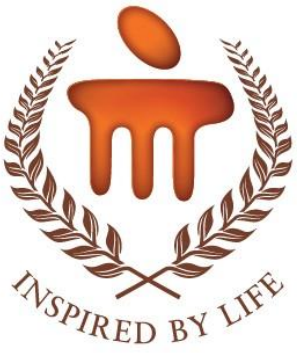

BACHELOR OF COMPUTER APPLICATIONS

SEMESTER *5*

DCA3102

VISUAL PROGRAMMING

Unit 3

Mastering VB Language

Table of Contents

1. INTRODUCTION

In the previous unit we discussed the concept of Visual Studio .NET and Microsoft Development Environment. Integrated design environment and integrated debugging environment that helps the program developers to develop the application in a convenient way. Some basic concepts and basic skills required to work with the Development Environment. We also discussed to develop a simple application in VB .NET environment and to run the application.

In this unit we are going to discuss the data types, operators, decision making, looping statements, error handling, techniques supported by VB

.NET platform. Also we are introducing the object oriented programming concepts discussing on classes and objects

1.1 Objectives:

After studying this unit, you will be able to:

- *List and explain the various data types which is supported by VB .NET*
- *List and explain the operators*
- *Analyse the program using decision making and looping structure*
- *Discuss the error handling techniques in VB .NET*

2. DATA TYPES

Data types are a broad concept that describes the various ways in which variables and functions can be declared. The size of a variable's storage and how its bits are interpreted are both affected by its type. A variable has a major role in programming languages. Any holder or container with a name to hold some value is called variable. These variables are the basic fundamental need for any data processing. Basically the string or the text variables are simple. Whereas the numerical type comes with the different size and precision in order to improve the performance of an application. An earlier version of VB .NET supported the default variable as variant. The unique values Empty, Error, Nothing, and Null can also be stored in a Variant. With the VarType or TypeName function, you can specify the behaviour of the data included in a Variant. If the programmer is not mentioning about the type of variable while declaration, it will be considered as variant. This variant type holds any types of data like number or string. It reduces the burden of the programmer to some extent and helps to decide later.NET does not support the variant type because it requires confirmation between the other languages.

The variant type, is found to efficient even though it had two, fatal flaws from the perspective of those who designed VB.NET. First, in some cases, VB had a hard time figuring out which type the variant should change to resulting in an error. Second, the other languages in the .NET universe do not use variants and the .NET philosophy requires conformity between its various languages (at least on the fundamental issues, such as variable typing). Therefore, the variant variable is no longer part of the VB language. It has been banished in VB.NET.

X = 10 and y = 10.3

Here when the value 10 assigned to the variable x, VB understands that this is the integer type. But the y value has the fractional part insist to change the type to floating type so here casting is happening.

```
x = "11"
y = 10
z = x + y
MsgBox (z)
```

From this example you will get the answer as 21, but before doing this calculation the casting process is require converting from string to integer. This interpretation process delays the processing or execution time of an application. Thus the VB .NET is not entertaining the usage of variant type. The easiest and simplest data type is Boolean. It can represent two states that is, True and False. The default value of the Boolean is False. This type can be used where you want to toggle between states of True and False or say On and Off. The syntax for the Boolean data type is:

Dim bool1 As Boolean

Another simple data type is the Integer and its larger size, the Long type. Before VB.NET, the Integer data type was 16 bits large and the Long data type was 32 bits large. Now these types are twice as big as they used to be: Integer is 32 bits large and Long is 64 bits large. If your program needs to use a 16-bit integer, use the new type Short. So if you're translating pre-

.NET VB code, you need to change as Integer or Cint commands to As Short and Cshort, respectively. Similarly, As Long and CLng now must be changed to As Integer and Cint. In most programming, the Integer is the most common numeric data type. If your non-fractional number is larger or smaller than an integer can hold, make it a Long data type.

Dim abc As Integer
Dim xyz As Long

The other major numeric type is called floating point. It has similar small and large versions called Single and Double, respectively. Use it when your program requires the precision of using fractions:

Dim fract1 As Single, fract2 As Double

VB.NET also has a new Char type, which is an unsigned 16-bit type that is used to store Unicode characters. The new Decimal type is a 96-bit signed integer scaled by a variable power of 10. Table 3.1 listing the data types available ib VB .NET.

