LISTING PROGRAM

1. Menu Utama

Dim j As Integer
Private Sub help_Click()
    frmHelp.Show
End Sub

Private Sub Hlp_Click()
    frmHelp.Show
End Sub

Private Sub mampat_Click()
    frmMampatQ.Show
End Sub

Private Sub Prof_Click()
    frmProfile.Show
End Sub

Private Sub quit_Click()
    End
End Sub

Private Sub profile_Click()
    frmProfile.Show
End Sub

Private Sub qi_Click()
    End
End Sub

Private Sub Timer1_Timer()
    Dim jud, jud1 As String
    jud = "Studi dan Implementasi Kompresi File Audio"
    jud1 = "Memanfaatkan Metode Adaptive Arithmetic Coding"
    Label1.Caption = Left(jud, j)
    Label2.Caption = Left(jud1, j)
    If j = 1 Then
        Label1.ForeColor = vbCyan
    ElseIf j > 10 Then
        Label1.ForeColor = vbCyan
    Else
        Label1.ForeColor = vbGreen
    End If
    j = j + 1
    If j > Len(jud) + 30 Then
        j = 1
    End If
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    If Button.Index = 1 Then
        frmAAC.Show
    ElseIf Button.Index = 2 Then
        frmAbout.Show
    ElseIf Button.Index = 3 Then
        frmHelp.Show
2. Kompress/Dekompres

Option Explicit
Private WithEvents AAC As clsAAC
Private Sub cmdCancel_Click()
Text1(0).Text = ""
Text1(1).Text = ""
Text1(2).Text = ""
Frame1.Caption = ""
Label2(0).Caption = ""
Label2(1).Caption = ""
Label2(2).Caption = ""
Label2(3).Caption = ""
Label2(4).Caption = ""
cmdCompres.Enabled = False
cmdDecompres.Enabled = False
cmdLoad.Enabled = True
cmdLoad.SetFocus
End Sub

Private Sub cmdLoad_Click()
With cd
.Filter = "Project Files (All Files (*.mp3)|*.mp3)
.DDialogTitle = "Open file wav| MP3"
.ShowOpen
Text1(0).Text = .Filename
Text1(1).Text = App.Path & "\" & "Kom_" & .FileTitle
Text1(2).Text = App.Path & "\" & "Dekom_" & .FileTitle
End With
cmdCompres.Enabled = True
cmdDecompres.Enabled = True
cmdCompres.SetFocus
End With
End Sub

Private Sub cmdCompres_Click()
cmdCompres.Enabled = False
Dim OldTimer As Single
On Error GoTo ErrorHandler
OldTimer = Timer
Call AAC.EncodeFile(Text1(0).Text, Text1(1).Text)
Label2(3).Caption = Timer - OldTimer & " s"
Label2(0).Caption = FileLen(Text1(0).Text) & " bytes"
Label2(1).Caption = FileLen(Text1(1).Text) & " bytes"
Val(Label2(1).Caption) / Val(Label2(0).Caption)) & "%"
Label2(2).Caption = Int((Val(Label2(0).Caption) -
Val(Label2(1).Caption)) & "%"
Val(Label2(0).Caption) * 100) & "%"
Call MsgBox("Kompresi telah sukses.", vbInformation)
Frame1.Caption = "Informasi Hasil Kompresi"
Exit Sub
ErrorHandler:
Call MsgBox("Kompresi error .." & vbCrLf & vbCrLf & Err.Description,_
vbExclamationation)
End Sub
Private Sub cmdDecompres_Click()
    cmdDecompres.Enabled = False
    Dim Filenr As Integer
    Dim OldTimer As Single
    On Error GoTo ErrorHandler
    OldTimer = Timer
    Call AAC.DecodeFile(Text1(1).Text, Text1(2).Text)
    Label2(3).Caption = Timer - OldTimer & " s"
    Label2(0).Caption = FileLen(Text1(1).Text) & " bytes"
    Label2(1).Caption = FileLen(Text1(2).Text) & " bytes"
    Label2(2).Caption = Int(Val(Label2(1).Caption) / Val(Label2(0).Caption) * 100) & "%"
    Frame1.Caption = "Informasi Hasil Dekompresi"
    Call MsgBox("Dekompresi telah sukses", vbInformation) Exit Sub
ErrorHandler:
    Call MsgBox("Dekompresi gagal" & vbCrLf & vbCrLf & Err.Description, vbExclamation)
End Sub

