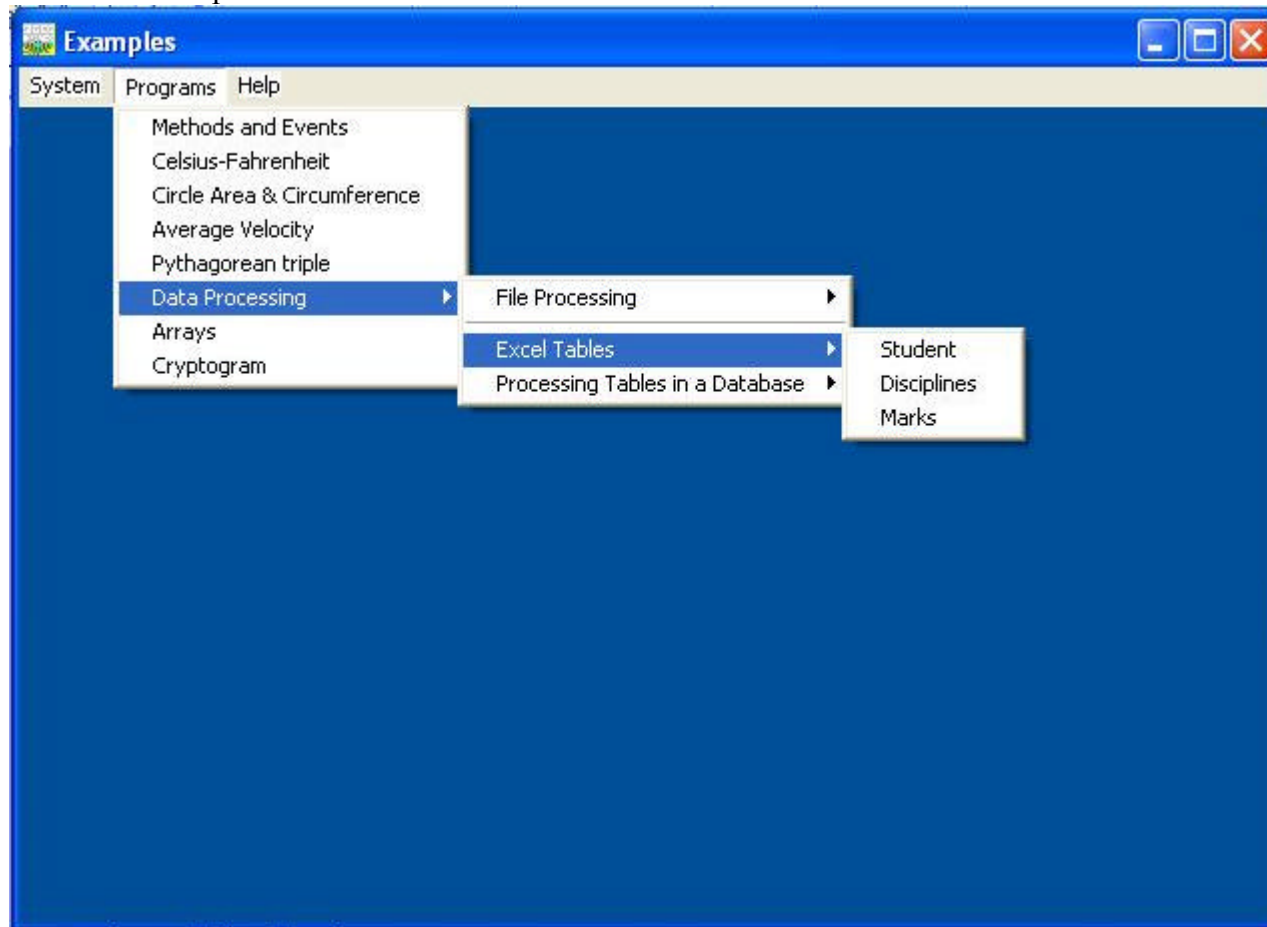


## VB Code Examples



The structure of the menu in the main form

## EXAMPLE 1

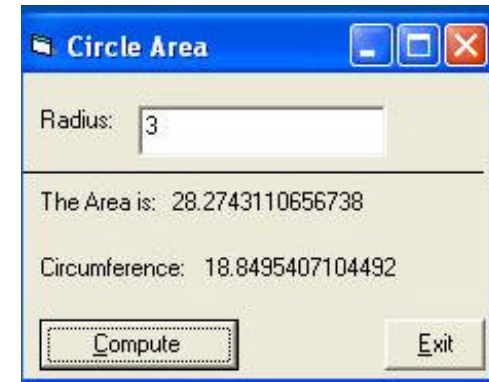
/\*-----\*/

Const Pi As Single = 3.14159

- ‘ Call examples: Attached to the Click event on Compute Button
- ‘ Area = CircleArea(Val(Trim(Radius)))
- ‘ Circumference = CircleCircumference(Val(Trim(Radius)))

Function CircleArea(Radius As Double) As Double  
    CircleArea = Pi \* Radius ^ 2  
End Function

Function CircleCircumference(Radius As Double) As Double  
    CircleCircumference = 2 \* Pi \* Radius  
End Function