Table 3.1: Data types in VB .NET

Visual Basic Type	Common Language Runtime Type Structure	torageSize	Value Range
Boolean	System.Boolean	2 bytes	True or False
Byte	System. Byte	1 byte	Unsigned 0 to 255
Char	System. Char	2 bytes	Unsigned 0 to 65535
Date	System. Date Time	8 bytes	January 1, 0001 to December 31, 9999
Decimal	System. Decimal	16 bytes	+/- 79,228,162,514,264,337,593,543,950,335 with no decimal point; +/- 7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest non-zero number is +/-0.0000000000000000000000000001
Double (double-precision floating-point)	System. Double	8 bytes	-1.79769313486231E+308 to 4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486231E+308 for positive values
Integer	System.Int32	4 bytes	-2,147,483,648 to 2,147,483,647
Long (long integer)	System.Int64	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Object	System. Object (class)	4 bytes	Any type can be stored in a variable of type Object
Short	System.Int16	2 bytes	-32,768 to 32,767
Single (single-precision floating-point)	System. Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E+38 for positive values
String (variable-length)	System. String (class)	Depends on platform	0 to approximately 2 billion Unicode characters
User-Defined Type (structure)	(inherits from System. Value Type)	Sum of the sizes of its members	Each member of the structure has a range determined by its data type and independent of the ranges of the other members

Data types Features:

We will the features of datatypes , and how data types help in programming . The data type describes which operations can securely be performed to create, transform and use the variable in different computation.

A strongly typed language prevents errors because it is logical to ask the computer to multiply a float by an integer. When a programming language wants a variable to be utilized in only ways that value its data type, the language is called to be strongly typed. When a programming language allows a variable of one data type to be used as if it were a value of another data type, the language is said to be weakly typed. Defining data types helps in clearly knowing the scope of a particular variable and the way of using the variable.

Strongly typed and weakely types data types:

Depending on the defining the data type we categories the programming language as strongly typed programming language and weakly typed programming language The extent to which a programming language discourages or prevents type error is known as type safety. A programming language is said to be strongly typed if its users are required to explicitly declare the types of data they are working with. When a programming language does not necessitate the explicit specification of multiple types of objects and variables, it is said to be weakly typed. To further elaborate on strongly typed language the expressions have types and types must be "match" for example :Java, C# are the strongly languages. Were as in weakly typed language there is no need of defining the data type explicitly , the variables or the expressions uses the daattypes just for memory allocations.

Importance of Datatypes:

Now lets see how important the data types are for a variables or an expressions, the data type helps the processor to know the expected amount of memory space to reserve depending on the type of the data type used it will try to reserve so much of memory space for the defined variable, It also indicates the type or category of data or the information and also helps to categories the related information of the real world entities into few categories. So in simple it helps in generalizing the category to which data type it falls and helps the

compiler easily compile the program. A variable's data type specifies the meaning of the bits stored in that variable's section of memory. The biggest benefit is the compiler's ability to warn you about inconsistencies and potential runtime issues.

programming language wants a variable to be utilized in only ways that value its data type, the language is called to be strongly typed. When a programming language allows a **rPrimitive and Non-Premitive Datatype:**a

So whenever we start with the data types firstly we will discus the basic or also called as primitive type of data types The above data types have values containing numbers, alphabets, etc. to make the processing of these huge data for programmes/machines easy. The language already has a set of predefined primitive kinds and gives them special names using reserved keywords. They stand for the linguistic building blocks. Some of the basic primitive data types are integer, long int, double, Boolean, byte and character. Integer is referred to as 32 bit signed integers, long int is said to be 64 bit signed integers, double is 32 bit single precision floating point , Boolean carries true or false value.

Example for primitive datatype:

Explanation on Data Types: Boolean, Byte and Char

We will see the detailed explanation and values of the datatypes like Boolean, byte and char , Boolean is a data type that contains only two values, i.e., 0 and 1 which usually represent two types of behaviour, either true or false.

The byte data type is utilized to save memory in large arrays where the memory savings is most needed EX:- 1,2 ,3, can be byte data type The greatest length of a CHAR value is 2000 bytes, it represents a single 16 bit character, The CHAR data type saves character values. It saves these data types as a fixed-length string. Ex:- A, B, C

There are some more data type which is discussed next

Explanation on Data Types: Integer, Long and Double

Talking about the integer datatype Integers are normally represented in a computer as a group of binary digits data (bits). It also has long integer and short integer , the long int is a

numeric data type which belong to primitive data type it takes 64 signed integer bit memory, generally used for larger integer storage, then we have double data type which is the default decimal point type programing language. It is expensive, carries more space and is more effective when more precision is needed The double data type may store 64 bits of decimal or floating point integers, making it a precision data type. It can thus store exactly twice as much information as a float. Similarly, the double is a built-in data type. That we are locked into its current name and meaning for the duration of the program's execution. Note, however, that it is noticeably slower than the float data type because of the greater amounts it can store. Being so large makes it slow, but it can store numbers with 15 or 17 digits on either side of the decimal point with no trouble at all.