Private Sub cmdQuit_Click()
    Unload Me
End Sub

Private Sub Form_Activate()
    cmdLoad.Enabled = True
    cmdCompres.Enabled = False
    cmdDecompres.Enabled = False
    cmdLoad.SetFocus
End Sub

Private Sub Form_Load()
    Set AAC = New clsAAC
End Sub

Private Sub AAC_Progress(Procent As Integer)
    Label2(4).Caption = Procent & "%"
    DoEvents
End Sub

3. Module-1

Option Explicit
Public location As String
Public MusicName As String
Public moving As Boolean
Public currentduration As Integer
Public autoloading As Boolean
Public currentlyplaying As String
Public PButton As Single
Public CTime As String
Public Activated As Boolean

Public Sub SaveFile()
    Dim FR As Integer
    Dim Line As String
    Dim i As Integer
    Dim Max As Integer
    FR = FreeFile

i = 0
Max = frmWav.Music.ListCount
Do While i < Max
Print #FR, frmWav.PathList.List(i)
i = i + 1
Loop
Close #FR
End Sub

Public Function ReadFile() As String
On Error GoTo BlankFile
Dim FR As Integer
Dim FileContents As String
Dim FileLine
FR = FreeFile
Do While Not EOF(FR)
Line Input #FR, FileLine
FileContents = FileContents & vbNewLine & FileLine
Loop
ReadFile = FileContents
Exit Function
BlankFile:
ReadFile = ""
End Function

Public Sub ConvertMinuets(ByVal Sec As Integer)
If Sec Mod 60 = 0 Then
CTime = Sec \ 60
Else
CTime = Sec \ 60 & " " & Sec Mod 60
End If
End Sub

Public Sub Play()
frmWav.WindowsMediaPlayer1.Controls.Play
Dim mlength As Integer
Dim mstring As String
mlength = Len(MusicName)
mstring = Right(MusicName, 3)
If mstring = ".wm" Then
MusicName = VBA.Strings.Left(MusicName, mlength - 3)
Else
mstring = Right(MusicName, 4)
If mstring = ".mp3" Or mstring = ".asf" Or mstring = ".wma" Or mstring = ".wav" Or mstring = ".snd" Then
MusicName = VBA.Strings.Left(MusicName, mlength - 4)
End If
End If
If (currentlyplaying = ")") = False Then MusicName = currentlyplaying
frmWav.Caption = MusicName
End Sub

Public Sub Pause()
frmWav.WindowsMediaPlayer1.Controls.Pause
End Sub

Public Sub StopSong()
frmWav.WindowsMediaPlayer1.Controls.Stop
End Sub
Public Sub MusicClick()
Dim mlength As Integer
Dim mstring As String
MusicName = frmWav.Music.Text
frmWav.WindowsMediaPlayer1.URL=frmWav.lblNmFile.Caption
frmWav.Caption = ""
mlength = Len(MusicName)
mstring = Right(MusicName, 3)
If mstring = ".wm" Then
    MusicName = VBA.Strings.Left(MusicName, mlength - 3)
Else
    mstring = Right(MusicName, 4)
    If mstring = ".mp3" Or mstring = ".asf" Or mstring = ".wma" Or
    mstring = ".wav" Or mstring = ".snd" Then
        MusicName = VBA.Strings.Left(MusicName, mlength - 4)
    End If
End If
End If
frmWav.Caption = MusicName
frmWav.WindowsMediaPlayer1.Controls.Play
currentduration = frmWav.WindowsMediaPlayer1.currentMedia.duration
frmWav.Timer1.Enabled = False
frmWav.Timer1.Enabled = True
frmWav.leftlabel.Caption = "0"
frmWav.playbutton.Enabled = False
frmWav.pausebutton.Enabled = True
frmWav.stopbutton.Enabled = True
currentlyplaying = MusicName

