Programming Constructs:

Variables:

- is a name that is used to store data in the memory.
- we can access the data from the computer through using the variable names only.
- The values stored in a variable can change during the execution of the program
- Each variable is associated with a specific ~~type~~ data type
- It tells the Comp! the amount of memory that can be provided for a variable
- The following are the rules for naming var.
  (i) 1st char should be alphabet
  (ii) It contains upto 40 chars
  (iii) It may include alphabets, numbers and (_)
  (iv) keywords should not be used.

Data Types:

- VB use the following data types. They are

1. Bodean - Used to store a value T or F
2. Integer - Used to store the whole numbers.
3. Long - Used to store the higher value of whole nos
4. Single - " " " fractional nos
5. Double - " " " higher value of fractional
6. Currency - Used to store large nos in integer format
7. Date - " " Date & time values
8. Object - " " a object
9. String - " " a series of chars
10. Variant - " " any type of data like integer, float, char, date
11. Byte - " " single, unsigned 8 bit nos.

Variable Declaration:

- We should declare our variables in a Program.

- Syn: Dim (Var name) As (data type)

eg: Dim Name as String

Scope of a Variable:

- It determines the lifetime of a
- In VB, there are 4 levels of Sc

1. Procedure level:

- The scope of a Procedure level Vari is only within the procedure where it is declar
- It uses the Dim statement for declaration
- It ~~should~~ does not retain their value after execution of the Procedure. completed.
- eg: Dim BookCost as Double

2. Procedure level, Static

- This is similar to Procedure level but it retains the value after execution of the Procedure is completed.
- Here we include Static keyword instead of Dim.
- But the scope of a Var is within that Procedure only.
- eg: Static BookCost as Double

- eg: Static BookCost as Double

3. Form level:

- Form level variables are available to all procedures within that form.
- It retains its value within that form only
- It should be declared in the declarations Part of the code window
- Here also we use the Dim keyword for declarations.
- eg: Dim NoofBooks as Integer

4. Global level:

- The Scope of the variable is throughout the application.
- We can use the global variables at any form & any Procedure
- It retain their value throughout the application.
- It is also called as Module level variable
- It should be declared in the declarations Part of the Module's code window.
- we must use the Global keyword.
- eg: Global NoofBooks as Integer

What is usage of Scope of a variable:

- Some Var may be required for ce Calculations only.
- Once the calculation is completed the var may not be use.
- In such situations it may be waste of memory space to retain such var.
- so, to avoid the locking of memory Space we use the scope of a var. (Procedure level) &
- The Procedure level var can't be accessed outside of it.

- if var has same name in two or more places then also it should be identified its scope only
- For eg: if variable Book is defined as global as well as Procedure level means within that procedure the local variable book is used.

outside of the procedure the global var book is used.

Eg: Module 1

~~Dim~~ Global x as Integer

Form1

Dim y as Integer

Sub Proc1()

Dim A as Integer

:

End Sub

Sub Proc2()

~~Dim~~ Static B as Integer

End Sub

Form 2

Dim z as Integer

Sub Proc3()

Dim c as Integer

:

End Sub

- x can be accessed by all the forms & Procedures.
- y can be " " only form1 and Procedures of form1
- A can be accessed by only Proc1() and it does not retain any value after Proc1() completed.
- B can be accessed by only Proc2() but it can retain its value after completion of Proc2()

Constants:

- Constants are name, its value does not change during the execution of a Program.

- Syn: for declaration of Constant

[Public | Private] Const Const-name [as datatype] = expression

eg: Const PI = 3.14

Symbolic Constants:

- Sometimes it is difficult to assign numeric value to Constants.
- In such situations, VB assigns some names to most widely used values. these are called as symbolic constans.

- For eg:

Form1. BackColor = &H FF000000

Instead of this we can use

Form1. BackColor = Vb Blue

Operators & Functions in VB

refer to Book.

Arrays:

- An array is a set of values that have a same data type stored in a common name.

- Individual items of an array are identified with the help of index.