Explanation on String Non-Primitive Data Type

Now we discuss about the non primitive data types , Because of their reliance on objects, on-Primitive data types in.NET are also referred to as reference types. Non-primitive data types are not predefined and must be constructed by the programmer. In contrast to primitive types, which cannot have null assigned to them, non-primitive types are objects. In contrast to primitive data types, all non-primitive data types have the same size and use the same amount of disc space.

So we will discuss String data type which belong to the category of non primitive datatype . A string is a data type utilised in programming, such as an integer and floating point unit but is utilized to show text rather than numbers. String is a reference data type or basic data type, depending on its creation. It is generally considered to be a special reference data type

Explanation on Non-Primitive Object Data Type

Object also belongs to the category of non primitive data type , Simply said, an object is a chunk of memory that has been allocated and set up in accordance with its design. Many instances of the same class may be generated by a programs. Instances (or objects) can be kept in a named variable or an array or collection. These variables are used by the client code to make method calls and read/write to the object's public properties. Typical code written in an OO language like C# consists of a collection of objects that communicate with one

another in real time. An object is an abstract data type with the joining of polymorphism and inheritance. An object is a complex data type that permits us to store groups of data

SELF-ASSESSMENT QUESTIONS - 1

1. Any holder or container with a name to hold some value is called_____ .
2. An earlier version of VB .NET supported the default variable as _____.
3. _____ is required for internal conversion when you use variant data type.
4. Byte data type takes the value range of - Unsigned 0 to 255. State [True/False].
5. _____ operators compare the relationship between given string or numbers.

3. OPERATORS

Operators are nothing but the symbols which insist the compiler to execute or perform specific logical or mathematical operations. In VB.NET, the execution order of several Operators is controlled by the Operator precedence. VB .NET has huge number of operators that supports to do arithmetic and logical operations. We are now going to discuss the most commonly used operators.

- Arithmetic operators
- Comparison operators
- Logical/Bitwise operators
- Assignment operators

Arithmetic operator

It is a mathematical operation calculated between any two operands. These operators can be used in expressions to perform sequence of calculations. Following table 3.2 shows the various arithmetic operations supported by VB .NET, you can assume here the variable A holds the value 2 and the variable B holds the value 7.

Table 3.2: arithmetic operators

Operator	Description	Example
^	Raises one operand to the power of another	B^A will give 49
+	Adds two operands	A + B will give 9
-	Subtracts second operand from the first	A - B will give -5
*	Multiply both operands	A * B will give 14
/	Divide one operand by another and returns a floating point result	B / A will give 3.5
\	Divide one operand by another and returns an integer result	B \ A will give 3
MOD	Modulus Operator and remainder of after an integer division	B MOD A will give 1

Comparison Operator

These operators compare the relationships between given string or numbers and have the value, if the condition is true then 1 otherwise 0 for false. String will be generally compared

by taking the character one by one from each of the string. The table 3.3 is listing of comparison operators supported by VB.NET language.

Table 3.3: Comparison operator

Operator	Description	Example
==	Two operands values are verified for its equality.If the condition is yes then the result is true.	(A == B) is not true.
<>	Two operands values are verified for its non-equality. If the condition is yes then the result is true	(A <> B) is true.
>	Verify whether the value of left operand is greater than the value of right operand, if yesthen the result is true.	(A > B) is not true.
<	Verify whether the value of left operand is less than the value of right operand, if yes then the result is true.	(A < B) is true.
>=	Verify whether the value of left operand isgreater than or equal to the value of right operand, if yes then the result is true.	(A >= B) is not true.
<=	Verify whether the value of left operand is lesser than or equal to the value of right operand, if yes then the result is true.	(A <= B) is true.

The following are the operators that VB.NET supports, in addition to the standard ones shared by nearly all programming languages.

IS operator: Compares two objects reference variables and determines if that two object references refer to the same object without performing value comparisons. If object1 and object2 both refers to the same object instance, then the result is True; otherwise, result is False.

Is Not Operator – Also compares two object reference variables and determines if two object references refer to different objects. If object1 and object2 both refer to the exact same object instance, result is False; otherwise, result is True.

Like Operator – This operator matches a string against a pattern.

Logical and Bitwise operators

Below are the listed logical operators supported by VB .Net here we need to assume that the variable A has the Boolean value as True and the B has the value as False. Table 3.4, listing the logical and bitwise operators.