If InTray = True Then
    Call ModifyIcon(frmWav, currentlyplaying)
End If
End Sub

4. Module-2

Public Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
Public Type SYSTEM_INFO
    dwOemID As Long
    dwPageSize As Long
    lpMinimumApplicationAddress As Long
    lpMaximumApplicationAddress As Long
    dwActiveProcessorMask As Long
    dwNumberOfProcessors As Long
    dwProcessorType As Long
    dwAllocationGranularity As Long
    wProcessorLevel As Integer
    wProcessorRevision As Integer
End Type

Public Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
End Type
dwAvailPhys As Long
dwTotalPageFile As Long
dwAvailPageFile As Long
dwTotalVirtual As Long
dwAvailVirtual As Long
End Type

Public Type LARGE_INTEGER
lowpart As Long
highpart As Long
End Type

Public Type OSVERSIONINFO
dwOSVersionInfoSize As Long
dwMajorVersion As Long
dwMinorVersion As Long
dwBuildNumber As Long
dwPlatformId As Long
szCSDVersion As String * 128
End Type

Public Type FREQ_INFO
in_cycles As Long
ex_ticks As Long
raw_freq As Long
norm_freq As Long
End Type

Public Const DT_TOP = &H0
Public Const DT_LEFT = &H0
Public Const DT_CENTER = &H1
Public Const DT_RIGHT = &H2
Public Const DT_VCENTER = &H4
Public Const DT_BOTTOM = &H8
Public Const DT_WDBREAK = &H10
Public Const DT_SINGLELINE = &H20
Public Const DT_EXPANDTAB = &H40
Public Const DT_TABSTOP = &H80
Public Const DT_NOPRINT = &H100
Public Const DT_EXTERNALLEADING = &H200
Public Const DT_CENTERCROP = &H400
Public Const DT_NOPREFIX = &H800
Public Const DT_INTERNAL = &H1000

Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPUSpeed" () As Long
Public Declare Function CPUSpeed Lib "cpuinf32" Alias "CPU...
Public Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As MEMORYSTATUS)
Public Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA" (lpBuffer As String, nSize As Long) As Long
Public Declare Function GetWindowsDirectory Lib "kernel32" Alias "GetWindowsDirectoryA" (lpBuffer As String, ByVal nSize As Long) As Long
Public Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (lpBuffer As String, ByVal nSize As Long) As Long
Public Declare Function GetVersionEx Lib "kernel32" Alias "GetVersionExA" (lpVersionInformation As OSVERSIONINFO) As Long
Public Declare Sub GetSystemInfo Lib "kernel32" (lpSystemInfo As SYSTEM_INFO)
Public Declare Function GetSystemMetrics Lib "user32" (nindex As Long) As Long
Public Declare Function QueryPerformanceFrequency Lib "kernel32" (lpFrequency As LARGE_INTEGER) As Long
Public Declare Function QueryPerformanceCounter Lib "kernel32" (lpPerformanceCount As LARGE_INTEGER) As Long
Public Declare Function GetTickCount Lib "kernel32" () As Long
Public Declare Function GetKeyboardType Lib "user32" (nTypeFlag As Long) As Long
Public Declare Function IsProcessorFeaturePresent Lib "wnaspi32" (ProcessorFeature As Long) As Boolean
Public Declare Function GetEnvironmentStrings Lib "kernel32" Alias "GetEnvironmentStringsA" () As String
Public Declare Function GetEnvironmentVariable Lib "kernel32" Alias "GetEnvironmentVariableA" (lpName As String, lpBuffer As String, ByVal nSize As Long) As Long
Public Declare Function RegQueryValue Lib "advapi32.dll" Alias "RegQueryValueA" (hKey As Long, lpSubKey As String, lpValue As String, pcbValue As Long) As Long
Public Declare Function RegOpenKey Lib "advapi32.dll" Alias "RegOpenKeyA" (hKey As Long, lpSubKey As String, phkResult As Long) As Long
Public Declare Function RegCloseKey Lib "advapi32.dll" (hKey As Long) As Long
Public Declare Function RegEnumKey Lib "advapi32.dll" Alias "RegEnumKeyA" (hKey As Long, dwIndex As Long, lpName As String, cbName As Long) As Long
Public Declare Function GetPrivateProfileSection Lib "kernel32" Alias "GetPrivateProfileSectionA" (lpAppName As String, lpReturnedString As String, nSize As Long, lpFileName As String) As Long
Public Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" (lpApplicationName As String, lpKeyName As String, lpDefault As String, lpReturnedString As String, nSize As Long, lpFileName As String) As Long
Public Declare Function GetPrivateProfileInt Lib "kernel32" Alias "GetPrivateProfileIntA" (lpApplicationName As String, lpKeyName As String, nDefault As Long, lpFileName As String) As Long
Public Declare Function WritePrivateProfileSection Lib "kernel32" Alias "WritePrivateProfileSectionA" (lpAppName As String, lpString As String, ByVal lpFileName As String) As Long
Public Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA" (lpApplicationName As String, lpKeyName As String, lpReturnedString As String, lpFileName As String) As Long
ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long