## EXAMPLE 2

```
/*-----*/
'-----
' An example of manipulating memory arrays
' The memory array is filled automatically in the sequence.
' If the check box labeled "As Dynamic Array" is checked
' then the array redefined (enlarged) dinamically with preserving his actual content
'-----
' Exemplu de manipulare masive de memorie
' Vectorul de memorie definit este completat automat in secventa
' daca se bifeaza caseta de validare cu eticheta As Dynamic Array
' atunci are loc redefinirea sa cu pastarea vechiului continut
'-----
Private Sub ArrayOperations_Click()
    ArrayEx.Cls 'Clear the screen - form ArrayEx
    If IsDynArray.Value = vbChecked Then 'Call the function that corresponds to the case
        DynamicArrayEx
    Else
        FixedArray
    End If
End Sub

Private Sub DynamicArrayEx()
    '-----
    'This function shows how to use dynamic array
    'The values for the elements accessed in the array are filled automatically
    'In a real context of usage you may choose the right way for you to fill data
    '-----
    'The sentence that follows redefines the array called DynamicArray and reserved first by using a Dim
statement
    'in Modules, Module1
    ReDim DynamicArray(1 To 5) As Variant
    Dim i As Integer
    Print "Dynamic Array Content:"
```

The screenshot shows a Windows-style window titled "Arrays". It contains two tables of data and a checkbox at the bottom.

**Dynamic Array Content:**

| Element | Value          |
|---------|----------------|
| 1       | This is cell 1 |
| 2       | This is cell 2 |
| 3       | This is cell 3 |
| 4       | This is cell 4 |
| 5       | This is cell 5 |

**Dynamic Array Preserved Content:**

| Element | Value          |
|---------|----------------|
| 1       | This is cell 1 |
| 2       | This is cell 2 |
| 3       | This is cell 3 |
| 4       | This is cell 4 |
| 5       | This is cell 5 |

**New added cells content**

|   |                                  |
|---|----------------------------------|
| 6 | This is an added value in cell 6 |
| 7 | This is an added value in cell 7 |
| 8 | This is an added value in cell 8 |

At the bottom, there is a checkbox labeled "As Dynamic Array" which is checked, and a "Process" button.

```
Print "-----"
Print "Element      Value"
Print "-----"
```

```
For i = 1 To 5
    DynamicArray(i) = "This is cell " & i
    Print i, DynamicArray(i)
Next i
```

'The DynamicArray array is extended (from 5 to 8 elements)with preserving the older content  
ReDim Preserve DynamicArray(1 To 8) As Variant '

```
'The preserved content is displayed
Print "Dynamic Array Prezerved Content:"
Print "-----"
Print "Element      Value"
Print "-----"
For i = 1 To 5
    Print i, DynamicArray(i)
Next i
```

'The content of extension lines (6, 7 and 8) is filled automatically and displayed onto the screen

```
Print "-----"
Print "New added cells content"
Print "-----"
```

```
For i = 6 To 8
    DynamicArray(i) = "This is an added value in cell " & i
    Print i, DynamicArray(i)
Next
End Sub
Private Sub FixedArray()
    'This sample shows how to use the fixed arrays
    Dim i As Integer, j As Integer ' The inex variables
    Dim arrayA(3) As Integer 'An array of 4 integers
```

| Arrays                                                                           |                   |
|----------------------------------------------------------------------------------|-------------------|
| Array A Content:                                                                 |                   |
| Element                                                                          | Value             |
| 1                                                                                | 100               |
| 2                                                                                | 101               |
| 3                                                                                | 102               |
| 4                                                                                | 103               |
| Array S Content:                                                                 |                   |
| Element                                                                          | Value             |
| 1                                                                                | This is Element 1 |
| 2                                                                                | This is Element 2 |
| 3                                                                                | This is Element 3 |
| 4                                                                                | This is Element 4 |
| Element 3 from array A: 102                                                      |                   |
| Element 2 from array S: This is Element 2                                        |                   |
| <input type="checkbox"/> As Dynamic Array <input type="button" value="Process"/> |                   |

```

For i = 0 To 3
    arrayA(i) = 100 + i 'Assigning a value to element i
Next i
Print "Array A Content:"
Print "-----"
Print "Element    Value"
Print "-----"
For i = 0 To 3
    Print i + 1, arrayA(i)
Next
Dim arrayS(3) As String 'An array of 4 strings
For i = 0 To 3
    arrayS(i) = "This is Element " & (i + 1) 'Assigning a value to element i
Next i
Print "Array S Content:"
Print "-----"
Print "Element    Value"
Print "-----"
For i = 0 To 3
    Print i + 1, arrayS(i)
Next
Dim arrayV(1 To 2) As Variant 'Declares an array of two elements
arrayV(1) = arrayA() ' Places the entire array A as an element
arrayV(2) = arrayS()
Print "Element 3 from array A:", arrayV(1)(2)
Print "Element 2 from array S:", arrayV(2)(1)

End Sub

```

### EXAMPLE 3

/\*-----\*/

Private Sub ComputeAV\_Click()

'This code sequence realizes a simple validation of required values

'by converting them from text to value - Val(Trim(x))

'and by checking the relationship that must be between them

'the detected errors will produce the displaying of the appropriate message

'the positioning of the cursor in the appropriate field and exits from routine

If Val(Trim(p2)) < Val(Trim(p1)) Then

MsgBox "p2 must be greater than p1 !", vbOKOnly

p2.SetFocus

Exit Sub

End If

If Val(Trim(t2)) <= Val(Trim(t1)) Then

MsgBox "t2 must be greater than t1 !", vbOKOnly

t2.SetFocus

Exit Sub

End If

'If no error detected then the function computing the average velocity invoked

AvgVelo = AverageVelocity(p1, p2, t1, t2)

End Sub

Function AverageVelocity(p1 As Variant, p2 As Variant, t1 As Variant, t2 As Variant) As Variant

AverageVelocity = (p2 - p1) / (t2 - t1)