Table 3.4: logical and bitwise operators

Operator	Description	Example
And	It is one of the logical and bitwise AND operator. Here If both the operands are true then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A And B) isFalse.
Or	It is the logical as well as bitwise OR operator. If anyof the two operands is true then the condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A Or B) is True.
Not	It is logical as well as bitwise NOT operator. Usedto reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	Not (A And B) is True.
Xor	It is the logical as well as bitwise Logical Exclusive OR operator. It returns True if both expressions are True or both expressions are False; otherwise it returns False. This operator does not perform short-circuiting, it always evaluates both expressions and there is no short-circuiting counterpart of this operator	A Xor B is True.
And Also	It is the logical AND operator. It works only onBoolean data. It performs short-circuiting.	(A And Also B)is False.
Or Else	It is the logical OR operator. It works only onBoolean data. It performs short-circuiting.	(A Or Else B)is True.
Is False	It determines whether given expression is False.	
Is True	It determines whether given expression is True.	

Bit Shift operator

Bitwise operations will be executed on the bits to perform bit operation, truth table for &, | and ^ are listed below.

P	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assignment Operator

Table 3.5 listing the assignment operators available in VB.NET

Table 3.5: Assignment operators

Operator	Description	Example
=	Simple assignment operator, Assigns the values from right side operands to left side operand	C = A + B will assignvalue of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent toC = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent toC = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent toC = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand(floating point division)	C /= A is equivalent toC = C / A
\=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand (Integer division)	C \= A is equivalent toC = C \A
^=	Exponentiation and assignment operator. It raises the left operand to the power of the right operand and assigns the result to left operand.	C^=A is equivalent toC = C ^ A
<<=	Left shift AND assignment operator	C <<= 2 is same asC = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same asC = C >> 2

&=	Concatenates a String expression to a String variable or property and assigns theresult to the variable or property.	Str1 &= Str2 is same asStr1 = Str1 & Str2

Other Functions supported by VB .NET

- **Address Of:** This function returns the address value of a given procedure.

  Example: Add Handler Button1.click, Addreess Of Button1_click

- **Await:** Until the awaited task completes it suspend the execution of procedure.

  Example:

  Dim result as res1 = Await AsyncMethodThatReturnsResult() Await AsyncMethod()

- **Get Type:** Returns the Type object for the given object like methods, events, properties etc.

  Example: MsgBox(GetType(Integer).ToString())

- **Function Expression:** Declares code and parameter of the function that has lambda expression.

  Dim add5 = Function(num As Integer) num + 10 Console.WriteLine(add5(10))

SELF-ASSESSMENT QUESTIONS - 2

6. _____ is the logical AND operator works only on Boolean data to perform short-circuiting.
7. _____ returns the Type object for the given object like methods, events, properties.

4. DECISION MAKING /CONTROLLED STATEMENTS AND LOOPS

Decision making statements are also called as control statements which is required for any programming language to support the programmer to decide the block of statements needs to be executed when the condition is true and the other blocks will be executed when it is false. In order for a programme to make a decision, the programmer must first specify one or more conditions to be evaluated or tested, followed by a statement or statements to be executed if the condition is determined to be true, and, optionally, other statements to be executed if the condition is determined to be false.

Features of controlled statement:

We will briefly see the features of control statements , In the control statement, we will use if- Then, if Then Else, if Then ElseIf and the Nested Select Case

- Statements are used to direct the flow of data by testing whether or not a predefined condition has been met; these statements can be bound to a single condition, or multiple conditions can be specified at once.
- If the specified condition evaluates to true, the statement or block is performed in accordance with that condition; otherwise, other statements are carried out.

Following are the list of conditional statements/ Controlled statements supported by VB.NET

- If ….. Then statement
- If……Then…Else statement
- Nested if statements
- Select Case statement
- Nested Select Case statement

Now we are going to discuss the if construct with an example that gives an idea about the if Then and the if Then Else statements

```
Sub Main()

Dim a As Integer = 100
```

```
If (a < 20) Then

Console.WriteLine("a is less than 20")

Else

Console.WriteLine("a is not less than 20")

 End If

Console.WriteLine("value of a is : {0}", a)

 Console.ReadLine()

End Sub
```

In the above example the variable a is declared and assigned value with

100. The if construct checks for the value of a is less than 20 if the condition is True it prints "a is less than 20". But as per this example, the condition is false so it enters into the else portion and prints "a is not less than 20". Further it comes out from the construct and prints the original value of a and stops the process.

We will see one more program below that depicts the nested if construct.

```
Sub Main()

Dim a As Integer = 100

Dim b As Integer = 200

If (a = 100) Then

        If (b = 200) Then

        Console.WriteLine("Value of a is 100 and b is 200")

 End If

End If
```

```
Console.WriteLine("Exact value of a is : {0}", a)

Console.WriteLine("Exact value of b is : {0}", b)

Console.ReadLine()

End Sub
```

In the above program a and b are the two integer variables declared locally. Here, the if constructs followed by the variable declaration check for the value of a is 100 and the result is a Boolean type. If the result is True then the control will enter in to one more if construct called the nested if. If the nested if boolean condition results True it prints the statement on the console. If the first construct result or the inner/nested if result is false then the statement come out from the if construct and prints the original value of a and b variables.

