Unit II: Decision control and loop Control

2.1 Data Types in VB.Net:

Data types in VB.NET define the type of data that a variable can hold. They specify the size and layout of the variable's memory, as well as the range of values that can be stored in the variable.

Syntax for declaration:

Dim variableName As DataType

Example:

```vb
Dim age As Integer
Dim name As String
Dim isStudent As Boolean
Dim weight As Double
Dim birthDate As Date
```

2.2 Operators in VB.Net

i. Arithmetic Operators:

Arithmetic operators in VB.NET are used to perform mathematical operations such as addition, subtraction, multiplication, division, and exponentiation.

Syntax:

```
result = operand1 + operand2   ' Addition
result = operand1 - operand2   ' Subtraction
result = operand1 * operand2   ' Multiplication
result = operand1 / operand2   ' Division
result = operand ^ exponent    ' Exponentiation
```

Example:

```vb
Dim x As Integer = 10
Dim y As Integer = 5
Dim sum As Integer = x + y         ' sum = 15
Dim difference As Integer = x - y  ' difference = 5
Dim product As Integer = x * y     ' product = 50
Dim quotient As Integer = x / y    ' quotient = 2
Dim power As Integer = x ^ 2       ' power = 100
```

ii. Logical Operators:

Logical operators in VB.NET are used to perform logical operations such as AND, OR, and NOT on Boolean values.

Syntax:

```
result = operand1 AndAlso operand2  ' Logical AND
result = operand1 OrElse operand2   ' Logical OR
result = Not operand                ' Logical NOT
```

Example:

```vb
Dim x As Boolean = True
Dim y As Boolean = False
Dim resultAnd As Boolean = x AndAlso y  ' resultAnd = False
Dim resultOr As Boolean = x OrElse y    ' resultOr = True
Dim resultNot As Boolean = Not x        ' resultNot = False
```

iii)Bit Shift Operators:

Bit shift operators in VB.NET are used to shift the bits of a binary number left or right.

Syntax:

```vb
result = operand << shiftCount  ' Left Shift
result = operand >> shiftCount  ' Right Shift
```

Example:

```vb
Dim x As Integer = 8
Dim leftShift As Integer = x << 1  ' leftShift = 16 (Binary: 1000 -> 10000)
Dim rightShift As Integer = x >> 1 ' rightShift = 4 (Binary: 1000 -> 100)
```

iv. Relational Operators:

Relational operators in VB.NET are used to compare values and determine the relationship between them (e.g., equal to, greater than, less than).

Syntax:

```
result = operand1 = operand2   ' Equal to
result = operand1 <> operand2  ' Not equal to
result = operand1 > operand2   ' Greater than
result = operand1 < operand2   ' Less than
result = operand1 >= operand2  ' Greater than or equal to
result = operand1 <= operand2  ' Less than or equal to
```

Example:

```
Dim x As Integer = 10
Dim y As Integer = 5
Dim isEqual As Boolean = x = y       ' isEqual = False
Dim isNotEqual As Boolean = x <> y   ' isNotEqual = True
Dim isGreater As Boolean = x > y     ' isGreater = True
Dim isLess As Boolean = x < y        ' isLess = False
Dim isGreaterOrEqual As Boolean = x >= y  ' isGreaterOrEqual = True
Dim isLessOrEqual As Boolean = x <= y     ' isLessOrEqual = False
```

v. Assignment Operators:

Assignment operators in VB.NET are used to assign values to variables and perform operations in a single step.

Syntax:

```
variable = expression          ' Simple Assignment
variable += expression         ' Addition Assignment
variable -= expression         ' Subtraction Assignment
variable *= expression         ' Multiplication Assignment
variable /= expression         ' Division Assignment
variable \= expression         ' Integer Division Assignment
variable ^= expression         ' Exponentiation Assignment
```

Example:

```vb
Dim x As Integer = 10
x += 5   ' Equivalent to x = x + 5, x becomes 15
x -= 3   ' Equivalent to x = x - 3, x becomes 12
x *= 2   ' Equivalent to x = x * 2, x becomes 24
x /= 4   ' Equivalent to x = x / 4, x becomes 6
x \= 2   ' Equivalent to x = x \ 2, x becomes 3
```

```
x ^= 2   ' Equivalent to x = x ^ 2, x becomes 9
```

2.3 Control Structures:

i. **IF Statement::**

The IF statement in VB.NET allows you to execute a block of code if a condition is true.