End Function

Private Sub Label5\_Click()

'If label "Average velocity" clicked the formula of computation displayed in a dialog box

MsgBox "Computes the average velocity" & Chr(10) & Chr(13) & " (p2-p1)" & Chr(10) & Chr(13) & \_  
" -----" & Chr(10) & Chr(13) & " (t2-t1)", vbOKOnly + vbInformation

End Sub

Average Velocity

p1 3 t1 5

p2 6 t2 9

Average Velocity: 0.75

Compute Average Velocity Exit

#### EXAMPLE 4

/\*-----\*/

' This is a simple encryption problem of a character string  
' It acts only on Capital Letters by replacing them in the original message  
' with the designed character (encrypted)  
' The problem can be extended to process all letters does no matter the  
case and the digits

-----  
' Aceasta este o problema simpla de criptare a unui sir de caractere  
' Actioneaza numai asupra literelor cu majuscule pe care le inlocuieste  
' in mesajul original cu caracterul desemnat  
' Problema poate fi extinsa pentru a trata literele indiferent de caz si  
cifrele

' Definirea unui vector cu caracterele de criptare corespunzatoare  
literelor alfabetului  
' Defining an array that will contain the encryption corresponding to the  
alphabet capital letters

```
Dim EnCryptedCharacter(1 To 26) As String * 1, currChar As String  
* 1
```

' Sirul de criptare (26 caractere)

```
Dim sirCriptare As String
```

' Iesire din aplicatie

```
Private Sub Exit_Click()
```

```
    Unload Me
```

```
End Sub
```

' Procedura de eveniment asociata butonului Encrypt (cu numele EncryptMessage)

' The Event procedure associated to the command button labeled Encrypt

```
Private Sub EncryptMessage_Click()
```

```
    ' Alphabet      ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
    ' Encryption Characters "STUVWXYZIJKLMNOPQRABCDEFGH"
```

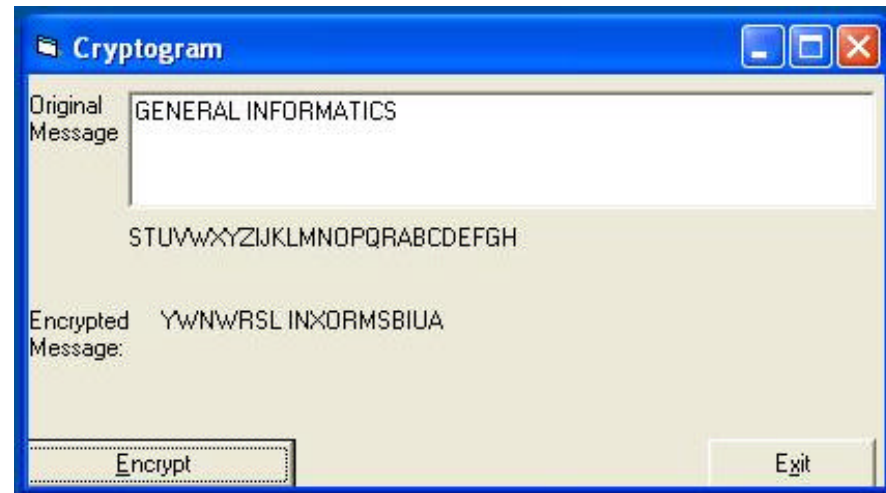
' Aceste caractere sunt utilizate pentru testare 'This characters used for testing

```
Dim encar As String, LgMess As Integer, ToBeEncrypted As String
```

' Apelul subrutinei care pune din sirul de criptare caracterul de substituire dat in sirul de criptare

' The call of the subroutine that forms the encryption characters

```
MakeEncryptionString
```



```

' Pentru testare se afiseaza in eticheta cu numele regulacri din formular sirul de criptare
' Only for test purpose this string displayed in the form in the label called regulacri
regulacri = sirCriptare
' Anulare variabila cu sirul vid 'Erasing the variable with the empty string
encar = ""
' Eliminare extraspatii din mesajul original (stanga si dreapta)
' Eliminating trailing balnks from the original message
ToBeEncrypted = Trim(OriginalMessage)
'Counting the length in characters of the message
LgMess = Len(ToBeEncrypted)
' Ciclu de transformare a caracterelor din mesajul original
' Encryption cycle
For i = 1 To LgMess
    ' Izolarea unui caracter din mesaj ' Insulating a character in the original message
    currChar = UCase(Mid(ToBeEncrypted, i, 1))
    ' If the isolated character is a capital letter yhen encrypt and include the encrypted char in the resulted string
    ' Else include the character as such in the resulting string
    ' Daca caracterul izolat este litera atunci ii pune echivalentul criptat
    If currChar >= "A" And currChar <= "Z" Then
        encar = encar & EnCryptedCharacter(IndexOf(currChar))
    Else
        ' Altfel se include caracterul ca atare in mesaj
        encar = encar & currChar
    End If
Next
' Afisarea mesajului criptat in formular in eticheta cu numele EncryptedMessage
' Display the encrypted string in the form in the label named EncryptedMessage
EncryptedMessage.Caption = encar

End Sub
Sub MakeEncryptionString()
    ' The encryption string/rule given as a string literal value
    ' Sirul/regula de criptare data ca valoare literal alfanumeric
    ' In a real context this string can be read from a file, from the keyboard, from the "wire" in network context,
    ' as values associated to the fingerprint, etc