Select Case

This construct allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each select case.

Select [Case] expression [Case expressionlist

[statements]] [Case Else

[elsestatements]]

End Select

expression: is an expression that must evaluate to any of the elementary data type in VB.Net, i.e., Boolean, Byte, Char, Date, Double, Decimal, Integer, Long, Object, SByte, Short, Single, String, UInteger, ULong, and UShort.

expression list: List of expression clauses representing match values for expression. Multiple expression clauses are separated by commas.

statements: statements following Case that run if the select expression matches any clause in expression list.

else statements: statements following Case Else that run if the select expression does not match any clause in the expression list of any of the Case statements.

Loop statements

Generally the program starts executing statements sequentially but there are situation where the programmer wanted to execute the set of statements for repeated number of times. So the languages supports by providing various loops construct to support the complex situation. Here loop statements execute either a single or group of statements for a specified number of times. Following are the various loop construct supported by VB

.NET.

- Do Loop
- For …. Next
- For Each…Next
- While…..End While
- With….End with
- Nested Loops

Do Loop

This loop executes the single or group of statements until the Boolean condition is True or it becomes True, also the loop can be terminated at any point of time using Exit Do statement. Below are the two construct for Do loop with entry level and exit level condition.

Do { While | Until } condition

Statements

Exit Do

Loop

Do

Statements

Exit Do

Loop { While | Until } condition

Now we will see one example for Do Loop using exit condition. This program starts the loop by printing the variable "a" with the initial values as 10, and continues until the value of a reaches 20. Here the condition checking is at the exit level so first time the loop will be executed and checks for condition to proceed further.

```
Sub Main()
Dim a As Integer = 10
 Do
Console.WriteLine("value of a: {0}", a)
a = a + 1
Loop Until (a = 20)
Console.ReadLine()
End Sub
```

Output : Value of a:11

-

-

Value of a: 19

For Next

This loop executes for single or group of statements for a specified number of times, here the loop index counts the number of iterations. This loop can be terminated at any point of time using Exit For statement.

For counter [As datatype] = start To end [Step step]

[statements]

[Exit For]

[statements] Next [counter]

For Each loop is used exclusively for accessing and manipulating all elements in an array or in VB.Net collection.

For Each element [As datatype] In group

[statements]

[Exit For]

Next [element]

You can see the example below, For Each loop statement that reads the array value one by one and prints the same. This loop continues to work until it reads entire items from an array.

```
Sub Main()
        Dim anArray() As Integer = {1, 3, 5, 7, 9}
        Dim arrayItem As Integer
        For Each arrayItem In anArray
          Console.WriteLine(arrayItem)
        Next Console.ReadLine()
        End Sub
```

Output: 1, 2, 5, 7, 9

While End While

It is yet another loop executes single or group of statements until the given condition is true. Main key point with while loop is that it starts with the condition of checking the conditional statement, if the condition results as false the loop statements will be skipped and the statement after the loop body will be executed.

While condition

[statements]

[Exit While]

[statements]

End While

Example:

```
Sub Main()
Dim a As Integer = 10
While a < 20
    Console.WriteLine("value of a: {0}", a)
     a = a + 1
    End While
    Console.ReadLine()
    End Sub
```

Output: Value of a:10

-

- *Value of a:19*

In the above example the variable a is assigned with the value of 10. While loop starts with the conditional testing of whether the value of a is less than

20. As per this example the condition is True so it starts execute the content of the loop. It prints the value of a and then the value of a is incremented with one. When the value of the

variable a reaches twenty, then the condition becomes false and it comes out of the loop and end the process.

With End With

If you take the *With End with construct* it is not exactly a looping construct. It does the execution of series of statements that repeatedly refers to a single structure or object.

```
With object
        [ statements ]
End With
        Public Class student
        Public Property Name As String
        Public Property course As String
        Public Property semester As String
End Class
Sub Main()
Dim stud As New student
With stud
     .Name = "Mr.X"
     .course = "MCA"
     .Semester = "Five"
      End With
     With stud
 Console.WriteLine(.Name)
```

Console.WriteLine(.course)

Console.WriteLine(.semester)

End With

Console.ReadLine()

End Sub

You can observe in the above program group of data is created using the same property and it is been referred by the With construct.

Nested Loops

As we discussed earlier nested are, building of loop construct inside the loop. VB .NET supports this concept with all the loops like Do, While For etc. Below you can find syntaxes pertaining to nested loops.

For counter1 [As datatype1] = start1 To end1 [Step step1]

For counter2 [As datatype2] = start2 To end2 [Step step2]

...