Syntax:

```
If condition Then
    ' Code to execute if condition is true
End If
```

Example:

```
Dim x As Integer = 10
If x > 5 Then
    Console.WriteLine("x is greater than 5")
End If
```

ii. **IF-ELSE Statement:**

The IF-ELSE statement in VB.NET allows you to execute one block of code if a condition is true and another block of code if the condition is false.

Syntax:

```
If condition Then
    ' Code to execute if condition is true
Else
    ' Code to execute if condition is false
End If
```

Example:

```
Dim x As Integer = 3
If x > 5 Then
    Console.WriteLine("x is greater than 5")
Else
    Console.WriteLine("x is not greater than 5")
End If
```

iii)**Select Case Statement:**

The Select Case statement in VB.NET allows you to execute one of several blocks of code based on the value of an expression.

Syntax:

```
Select Case expression
    Case value1
        ' Code to execute if expression equals value1
    Case value2
        ' Code to execute if expression equals value2
    Case Else
        ' Code to execute if expression does not match any specified values
End Select
```

Example:

```vb
Dim dayOfWeek As Integer = 3
Select Case dayOfWeek
    Case 1
        Console.WriteLine("Sunday")
    Case 2
        Console.WriteLine("Monday")
    Case 3
        Console.WriteLine("Tuesday")
    Case Else
```

```
    Console.WriteLine("Other")
End Select
```

2.4 Loops in VB.Net:

i.) For Loop:

The For loop in VB.NET is used to execute a block of code a specified number of times. It's particularly useful when you know how many times you want to execute a block of code.

Syntax:

```
For index As Integer = startValue To endValue Step stepValue
    ' Code to be executed
Next
```

Example:

```
For i As Integer = 1 To 5
    Console.WriteLine("Iteration: " & i)
Next
```

ii.) While Loop:

The While loop in VB.NET is used to execute a block of code repeatedly as long as a specified condition evaluates to true.

Syntax:

```
While condition
    ' Code to be executed
End While
```

Example:

```vb
Dim i As Integer = 1
While i <= 5
    Console.WriteLine("Iteration: " & i)
    i += 1
End While
```

iii. Do Loop:

The Do loop in VB.NET is used to execute a block of code repeatedly either until a condition becomes false (Do While) or until a condition becomes true (Do Until).

Syntax:

```
Do
    ' Code to be executed
Loop While condition

Do
    ' Code to be executed
Loop Until condition
```

Example (Do While):

```vb
Dim i As Integer = 1
Do
    Console.WriteLine("Iteration: " & i)
    i += 1
Loop While i <= 5
```

Example (Do Until):

```vb
Dim i As Integer = 1
```

```
Do
    Console.WriteLine("Iteration: " & i)
    i += 1
Loop Until i > 5
```

iv. For Each Loop

The For Each loop in VB.NET is used to iterate over elements in a collection or array without the need for an explicit loop counter.

Syntax:

```
For Each element As DataType In collection
    ' Code to be executed
Next
```

Example:

```
Dim numbers() As Integer = {1, 2, 3, 4, 5}
For Each num As Integer In numbers
    Console.WriteLine("Number: " & num)
Next
```

2.5 Form Controls in VB.Net and Its Properties:

Button:

- Properties: Text, Name, Enabled, Visible, Click Event.

Text Box:

- Properties: Text, Name, Enabled, Visible, Multiline, PasswordChar.

Label:

- Properties: Text, Name, Enabled, Visible, Font, ForeColor, BackColor.

Radio Button:

- Properties: Text, Name, Enabled, Visible, Checked.

Check Box:

- Properties: Text, Name, Enabled, Visible, Checked.

List Box:

- Properties: Items, SelectedIndex, Name, Enabled, Visible.

Combo Box:

- Properties: Items, SelectedIndex, Name, Enabled, Visible.

Picture Box:

- Properties: Image, Name, Enabled, Visible, SizeMode.

Panel:

- Properties: Name, Enabled, Visible, BackColor.

Tab Control:

- Properties: TabPages, SelectedTab, Name, Enabled, Visible.

Timer:

- Properties: Interval, Enabled, Name.

These are some of the common form controls in VB.NET along with their properties that you can manipulate to build interactive user interfaces.