```



```

sirCiptare = "STUVWXYZIJKLMNOPQRABCDEFGH"
For i = 1 To 26
    EnCryptedCharacter(i) = Mid(sirCiptare, i, 1)
Next i
'EnCryptedCharacter = "STUVWXYZIJKLMNOPQRABCDEFGH"
End Sub
Function IndexOf(aChar As String) As Integer
' Here is an implementation that shows how to use the Case Of sentence
' A more elegant solution can be given by determining the index of the letter from his ASCII CODE
' -----
' Este un exemplu de utilizarea a comenzii Case Of
' O solutie mai eleganta ar fi de a determina indexul din codul ASCII
' al literelor
    Select Case aChar
        Case "A": IndexOf = 1
        Case "B": IndexOf = 2
        Case "C": IndexOf = 3
        Case "D": IndexOf = 4
        Case "E": IndexOf = 5
        Case "F": IndexOf = 6
        Case "G": IndexOf = 7
        Case "H": IndexOf = 8
        Case "I": IndexOf = 9
        Case "J": IndexOf = 10
        Case "K": IndexOf = 11
        Case "L": IndexOf = 12
        Case "M": IndexOf = 13
        Case "N": IndexOf = 14
        Case "O": IndexOf = 15
        Case "P": IndexOf = 16
        Case "Q": IndexOf = 17
        Case "R": IndexOf = 18
        Case "S": IndexOf = 19
        Case "T": IndexOf = 20
        Case "U": IndexOf = 21

```

```
Case "V": IndexOf = 22
Case "W": IndexOf = 23
Case "X": IndexOf = 24
Case "Y": IndexOf = 25
Case "Z": IndexOf = 26
End Select
End Function
```

## EXAMPLE 5

/\*-----\*/

' Definiri de variabile globale  
' - Definirea unei variabile pentru inregistrari denumita Student;  
' tipul Person este definit in Module1  
' - Variabila LastPosition este utilizata pentru a indica ultima inregistrare  
' - Variabila CurrentRecord este utilizata pentru a indica inregistrarea curenta

Dim Student As Person, LastPosition As Long, CurrentRecord As Long

Private Sub AAge\_Change()

' Sterge mesaj de stare la modificarea valorii dupa o eroare

ClearStatusMessage

End Sub

Private Sub AppendRecord\_Click()

' Procedura de eveniment asociata butonului de comanda Append

' Daca datele introduse sunt valide scrie inregistrarea

' altfel listeaza mesajul de eroare

'

If CheckDataForErrors = False Then

MsgBox "Eroare date! ", vbOKOnly, "Diverse ..."

Exit Sub

Else

WriteStudentToFile LastPosition

LastPosition = LastPosition + 1

End If

End Sub

Private Sub ConfirmDeletion\_Click()

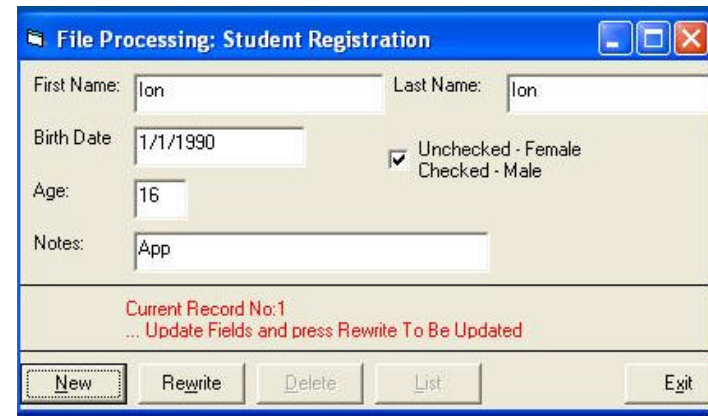
' Stergerea este numai logica.

' Continutul casetelor de text din formular sunt anulate

' si scrise in fisier; caseta de validare Gender este pozitionata la vbGrayed

Student.FirstName = ""

Student.LastName = ""



```

Student.BirthDate = Date
Student.Gender = vbGrayed
' MsgBox Student.Gender & "<>" & vbGrayed
Student.Age = 0
Student.Notes = ""
Put #1, CurrentRecord, Student
DisplayRecordToForm
StatusMessage.Caption = "Record No:" & CurrentRecord & " ... Deleted"
HideConfirmDeletion
EnableButtons
ShowNewRecord
End Sub

```

```

Private Sub DDate_Change()
' Sterge mesaj de stare la modificarea valorii dupa o eroare
ClearStatusMessage
End Sub

```

```

Private Sub DeleteRecord_Click()
Dim WantedRecord As String
DisableButtons
WantedRecord = InputBox("Type the number of the record ypu want Delete" & Chr(13) & "The number must be between 1 and " & _
(LastPosition - 1), "Record Number", 1)
CurrentRecord = Val(Trim(WantedRecord))
If CurrentRecord <> 0 And (CurrentRecord >= 1 And CurrentRecord < LastPosition) Then
Get #1, CurrentRecord, Student
MsgBox Student.Gender
If Student.Gender <> vbGrayed Then
DisplayRecordToForm
StatusMessage.Caption = "Current Record No:" & CurrentRecord & Chr(13) & _
" ... Press Confirm Deletion to Procced"
DisableButtons
ShowConfirmDeletion
Else
MsgBox "The Record allready deleted; You can only update ! ", vbOKOnly, "Diverse ..."