Next [counter2]

Next [counter 1]

While condition1

While condition2

...

End While

End While

SELF-ASSESSMENT QUESTIONS - 3

8. Statements does not fall in any the case in Select case fall in_____ .
9. DO

   Statements

   Exit Do Loop

   {while|Until} condition is a entry check statement. State [True/False]

5. ERROR HANDLING

VB .NET supports two types of error handling through which we can avoid the termination of program during execution. The broad categories are

- Unstructured error handling
- Structured error handling

Generally error happens during the run time also called as exceptions generated due to unexpected or abnormal condition during execution of code.

Unstructured error handling is executed with the help of On Error statement, generally placed at starting of the block of the code in order to handle all the errors which can be raised due to unexpected situation. All VB

.NET errors can be handled using Microsoft. Visual Basic. Information. Err namespace. When the procedure is called the handler will be set with Nothing. Better to have single On Error statement is one procedure more than disable other previous handlers defined in that procedure.

On Error statement is used to either, enable, disable or specify to branch to the location.

On Error {Go To [line | 0 |-1] | Resume Next}

Here Go To line is used to enable the error handling routine, located at the starting of the line, it may be either label or number. When the specified line number or label does not occur in the procedure it raises the compiler error. To avoid the un expected behavior when no error occur we can place Exit Sub, Exit property or Exit Function just near the line number or label.

GoTo 0: Disables the enabled error handler that is defined within the current procedure and resets it to Nothing.

GoTo -1: Disables the enabled exception that is defined within the current procedure and resets it to Nothing.

Resume Next: Moves the control of execution to the statement that follows immediately after the statement that caused the run-time error to occur, and continues the execution from this point forward. This is the preferred form to use to access objects, rather than using the On Error GoTo statement.

Below you can find the situation where and how these error techniques can be handled.

```
'Generate an error if the user cancels.:
 dlgOpenFile.CancelError = True
'Ignore errors for now :
On Error Resume Next
'Present the dialog :
dlgOpenFile.ShowOpen
'See if there was an error:
If Err.Number = cdlCancel Then
'The user canceled. Do nothing:
Exit Sub
'Unknown error. Take more action.
 ElseIf Err.Number <> 0 Then
End If
'Resume normal error handling:
On Error GoTo 0
```

Structured error handling helps the programmers to handle the unexpected errors efficiently and provide support to the programmer to develop application efficiently and for easier maintenance for the same. Structures error handling used the Try.. Catch…Finally

statement to handle the errors. The code that is suspected for error generation during execution can be put inside this block. If this block produces an error during execution It tries to match with the error in the catch block. If the matches found, the control will be transferred to the initial line of the catch block. The Finally, exist as a last statement in the Try catch Finally block will be executed immaterial of errors found or not. If there this no error match with the catch block the Finally statement execute to propagate outer statements.

The example gives you an idea about the role of Try…Catch…Finally statements.

```
Public Sub TryExample()
    Dim x1 As Integer = 5
    Dim y1 As Integer = 0
    Try
    X1 /= y1 ' Lead to "Divide by Zero" error.
    Catch ex As Exception When y = 0 ' To Catch error.
    MsgBox(ex.toString)
    Finally
    Beep() ' sound after processing of error.
    End Try
    End Sub
```

After clearly understanding what exactly is an error and how does an error occurs, lets see the types of visual basic errors.The basic error is categorized into three types such as compile time error , runtime error and logical time error . Compile time error are the error which is found or appears during the time of compilation, runtime error is the error which is considered during the runtime that is when program gets starts its execution and finally

logical error are those errors were there is a logically error in the program or the logic is wriiten wrong so that it effects the flow of the program.

Futher the visual basic error are broadly classified as

1. Compile Time Error
2. Run- Time Error
3. Logical Error

Compile Time Error:

An error discovered during the compilation process is known as a compiler error.An error is generated at compilation time if the syntax rule is violated. An error will be generated at compile time if you use the wrong punctuation or put it in the wrong place. For example Using the wrong punctuation or placing the punctuation at wrong place can create compile time error. In short we can term compile time error as errors in semantics or syntax are the most common kind of compile-time mistakes. Until all of the mistakes have been fixed, the compiler will not allow the programme to run. After all bugs have been fixed, the compiler will produce the executable file. The compiler will issue a syntax error if the programmer violates the rules of the language they are using.

Example

```
Void main ()
{       int  = 5;
int  = 8;
printf ("%d", (a,  b))
}
```

Output- error : expected ';'before '}' token

Compiler Error handling Technique:

The compiler will issue a syntax error if the programmer violates the rules of the language they are using and When the compiler cannot make sense of the statements, a semantic error occurs. Both belongs to compile time error, A compile error is normally easy to fix, as the VBA compiler pops up a message box, which provides data on the nature of the error. For example, if you get the message "Compile error: Variable not defined" when you try to run your code, this shows that you are referencing a variable that has not been defined in the present scope.

