## FOR...NEXT LOOP

A FOR...NEXT loop is generally used as a counter loop when you know exactly how many times you need to execute the loop. The form of a FOR...NEXT loop is as follows:

```
FOR <Variable> = <startVal> TO <EndVal> [STEP <increment>]
-----------
-----------
NEXT <variable name>
```

The Variable is any variable used as a counter. It will be first initialized to the specified StartVal. The EndVal marks the condition when the loop should end "looping". The optional STEP value specifies how large to increase(+ve)/decrease(-ve) the loop control variable. If not provided, the default STEP value is 1.

## FOR...NEXT LOOP [EXAMPLES]

[EXAMPE 1]

```
CLS
x = 10
y = 1
FOR I = 1 TO 5
     PRINT x
     x = x + y
NEXT I
```

[EXAMPE 2]

```
CLS
FOR x = 1 TO 10 STEP 2
     PRINT x
NEXT x
```

[EXAMPE 3]

```
CLS
FOR x = 100 TO 0 STEP -5
     PRINT x,
NEXT x
```

## STOPPING LOOPS

To stop a loop prematurely, use the EXIT command, followed by either FOR or DO.

```
FOR x = 1 TO 5
    PRINT x
    IF x = 3 THEN EXIT FOR
NEXT x
```

NOTE: This command only works with the DO...LOOP and FOR...NEXT LOOP, not with WHILE...WEND.

END OF LECTURE

28