```

```

    End If
Else
    MsgBox "Incorrect Record Specification ! ", vbOKOnly, "Diverse ..."
    EnableButtons
End If
End Sub

```

```

Private Sub FName_Change()
' Sterge mesaj de stare la modificarea valorii dupa o eroare
    ClearStatusMessage
End Sub

```

```

Private Sub ClearStatusMessage()
' Sterge continutul etichetei denumite StatusMessage
    StatusMessage.Caption = ""
End Sub

```

```

Private Sub Iesire_Click()
' Inchidere fisier si iesire
    Close #1
    Unload Me
End Sub

```

```

Private Sub Form_Load()
' La incarcarea formularului se deschide fisierul
' cu inregistrari de lungime fixa
Dim RecLen As Long, FileNum As Integer
RecLen = Len(Student)
LastPosition = 0
' Proprietatea App.Path contine calea catre directorul
' din care se starteaza aplicatia
Open App.Path & "\Studentr.Dat" For Random As #1 Len = RecLen
' Pozitia ultimei inregistrari (neutilizata inca)
LastPosition = LastRecord(1)
' MsgBox "Urmatoarea inregistrare libera este " & LastPosition, vbOKOnly, "Diverse ..."
HideRewrite
HideConfirmDeletion

```

```

    ShowNewRecord
    StatusMessage.Caption = "The file contains: " & LastPosition & " records."
End Sub
Private Sub HideRewrite()
    RewriteRecord.Enabled = False
    RewriteRecord.Visible = False
End Sub
Private Sub ShowRewrite()
    RewriteRecord.Enabled = True
    RewriteRecord.Visible = True
End Sub
Private Sub HideNewRecord()
    NewRecord.Enabled = False
    NewRecord.Visible = False
End Sub
Private Sub ShowNewRecord()
    NewRecord.Enabled = True
    NewRecord.Visible = True
End Sub

Private Sub HideConfirmDeletion()
    ConfirmDeletion.Enabled = False
    ConfirmDeletion.Visible = False
End Sub
Private Sub ShowConfirmDeletion()
    ConfirmDeletion.Enabled = True
    ConfirmDeletion.Visible = True
End Sub

Private Function LastRecord(filenum As Long) As Long
' Rutina Furnizeaza numarul primei inregistrari libere
    LastRecord = 1
' Ciclu citire inregistrari pana la detectare EOF
    Do While Not EOF(1) ' Citeste urmatoarea inregistrare daca nu este EOF

```

```

    Get #1, LastRecord, Student
    LastRecord = LastRecord + 1
Loop ' Reluare ciclu
' Corectie pozitie si atribuire la functie
LastRecord = LastRecord - 1
End Function
Private Function CheckDataForErrors() As Boolean
    CheckDataForErrors = True
    If Trim(FName) = "" Then
        FName.SetFocus
        StatusMessage.Caption = "Error: Fill a value for First Name !"
        CheckDataForErrors = False
    End If
    If Trim(LName) = "" Then
        LName.SetFocus
        StatusMessage.Caption = "Error: Fill a value for Last Name !"
        CheckDataForErrors = False
    End If
    If Trim(DDate) = "" Then
        DDate.SetFocus
        StatusMessage.Caption = "Error: Fill a value for Birth Date !"
        CheckDataForErrors = False
    End If
    If Val(Trim(AAge)) = 0 Or Val(Trim(AAge)) < 12 Then
        AAge.SetFocus
        StatusMessage.Caption = "Error: Fill a valid value for Age !"
        CheckDataForErrors = False
    End If
End Function
Private Sub WriteStudentToFile(RecordNumber As Long)
' Continutul casetelor de text din formular sunt transferate in inregistrare
' si scrise in fisier
    Student.FirstName = FName
    Student.LastName = LName
    Student.BirthDate = DDate

```

```
Student.Gender = FGender.Value
Student.Age = AAge
Student.Notes = NNotes
Put #1, RecordNumber, Student
StatusMessage.Caption = "Record No:" & RecordNumber
```

End Sub

```
Private Sub DisplayRecordToForm()
```

```
' Continutul inregistrarii este afisat in casetele de text din formular
```

```
 FName = Student.FirstName
```

```
 LName = Student.LastName
```

```
 DDate = Student.BirthDate
```

```
 Select Case Student.Gender
```

```
     Case vbUnchecked '=0
```

```
         FGender.Value = vbUnchecked
```

```
     Case vbChecked  '=1
```

```
         FGender.Value = vbChecked
```

```
     Case vbGrayed   '=2
```

```
         FGender.Value = vbGrayed
```

```
 End Select
```

```
 AAge = Student.Age
```

```
 NNotes = Student.Notes
```