Global myVer As OSVERSIONINFO

Public Function run(sProgram As String, bPause As Boolean) As Long
    Dim retval
    Dim temp As String
    retval = Shell(sProgram, vbNormalFocus)
    If bPause Then
        frmSpawn.lblSpawn = sProgram
        frmSpawn.Show vbModal
    End If
    run = retval
End Function

Public Function getWindowsDir()
    Dim temp$ As String
    Dim Value As Long
    Value = GetWindowsDirectory(temp, 0)
    If Value <> 0 Then
        getWindowsDir = temp
    Else
        getWindowsDir = "error"
    End If
End Function

Public Function LPSTRToVBString$(ByVal s$)
    Dim nullpos& As Long
    nullpos& = InStr(s$, Chr$(0))
    If nullpos > 0 Then
        LPSTRToVBString = Left$(s$, nullpos - 1)
    Else
        LPSTRToVBString = ""
    End If
End Function

5. Module 3

Public Declare Function Shell_NotifyIconA Lib "shell32.dll" (ByVal dwMessage As Long, lpData As NOTIFYICONDATA) As Integer
Const NIM_ADD = &H0
Const NIM_MODIFY = &H1
Const NIM_DELETE = &H2
Const NIF_MESSAGE = &H1
Const NIF_ICON = &H2
Const NIF_TIP = &H4
Public Const WM_LBUTTONDOWN = &H201
Public Const WM_RBUTTONDOWN = &H204
Public Const WM_LBUTTONDOWNCLK = &H203
Public Type NOTIFYICONDATA
    cbSize As Long
    hwnd As Long
    uID As Long
    uFlags As Long
Public InTray As Boolean
Public Sub AddIcon(TargetForm As Form, ToolTip As String)
    Dim AddIconData As NOTIFYICONDATA
    With AddIconData
        .cbSize = Len(AddIconData)
        .hIcon = TargetForm.Icon
        .hwnd = TargetForm.hwnd
        .szTip = ToolTip & vbNullChar
        .uCallbackMessage = WM_LBUTTONDOWN
        .uFlags = NIF_MESSAGE Or NIF_ICON Or NIF_TIP
        .uID = vbNull
    End With
    Call Shell_NotifyIconA(NIM_ADD, AddIconData)
    InTray = True
End Sub

Public Sub ModifyIcon(TargetForm As Form, ToolTip As String)
    Dim AddIconData As NOTIFYICONDATA
    With AddIconData
        .cbSize = Len(AddIconData)
        .hIcon = TargetForm.Icon
        .hwnd = TargetForm.hwnd
        .szTip = ToolTip & vbNullChar
        .uCallbackMessage = WM_LBUTTONDOWN
        .uFlags = NIF_MESSAGE Or NIF_ICON Or NIF_TIP
        .uID = vbNull
    End With
    Call Shell_NotifyIconA(NIM_MODIFY, AddIconData)
    InTray = True
End Sub

