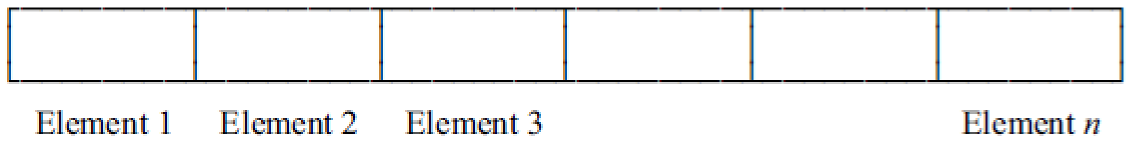
VISUAL BASIC PROGRAMMING- UNIT IV

UNIT IV: Arrays –Parsing Arrays to Procedures- Dynamic Arrays –Array Function –Control Arrays –Data Files –Processing A Data Files –Sequential File –Random Access File.

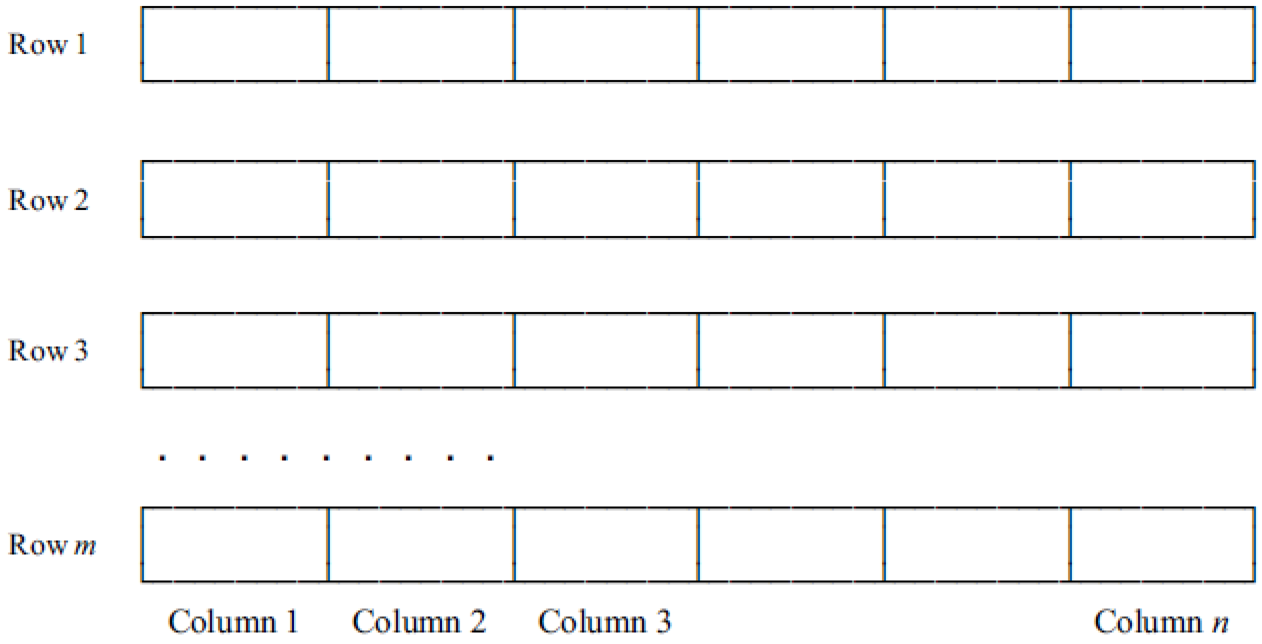
Arrays

- Many applications require the processing of multiple data items that have common characteristics, such as a set of numerical data items represented by *x*1, *x*2, . . ., *xn*. In such situations, it is often convenient to place the data items into an *array*, where they will all share the same name (e.g., x).
- The data items that make up an array can be any data type, though they must all be the same data type. (An exception is the variant-type array, where each data item may be of a different data type).
- Each individual array element (i.e., each individual data item) is referred to by specifying the array name followed by one or more *subscripts*, enclosed in parentheses.



One dimensional array

- A two-dimensional array as a *table*, where i refers to the row number and j refers to the column number.



Two dimensional array

- Higher-dimensional arrays, such as the three-dimensional array z(i, j, k), are formed by specifying additional subscripts in the same manner.