- Syn Dim arrayname(size) as datatype
- Eg: Dim a(9) as Integer.
- a has 10 elements starts with a(0) and ends with a(9)

- Eg: Dim b(10 to 20) as Integer
- b has 11 elements starts with b(10) and ends with b(20)

Multidimensional array

- Sometimes we want to store a data in table format, for eg class Time table such situations we use multidimensional arrays.

- for eg: Static MatA(9,9) as Integer
stores the two dimensional array.

- eg: Static matB (1 to 10, 1 to 10) as Int

Dim matC (9, 10) as Int

Visual Basic statements:

- Statements give the meaning to the Program.

- There are various statements are there. They are.

1) Assignment Statements
2) Comment "
3) Branching "
4. Looping "

1. Assignment Statements:

- This St is used to assign a Value to a variable or Compute any expr and store the computed value

- Syn Variable = Value (or) expression

- eg: a = 10
d = a+b*c

2. Comment Statement

- It is a non-executable st
- It is used to improve the understanding of a program.
- It ~~Comes~~ gives documentation Part of a Pgm.
- Comment line starts with ' or REM
- for eg: ' Program for addition
REM " "

3. Branching Statements:

- Branching statements are used to jump from one Part of the Program to another Part of the program.
- This transfer of control based on Certain Conditions. So these sts called as Conditional sts (or decision making sts

- The different types of Branching sts are

1. Simple If:

- Syn If (logical expression or condition)
  then statement.

- eg: If a > b then
  Print "a is bigger"

2. If.. then.. endif

- This st allows a block of multiple sts to be executed if the condition is true.

- Syn: If (Condition) then
  Statement 1
  Statement 2
  Endif

- ~~if the Con~~

3. If.. then.. else.. endif

3. If.. then.. else .. endif

- This st includes two blocks of sts.

- Syn If (condition) then

Statement 1
" 2

else

Stat 3
Stat 4

endif

- If the condition is true the st1 & st 2 are executed.

- it is false st 3 & st4 are executed.

4. If - then - elseIf:

- In the above st if the first condition is false means the else block will be executed.

- Suppose we want to check another condition for else block mean it is not possible.

- So, we need the if-then-else If

- This st includes many block of sts. but execute only one block depending upon the conditions.

- Syn If (Condition 1) then
  Stat 1
  else If (Condition 2) then
  St 2
  else If (Condition 3) then
  St 3
  else
  St 4
  end if.

eg:

If mark < 35 then
  Print "You have failed"
else if mark < 50 then
  Print "You have passed in 3rd class"
else if mark < 60 then
  Print "You have passed in 2nd class"
else
  Print "You have passed in 1st class"
end if

5. Select case:

- An alternative to the if-then-else is the select case st

- Syn Select case (variable)

Case Value 1

St 1

Case Value 2

St 2

Case Value 3

St 3

end Select.

- eg: Select (n)

case 1

Print "perform addition operation"

case 2

Print "peform subtraction"

end select.

- In the above example if n = 1 then St 1 will be executed if n = 2 then St 2 will be executed.

- If n is equal to other than 1 & 2 (ie) 3 means no sts will be executed.

- To avoid this we use else clause in select st. (ie) default st.

- eg: case 2
st2
case else
st 4
end select.

- In select st we have to give exact value
- we not give the condition or expression
- for eg: case 1
case n>1 is invalid.

4. Looping statements:

- looping sts allow repeat the execution of set of instructions again & again.

2. Do until Loop

- It is also a pre tested loop.
- set of sts repeatedly executed while Condition is false or until it becomes true

- Syn Do until <condition>
  St 1;
  St 2
  Loop.

3. Do Loop until

- In pre-tested loop the st may not be executed the condition is ~~true~~. false true
- But in this Do loop until the sts executed atleast Once then only it checks the Condition.
- If the condition is false means it go inside the loop otherwise it comes ok of the block.

Syn: Do

Syn: Do

St 1

St 2

Loop Until <Condition>

4. For - Next loop

- The above looping sts are used when we don't know the number of times it executed repetedly.

- If we know the no. of times we use For next loop

- Syn For Variable = initial value TO Final value [Step Step value]

St 1

St 2

Next variable

- eg: For I = 1 TO 100 Step 5

S = S + I

Next I