Run Time Error: Errors that occur during execution but after compilation are called runtime errors. Instances of runtime mistakes include division by zero and similar operations. Unfortunately, the compiler does not flag these kinds of mistakes, thus they can be difficult to find. Run-Time Errors happen during the execution of your code, and effect the code to stop running. Errors can as well come up while the execution of a program and are named run-time errors. An example of a run-time error might be the wrong data type given for a field in a database or trying to divide by zero. In this case, the compiler does not detect the error, so it cannot prevent the code from the execution.

Run time error handling technique: We will now see the techniques for handling the runtime error, the error is also usually comparatively easy to fix, as you will be given details of the nature of the error, and given the location where the code has stopped running. For example , if your code attempts to divide by zero, you will be presented with a message box, which states "Run-time error '11': Division by zero".

Let us see how the run time error is handled by Visual Basics Applications , VBA depends on the structure for any of the application , particularly we may be given the option to debug the code, by clicking the debug button on the application the debug message box causes the line of code that has generated the runtime error to be highlighted in your VBA editor. If code is more complex, you can gain further information on the reason for the VBA error by looking at the values of the variables in use. This can be done in the VBA editor by simply hovering your mouse cursor over the variable name, or by opening the local variables window. Best

example to discuss about the run time error is error code 13 that indicates type mismatch . We can debug the same using debug code.

Example-

```
Void main()
{
int n=22, div = 0;
// wrong logic
div = n/0;
printf("result = %d", div);
}
```

Output- warning division by zero . Div = n/0;

Common Types Of Run Time Error Codes:

After knowing what is runtime error and how can visual Basic Applications handle the runtime error, we will look into the common types of runtime errors:

- error code 5 is one of the common types of which indicates invalid procedure call, similarly
- error code 7 indicates out of memory,
- Error code 9 tells subscript out of range ,
- Error code 11 indicates division by zero error has occurred ,
- Error 13 tells type mismatch error and finally
- Error code 53 indicates file not found error.

These are some of the common run time errors were the programmer should know the meaning of the error code and what exactly it means.

Trapping Runtime errors:

After the error handling code has run, the programmer can request that the VBA code resumes from the point of the error, or alternatively, the macro can be terminated cleanly A professional approach would be to instead of letting your macro crash, to "trap" the error

and create VBA code to handle it so that your macro exits properly. The On Error and Resume statements in Visual Basics Application can assist with error handling at runtime. A runtime error is captured by these statements.

Example:

- >Code::: Sub procedure to set the supplied values, Val1 and Val2 to the values

' in cells A1 and B1 of the Workbook "Data.xls" in the C: \ directory

- Sub Set_Values(Val1 As Double, Val2 As Double)
- Dim DataWorkbook As WorkbookOn Error GoTo ErrorHandling
- ' Open the Data WorkbookSet DataWorkbook = Workbooks.Open ("C:\Documents and Settings\Data")
- ' Set the variables Val1 and Val2 from the data in Data Workbook
- Val1 = Sheets("Sheet1").Cells(1, 1)

Val2 = Sheets("Sheet1").Cells(1, 2)

- DataWorkbook.CloseExit SubErrorHandling:
- ' If the file is not found , ask the user to place it in to

' the correct directory and then resume

- MsgBox "Data Workbook not found;" & _

"Please add to the workbook to C:\\Documents and Settings and click OK"

Resume End Sub Code>

When we look into the above example of handling the runtime error, considering a situation the code tries to open the Excel File 'Data' and if it fails to get the specified file, prompts the user to place the data file into the right folder. Once the user does this and clicks OK, the code is again run and a further trial is created to open the file. If desired, instead of re-trying the file, the Sub procedure could be terminated at this point, by utilizing the Exit Sub command. With the above scenario we clearly understood how run time error is handled.

3. **Logical Error:**

The inconsistencies in the way the program is supposed to work are known as logic mistakes. The developer may have made an incorrect assumption, or the user may have triggered an undesirable or unanticipated consequence . Logical mistakes are the most challenging to detect and remedy, as they might cause the macro to behave in unexpected ways or produce inaccurate results. The VBA compiler does not have an equivalent of the way it handles build and runtime errors, where it can locate the problem and "point to" it.