End Sub

```
Private Sub ListRecord_Click()
```

```
 Dim WantedRecord As String
```

```
 WantedRecord = InputBox("Type the number of the record ypu want Update" & Chr(13) & "The number must be between 1 and " & _  
 (LastPosition - 1), "Record Number", 1)
```

```
 CurrentRecord = Val(Trim(WantedRecord))
```

```
 If CurrentRecord <> 0 And (CurrentRecord >= 1 And CurrentRecord < LastPosition) Then
```

```
     Get #1, CurrentRecord, Student
```

```
     DisplayRecordToForm
```

```
     StatusMessage.Caption = "Current Record No:" & CurrentRecord
```

```
 Else
```

```
     MsgBox "Incorrect Record Specification ! ", vbOKOnly, "Diverse ..."
```



```
End If
End Sub
```

```
Private Sub LName_Change()
    ClearStatusMessage
End Sub
```

```
Private Sub NewRecord_Click()
' La apasarea butonului New se sterg campurile din formular
' si se modifica mesajul de stare
    ClearFieldsInForm
    StatusMessage.Caption = "Fill data and then press Append to donne!"
' Se invalideaza butonul New
    HideNewRecord
' Cursorul este pozitionat in primul camp
    FName.SetFocus
End Sub
```

```
Private Sub RewriteRecord_Click()
' Daca nu sunt indeplinite conditiile de validare
' se iese din subrutina
    If CheckDataForErrors = False Then
        Exit Sub
    End If
    WriteStudentToFile CurrentRecord
    HideRewrite
    EnableButtons
End Sub
```

```
Private Sub UpdateRecord_Click()
    Dim WantedRecord As String
    DisableButtons
    WantedRecord = InputBox("Type the number of the record ypu want Update" & Chr(13) & "The number must be between 1 and " & _
        (LastPosition - 1), "Record Number", 1)
    CurrentRecord = Val(Trim(WantedRecord))
```

```

If CurrentRecord <> 0 And (CurrentRecord >= 1 And CurrentRecord < LastPosition) Then
    Get #1, CurrentRecord, Student
    DisplayRecordToForm
    StatusMessage.Caption = "Current Record No:" & CurrentRecord & Chr(13) & _
        " ... Update Fields and press Rewrite To Be Updated"
    DisableButtons
    ShowRewrite
Else
    MsgBox "Incorrect Record Specification ! ", vbOKOnly, "Diverse ..."
    EnableButtons
End If
End Sub
Private Sub DisableButtons()
    AppendRecord.Enabled = False
    UpdateRecord.Enabled = False
    DeleteRecord.Enabled = False
    ListRecord.Enabled = False
End Sub
Private Sub EnableButtons()
    AppendRecord.Enabled = True
    UpdateRecord.Enabled = True
    DeleteRecord.Enabled = True
    ListRecord.Enabled = True
End Sub

Private Sub ClearFieldsInForm()
    FName = ""
    LName = ""
    DDate = ""
    FGender.Value = vbUnchecked
    AAge = ""
    NNotes = ""
    StatusMessage.Caption = "Fill data and then press Append to donne!" & Chr(13) & _
        "Or choose another operation !"
End Sub

```

## EXAMPLE 6

' Definiri de variabile globale:  
' - Definirea unei variabile pentru inregistrari denumita Student;  
' tipul Person este definit in Module1  
' - Variabila CurrentRecord este utilizata pentru a indica inregistrarea curenta

```
Dim Student As Person, CurrentRecord As Long, mySortOrder As Integer  
Dim RowInFlex As Long, TableHeading As String
```

```
Private Sub ExitList_Click()  
    Unload Me  
End Sub
```

```
Sub InitialSettings()  
    Dim i As Integer  
    TabelStudenti.Row = 0  
    TabelStudenti.ColAlignment(0) = 7  
    'TabelStudenti.Col = 5  
    'TabelStudenti.MergeCol(5) = True    ' Allow merge on Column 5  
    For i = 0 To TabelStudenti.Cols - 1  
        TabelStudenti.Col = i  
        TabelStudenti.MergeCol(i) = True    ' Allow merge on Columns 0 thru 7  
    Next i  
    TabelStudenti.MergeCells = flexMergeRestrictColumns  
    TabelStudenti.Redraw = True  
End Sub
```

```
Private Sub Form_Load()  
    Dim RecLen As Long, FileNum As Integer  
    RecLen = Len(Student)  
    TableHeading = TabelStudenti.FormatString  
    ' Proprietatea App.Path contine calea catre directorul  
    ' din care se starteaza aplicatia  
    Open App.Path & "\Studentr.Dat" For Random As #1 Len = RecLen
```

| Record No | First Name | Last Name | Birth Date | Gender/Status | Age | Notes |
|-----------|------------|-----------|------------|---------------|-----|-------|
| 1         | Ion        | Ion       | 1/1/1990   | Male          | 16  | App   |
| 2         | A          | Doua      | 1/1/1980   | Female        | 26  | Inreg |
| 3         | Last       | L         | 1/20/1990  |               | 16  | App   |

End Sub

Private Sub orderAsc\_Click()

mySortOrder = 1 'ASC

SortOnColumns

End Sub

Private Sub orderDSC\_Click()

mySortOrder = 2 'DSC

SortOnColumns

End Sub

Private Sub ShowRecords\_Click()

TabelStudenti.FormatString = TableHeading

CurrentRecord = 1

mySortOrder = 1 'Asc

RowInFlex = 0

' Ciclu citire inregistrari pana la detectare EOF

Do While True = True

' Citeste urmatoarea inregistrare

Get #1, CurrentRecord, Student

If Not EOF(1) Then

Select Case ShowAll.Value

Case vbUnchecked

' If deleted records must not be included in list

' skip over deleted

If Student.Gender <> vbGrayed Then

RowInFlex = RowInFlex + 1

FillRecordInRow

End If

Case vbChecked

RowInFlex = RowInFlex + 1

FillRecordInRow

```

        End Select
    Else
        Exit Do
    End If
    CurrentRecord = CurrentRecord + 1
Loop ' Reluare ciclu
InitialSettings
SortOnColumns
End Sub
Sub FillRecordInRow()
    ' Create a row and fill data in the corresponding fields from the just read record
    With TabelStudenti
        .Rows = RowInFlex 'CurrentRecord
        .AddItem (Str(RowInFlex))
        .Row = RowInFlex 'CurrentRecord ' The row in the FlexGrid
        .Col = 1 ' The Column in the current row
        .Text = Student.FirstName
        .Col = 2
        .Text = Student.LastName
        .Col = 3
        .Text = Student.BirthDate
        .Col = 4
        Select Case Student.Gender
            Case vbUnchecked
                .Text = "Female"
            Case vbChecked
                .Text = "Male"
            Case vbGrayed
                .Text = "Deleted"
        End Select
        .Col = 5
        .Text = Student.Age
        .Col = 6
        .Text = Student.Notes
    End With