ARRAY DECLARATIONS

- An array must be declared before it can appear within an executable statement. The Dim statement is used for this purpose. This statement defines the *dimensionality.*
- Within the Dim statement, each array name must be followed by one or more integer constants, enclosed in parentheses. If several integer constants are present (indicating a multidimensional array), they must be separated by commas.
- To declare an array within a procedure, the Dim statement is generally written as

Dim *array name* (*subscript 1 upper limit, subscript 2 upper limit, etc.*) As *data type*

- Within a module (but outside of a procedure), array declarations are written as

Private *array name* (*subscript 1 upper limit, subscript 2 upper limit, etc.*) As *data type*
or
Public *array name* (*subscript 1 upper limit, subscript 2 upper limit, etc.*) As *data type*

- Each subscript normally ranges from 0 to the specified *upper limit*. Thus, the Dim statement for example:

  Dim c(10) As Integer

- The specification of a different lower limit can also be included within a Dim statement (or a Public or Private statement). In this case, the general form of the Dim statement is

  Dim *array name* (*subscript 1 lower limit* To *subscript 1 upper limit, subscript 2 lower limit* To *subscript 2 upper limit, etc.*) As *data type*

- Public and Private statements are written in the same manner
- The Option Base statement allows the lower limit for all arrays within a module to be changed to 1. This statement is written simply as

  Option Base 1

- Option Base must appear at the module level (not within a procedure), and it must precede any array declarations within the module.
- A Visual Basic module includes the following array declarations.

```
Option Base 1
DIM Customers(200) As String, Net(100) As Single, Sales(50, 100) As Single
```

- The elements of an array may be a user-defined data type rather than a standard data type. This is handled in the same manner as ordinary variables.
- In general terms, the data type definition is written as

```
Type data type name
        member name 1 As data type 1
        member name 2 As data type 2
        . . . . .
   End Type
```

The array declarations can then be written as

Dim *array name* (*subscript 1 lower limit* To *subscript 1 upper limit,*
subscript 2 lower limit To *subscript 2 upper limit, etc*.) As *user-defined data type.*

Here is a typical user-defined data type,

```
Type Customer
    CustomerName As String
    AcctNo As Integer
    Balance As Single
End Type
```

Once the data type has been defined, we can declare one or more variables of this data type, as follows.

```
   Dim OldCustomer(100) As Customer, NewCustomer(100) As Customer
```

Ex:

```
OldCustomer(5).CustomerName = "Smith"
NewCustomer(2).CustomerName = "Jones"
OldCustomer(i).AcctNo = 1215
NewCustomer(j).AcctNo = 1610
OldCustomer(i + 3).Balance = 44.75
 NewCustomer(i + j).Balance = 187.32
```

Example for array

```
Private sub combo1_click()
Dim water[5] as string
Water[0]=“thanner”
Water[1]=“vellam”
Water[2]=“water”
Water[3]=“pani”
Water[4]=“neelu”
Text1.text=water[combo1.listindex]
End sub
Private sub command1_click()
End
End sub
```

PASSING ARRAYS TO PROCEDURES

- Arrays can be passed to procedures as arguments, in much the same manner as ordinary variables are passed as arguments. If an argument is an array, however, an empty pair of parentheses must follow the array name. This requirement must be satisfied in both the procedure access and the first line of the procedure definition.

```
Dim x(2) As Single

Private Sub Command1_Click()
    x(1) = Val(Text1.Text)
    x(2) = Val(Text2.Text)
    Call Smallest(x())
End Sub
```

```
Sub Smallest(x() As Single)
    Dim Text As String

    If (x(1) < x(2)) Then
        x(0) = x(1)
        Text = "a is smaller (a = "
    ElseIf (x(1) > x(2)) Then
        x(0) = x(2)
        Text = "b is smaller (b = "
    Else
        x(0) = x(1)
        Text = "Both values are equal (a, b = "
    End If
    Call Message(Text, x(0))
End Sub
```

PASSING ARRAYS TO PROCEDURE

Arrays can be passed to procedure as arguments in the same manner as ordinary variables are passed as arguments.

Ex:

```
Dim x(10) as integer, n as integer
----------
---------
N=-----------
Call setup(x(),n)
Or setup(x(),n)
Private sum setup(v() as integer, n as integer)
Dim I as integer
For i=0 to n
   v(i)=i^2
Next I
End sub
```

DYNAMIC ARRAYS

Dynamic arrays is an array who's size can be changed at various places within a program.

To declare a dynamic array we use DIM statement followed by the array and empty pair of parenthesis.