Example for logical error:

Let us discuss the example of logical errors , In computer programs, this error can occur in many different forms. A simple example is when a programmer assigns the wrong value to a variable. These errors are very difficult to track down because the compiler can not provide assistance. In a second example we consider dividing a number by zero or a script that is programmed and which reaches into infinite loop, one example to discuss let us consider the logic, the user needs to check divisibility by 10. But, in the 'if' condition instead of using '==' it is '=' which is the assignment operator. Therefore, we need the '==' operator to check if the remainder is equal to 0 or not. So this cauase logical error.

Error handling techniques of logical errors:

.To help you detect and repair logical issues in your Visual Basic Application code, the Excel Visual Basic application editor includes a number of debugging tools. Error handling makes it simpler to include input specifications into the code, eliminating the need to reference the design during development.

Advantages of error handling technique:

- The procedure that consists of fixing software bugs
- . Error management is simplified as a result of its emphasis on detection and correction.
- Error management aids in preserving the smooth progression of program operation.

6. CLASSES AND OBJECTS

Class is defined as "A container for data and code. The data within the class can be accessed with properties. The code is referred to as methods".

Object is defined as "An instance of a class in memory. An instance is created using a Dim statement and the New keyword"

The definition of class will gives you a blueprint of a data type. Actually whenever you define a class doesn't mean that you are defining a data. It indicates the content of the object of the class and various operations that can be performed unit this objects. The data members and the functions to perform operations on these data are the members of the class. Objects are nothing but the instance of a class. You can see below the class construct that can be defined in VB .NET language. It starts with the key word called class and followed by the name of the class. Inside the body list of members will be declared and ends with the End Class statement.

[<attributelist>] [accessmodifier] [Shadows] [MustInherit | NotInheritable

] [Partial] _

Class name [(Of typelist)]

[Inherits classname]

[Implements interfacenames]

[statements]

End Class

attribute list: attributes list of the table all the square brackets in the syntax indicates, it is optional.

Access modifier: Has three types of access specifier like private, protected and public defines the scope of the class member.

Shadows: Declared and hidden as overload member in the name class.

Must Inherit: Indicates you cannot create object for this class only this class can be used for inheritance.

Partial: specify the fractional definition of the class.

Inherits: Indicates which class is inherited.

Implements: used to specify the interface of the class inherited from

In VB .NET environment if you observe, whenever you create a form it comes with Public Class Form1 in right top of the window. The form what you have started itself a class. Form1 can be name of the class. Whenever you add a control to the form, you are adding members in to the form class. When you initiate or start the form VB does the instantiation it is converting in to an object.

SELF-ASSESSMENT QUESTIONS - 4

10. _____ and_____are the two types of error handling statements available in VB. NET.
11. The attribute Shadows Indicates that you cannot create object for that class. State [True/False].

7. SUMMARY

- Each variable in VB .NET has a specific type that defines the size and layout of the variable in memory.
- Among all the data types Boolean data type is considered as a simple one.
- Operators are the symbols which insist the compiler to execute or perform specific logical or mathematical operations.
- AddressOf, Await, GetType and Function Expression are the special data types supported in VB .NET
- VB .NET supports two types of error handling through which we can avoid the termination of program during execution are structured and unstructured error handling techniques.
- Class is defined as "A container for data and code. The data within the class can be accessed with properties. The code is referred to as methods".
- Object is defined as "An instance of a class in memory. An instance is created using a Dim statement and the New keyword"

8. TERMINAL QUESTIONS

1. List and explain the list of data types supported by VB .NET.
2. Explain the comparison and logical operators available in VB .NET.
3. Discuss the decision making statements with example.
4. List and explain the looping statements.
5. Discuss in detail the error handling techniques.
6. Brief about classes and objects.

9. ANSWERS

Self Assessment Questions

1. Variable
2. Variant
3. Casting
4. True
5. Comparison
6. And Also
7. Get Type
8. Else statement
9. False
10. Structured and unstructured
11. False

Terminal Questions

1. Boolean, Numeric, single, double, char, string etc are the various data types can be used in VB.NET. For more details refer section 3.2.
2. Equal to, note qual to, less than, greater than, and or, not are the various comparative and logical operators available in VB. NET. For more details refer section 3.3.
3. If, if then else, select, nested if, nested select are the various decision making statements available in VB.NET. For more details refer section 3.4.

4. Do, For, While, With are the few looping statements of VB.NET. For more details refer section 3.4.
5. Structures and unstructured are the two techniques used for error handling. For more details refer section 3.5.
6. The definition of class will gives you a blueprint of a data type. Actually whenever you define a class doesn't mean that you are defining a data. For more details refer section 3.6.

10. E-REFERENCE:

- http://www.tutorialspoint.com/vb.net/vb.net_operators.htm
- http://support.microsoft.com/kb/311326
- http://www.vb-helper.com/err_sample_text.html
- http://msdn.microsoft.com/en-us/library/fk6t46tz(v=vs.71).aspx
- NOTE