```

```
End Sub
Sub SortOnColumns()
    With TabelStudenti
        .Col = 0
        .ColSel = TabelStudenti.Cols - 1
        '.Sort = 1 ' Generic Ascending
        .Sort = mySortOrder
    End With
```

```
End Sub
```

```
Private Sub TabelStudenti_DragDrop(Source As VB.Control, X As Single, Y As Single)
    If TabelStudenti.Tag = "" Then Exit Sub
    TabelStudenti.Redraw = False
    TabelStudenti.ColPosition(Val(TabelStudenti.Tag)) = TabelStudenti.MouseCol
    SortOnColumns
    TabelStudenti.Redraw = True
End Sub
```

```
Private Sub TabelStudenti_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    TabelStudenti.Tag = ""
    If TabelStudenti.MouseRow <> 0 Then Exit Sub
    TabelStudenti.Tag = Str(TabelStudenti.MouseCol)
    TabelStudenti.Drag 1
End Sub
```

## EXAMPLE 7

```
/*-----*/
```

```
Attribute VB_Name = "Module1"
```

### EXAMPLE 7.1 Enumerations and Data Structures definition

Rem Defining an array of records: the user type called Student

```
Dim RecordArray(1 To 31) As Student
```

Rem Defining a dynamic array. The array have no dimension. This will be given at usage time

```
Dim DynamicArray() As Variant
```

' Definirea unei enumerari; numele poate fi utilizat in locul valorii asociate

' Defining enumerations; the name can be used instead of associated value

```
Public Enum Zile
```

```
    Luni = 1
```

```
    Marti = 2
```

```
    Miercuri = 3
```

```
    Joi = 4
```

```
    Vineri = 5
```

```
    Sambata = 6
```

```
    Duminica = 7
```

```
End Enum
```

```
Public Enum Days
```

```
    Monday = 1
```

```
    Tuesday = 2
```

```
    Wednesday = 3
```

```
    Thursday = 4
```

```
    Friday = 5
```

```
    Saturday = 6
```

```
    Sunday = 7
```

```
End Enum
```

' Definirea unei variabile utilizator de tip inregistrare

' Defining a type user variable of record type

```
Type Person          'Fixed length record
```

```

    LastName As String * 15    ' The last name limited to 15 bytes (fixed length string)
    FirstName As String * 15   ' The first name limited to 15 bytes
    BirthDate As Date          ' The birth date 10 bytes
    Age As Byte                 ' 1 byte
    Gender As Byte              ' 1 byte
    Notes As String * 20       ' 20 bytes
End Type                      ' Record length 62 bytes

```

```

Type PersonB                  'Variable length record
    LastName As String         'Variable length string
    FirstName As String
    BirthDate As Date
    Age As Byte
    Gender As Boolean
    Notes As String
End Type

```

```

Type GenericAddress 'This type will be used to define and structure an address anywhere this required
    State As String
    City As String
    Zip As Variant
End Type

```

‘ Using a data type definition in another definition  
 ' Utilizarea unei definitii tip in alta definitie

```

Type Student
    StudentID As Long          'Student identifier
    Name As String              'Name (First+last)
    Adress As GenericAddress    'The student address
End Type

```



### EXAMPLE 7.2 Functions and Procedures definition

Function testrecarr()

'Shows how to use the records of type Student

Dim aStudent As Student 'The aStudent record reservation

aStudent.Name = "Michael DOUGLAS" 'Changing the name by assigning a literal value

aStudent.StudentID = 1

aStudent.Address.City = "Bucharest" 'Changing values for address components

aStudent.Address.State = "Romania"

aStudent.Address.Zip = "030829"

'Shows how to use an array of records and how to reference his elements (records) and the attributes of an element

Dim arrayStudent(1 To 31) As Student

Dim i As Integer

For i = 1 To 31

arrayStudent(i).Name = "StudName " & i

arrayStudent(i).StudentID = 5000 + i

arrayStudent(i).Address.City = "Bu " & i

arrayStudent(i).Address.State = "Ro"

arrayStudent(i).Address.Zip = "030829"

Next

End Function

Function SmallestValuePosition(v As Variant, n As Long) As Long

' Returns the position of the smallest element value in the array

Dim i As Long ' Define a index position

Dim Position As Long, minValue As Variant 'Working variables reservation

minValue = v(1) 'The first value in the serie is taken as min

Position = 1 'Initialize the position of the smallest value to 1

For i = n To 1 'Start a cycle in reverse order (from n to 1) on the array

If minValue < v(i) Then 'If the inspected value is less than the one recorded as min

Position = i 'save his position

minValue = v(i) 'and his real value

End If

Next i 'Decrease i by 1 and go to start a new cycle

SmallestValuePosition = Position ' Assign to the function name the value to be returned to the caller

End Function

```

Function SearchingValuePosition(v As Variant, n As Long, wantedValue As Variant) As Long
    ' Returns the position of the item containing a wanted value in the array (the first occurrence)
    ' The function returns a positive value if an occurrence of the wanted value was found in the array
    ' or a -1 value if no occurrence of the searched value found in the array
    Dim i As Long ' Define a index position
    Dim Position As Long, Success As Boolean
    Success = False
    Position = 1
    For i = 1 To n
        If v(i) = wantedValue Then
            Position = i
            Success = True
            Exit For
        End If
    Next i
    If Success = True Then
        SearchingValuePosition = Position
    Else
        SearchingValuePosition = -1
    End If
End Function