Format:

```
Dim arrayname() as datatype
```

Within a module but outside a procedure we can use private or public statement.

```
Private arrayname() as datatype   (or)
Public arrayname() as datatype
```

To specify actual array size we use Re Dim statement.

```
Redim arrayname(subscript1 upper limit, subscript2 upper limit….) as datatype
(or)
Redim arrayname(Subscript1 lower limit to subscript1 upper limit, subscript2 lower limit to subscript2 upper limit) as datatype.
```

example

```
Dim x() as integer, n as integer
------
------
Redim x(10)
-----
-----
Redim x(30)
----------
------
N=--------
Redim x(n)
End sub
```

Array Related Functions

Function array:

Application:-

```
Dim x as variants
x=array(3,0.2,"ok")
```

Description:

x is a three element array containing x(0)=3, x(1)=0.2. x(2)="ok"

Function

Isarray

Application

```
if isarray(x) then
    msgbox "array defined"
else
    msgbox "array undefined"
```

Description

Return true or false value

Function

LBound

Application

Dim x(3 to 10)

Y=LBound(x)

Description

Return the subscript lower limit of the array argument

Function

Ubound

Description

Return a subscript upper limit of the array element.

Application

Dim x(3 to 10)

Y=UBound(x)

CONTROL ARRAYS

- ✓ Multiple controls of the same type can be grouped into an array in the same manner as a collection of data items. Such a grouping is known as control array.
- ✓ A control array can be created by placing a control within a form design window and assigning a value of zero to its index property. Then copy and paste the control resulting in a new control with the same name but the index value of 1.

Ex:

Form1	**properties**
Label1	enabled
	Index 0
Copy and paste the same label again	
Label 1	Enabled
	Index 1

DATA FILE CHARACTERISTICS

- ✓ Visual data basic consist of three different types of files
- ✓ Sequential file also called text file. Sequential files can be created by a text editor or word processor or by a visual basic program in order to access a particular line of text.
- ✓ You must start at the begin of the file and process through file sequentially until the line has been located.
- ✓ Random access file is organized into fixed length record and can be accessed directly by specifying the record number or record location.

Accessing and saving a file in Visual Basic(Common dialog control)

- ✓ For opening an existing file or saving a new file we need common dialog control it can be selected as follows:
- ✓ Project $\longrightarrow$ components $\longrightarrow$ microsoft common dialog control

To open an existing file

Commondialog1.showopen

- ✓ To access a file within the currently active folder the type of files can be specified with filter property.

To save a file we give

Commondialog1.showsave

To print the file

Commondialog1.showprint

Processing a datafile:

The following task must be carried out

a. open the datafile

b. process the datafile

c. close the datafile

- ✓ Opening the data file associates a file number with a named data file.
- ✓ It specifies the information about the data file such as the mode(input, output, append, random, read, write)
- ✓ Processing the data file generally involves reading the data item , modifying the data item, displaying the data item and then writing the modified data items.

SEQUENTIAL DATA FILES(TEXT FILES)

- ✓ A sequential data file is characterized by the fact that the data is stored sequentially as plain text.
- ✓ The text is organized into individual lines, with each line ending with ASCII line feed(LF) and carriage return(CF) characters.
- ✓ Each line of text can contain both numeric constants and strings in any combination.
- ✓ When opening a sequential file, the open statement is generally written in three ways depending whether the file is an input file, an output file or an append file.

  open filename for input as #n

  Open filename for output as #n

  open filename for append as #n

  Where n refers to the file number.
- ✓ Data items are usually read from a sequential data file via the input# statement. The statement is written in general form as:

  input #n, data items

Where n refers to the channel number and data items refers to the line of input data items separated by commas.

- ✓ Data items are usually written to sequential data file using print statement.

```
print #n, data items
```

Data items can be constants, variables, expressions, array elements, control properties.

RANDOM ACCESS(DIRECT) DATA FILES

- ✓ Random access data files consist of fixed length records, each of which is assigned a unique record number.
- ✓ An individual record can be accessed by referencing its record number.
- ✓ Information can be read from the record or written to the record as required by the individual application.

Random access files-Open Statement

openfilename For Random As #n Len=record length

(or)

Open filename As #n Len=record length

- The record length must large enough to accommodate each of the fixed-length data items that will be stored within the random access data file.
- Data items are read from a random access data file, one record at a time, using Get # statement.

Get #n, record number, data item (or)

Get #n, data item

- Data items are written to a random access data file, one record at a time, using Put# statement.

Put #n, record number, data item

(or)

Put #n, data item