```

```

Function ADUNA(X As Variant, Y As Variant) As Variant
    ADUNA = Val(X) + Val(Y)
End Function

```

Sub CelsiusToFahrenheit()

' Acest exemplu utilizeaza functia MsgBox pentru a afisa un tabel 'This sample uses the MsgBox function to display a table

' Liniile sunt numerotate ca in versiunile vechi de Basic 'The lines are numbered as in the older Basic versions

' Variabilele min,max,pas,celsius si tabel sunt de tip Variant 'The variables min,max,pas,celsius and tabel are of Variant datatype

' Variabila fahrenheit# este de tip real 'the Variable fahrenheit# is of real datatype

0 Dim min, max, pas, fahrenheit#, celsius, tabel

1 ' Calculul corespondentei grade C - grade F 'The computation of Celsius-Fahrenheit correspondence

2 min = 0 ' Valoare de start 'Starting value

3 max = 300 ' Valoare finala 'Ending value

4 pas = 20 ' Din cite in cite grade 'Numbers of degrees between two values

5 tabel = "Fahrenheit | Celsius " & Chr(13) & Chr(10) & \_  
String(36, "-") & Chr(13) & Chr(10)

6 For fahrenheit = min To max Step pas

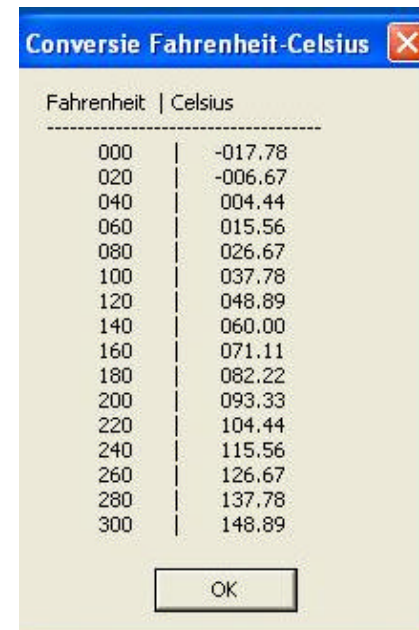
7 celsius = (5 / 9) \* (fahrenheit - 32)

8 tabel = tabel & Right(Space(12) & Format(fahrenheit, "#000"), 12) & " | " & Right(Space(12) & Format(celsius, "#000.00"), 12) & Chr(13)  
& Chr(10)

9 Next fahrenheit

10 MsgBox tabel, , "Conversie Fahrenheit-Celsius"

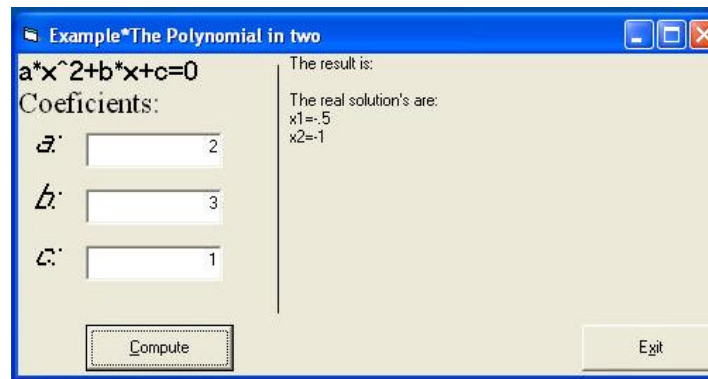
11 End Sub



```

Function EQ2v1(A As Double, B As Double, C As Double) As String
'This function requires at call time the values for the coefficients A, B and C for the polynomial in two
'and returns as a string containing the solution or the warning message
    Dim x1, x2 ' defined with the default datatype Variant
    Dim ipart As Double, D As Double
    If A = 0 Then
        If B = 0 Then
            EQ2v1 = "The equation is extremely degenerate !"
        Else
            x1 = -B / C
            EQ2v1 = "The solution of the equation of degree 1 is x1=" + Str(x1)
        End If
    Else
        D = B ^ 2 - 4 * A * C
        If D >= 0 Then
            x1 = (-B + Sqr(D)) / (2 * A)
            x2 = (-B - Sqr(D)) / (2 * A)
            EQ2v1 = "The real solution's are:" + Chr(13) + "x1=" + Str(x1) + Chr(13) + " x2=" + Str(x2)
        Else
            ' MsgBox "D=" & D
            ipart = (Sqr(Abs(D))) / (2 * A)
            x1 = Str(-B / (2 * A)) & " + i*" & Str(ipart)
            x2 = Str(-B / (2 * A)) & " - i*" & Str(ipart)
            EQ2v1 = "The complex solution's are:" + Chr(13) + "x1=" + x1 + Chr(13) + " x2=" + x2
        End If
    End If
End Function

```



Filename: ProceduresExamples  
Directory: C:\GeneralInformatics-EdEconomica  
Template: C:\Documents and Settings\Vio\Application  
Data\Microsoft\Templates\Normal.dot  
Title: Code Examples  
Subject: Visual Basic 6.0  
Author: Vasile Avram  
Keywords: Array, File, Forms, Processing Structures, Event Procedures, General  
Procedures  
Comments:  
Creation Date: 12/13/2006 7:47 AM  
Change Number: 2  
Last Saved On: 12/13/2006 7:47 AM  
Last Saved By: Vasile Avram  
Total Editing Time: 0 Minutes  
Last Printed On: 12/13/2006 7:47 AM  
As of Last Complete Printing  
Number of Pages: 28  
Number of Words: 4,727 (approx.)  
Number of Characters: 26,947 (approx.)