Public Sub DeleteIcon(TargetForm As Form)
    Dim AddIconData As NOTIFYICONDATA
    With AddIconData
        .cbSize = Len(AddIconData)
        .hIcon = vbNull
        .hwnd = TargetForm.hwnd
        .szTip = vbNullChar
        .uCallbackMessage = vbNull
        .uFlags = NIF_MESSAGE Or NIF_ICON Or NIF_TIP
        .uID = vbNull
    End With
    Call Shell_NotifyIconA(NIM_DELETE, AddIconData)
    InTray = False
End Sub

6. ClasModule ClsACC

Option Explicit
Private Const PROGRESS_CALCFREQUENCY = 7
Private Const PROGRESS_CALCCRC = 5
Private Const PROGRESS_ENCODING = 88
Private Const PROGRESS_DECODING = 89
Private Const PROGRESS_CHECKCRC = 11
Event Progress(Procent As Integer)
Private Type HUFFMANTREE
    ParentNode As Integer
    RightNode As Integer
    LeftNode As Integer
    Value As Integer
    Weight As Long
End Type

Private Type ByteArray
    Count As Integer
    Data() As Byte
End Type

Private Declare Sub CopyMem Lib "kernel32" Alias "RtlMoveMemory"
(Destination As Any, Source As Any, ByVal Length As Long)

Public Sub EncodeFile(SourceFile As String, DestFile As String)
    Dim ByteArray() As Byte
    Dim Filenr As Integer

    'Make sure the source file exists
    If (Not FileExist(SourceFile)) Then
        does not exist"
    End If
    Filenr = FreeFile
    Open SourceFile For Binary As #Filenr
    ReDim ByteArray(0 To LOF(Filenr) - 1)
    Get #Filenr, , ByteArray()
    Close #Filenr
    Call EncodeByte(ByteArray(), UBound.ByteArray) + 1)
    If (FileExist(DestFile)) Then Kill DestFile
    Open DestFile For Binary As #Filenr
    Put #Filenr, , ByteArray()
    Close #Filenr
End Sub

Public Sub DecodeFile(SourceFile As String, DestFile As String)
    Dim ByteArray() As Byte
    Dim Filenr As Integer

    If (Not FileExist(SourceFile)) Then
        Err.Raise vbObjectError, "clsaac.DecodeFile()", "Source file
        does not exist"
    End If
    Filenr = FreeFile
    Open SourceFile For Binary As #Filenr
    ReDim ByteArray(0 To LOF(Filenr) - 1)
    Get #Filenr, , ByteArray()
    Close #Filenr
    Call DecodeByte(ByteArray(), UBound.ByteArray) + 1)
    If (FileExist(DestFile)) Then Kill DestFile
    Open DestFile For Binary As #Filenr
    Put #Filenr, , ByteArray()
    Close #Filenr
End Sub
Private Sub CreateTree(Nodes() As HUFFMANTREE, NodesCount As Long, Char As Long, Bytes As ByteArray)
    Dim a As Integer
    Dim NodeIndex As Long
    NodeIndex = 0
    For a = 0 To (Bytes.Count - 1)
        If (Bytes.Data(a) = 0) Then
            If (Nodes(NodeIndex).LeftNode = -1) Then
                Nodes(NodeIndex).LeftNode = NodesCount
                Nodes(NodesCount).ParentNode = NodeIndex
                Nodes(NodesCount).LeftNode = -1
                Nodes(NodesCount).RightNode = -1
                Nodes(NodesCount).Value = -1
                NodesCount = NodesCount + 1
            End If
            NodeIndex = Nodes(NodeIndex).LeftNode
        ElseIf (Bytes.Data(a) = 1) Then
            If (Nodes(NodeIndex).RightNode = -1) Then
                Nodes(NodeIndex).RightNode = NodesCount
                Nodes(NodesCount).ParentNode = NodeIndex
                Nodes(NodesCount).LeftNode = -1
                Nodes(NodesCount).RightNode = -1
                Nodes(NodesCount).Value = -1
                NodesCount = NodesCount + 1
            End If
            NodeIndex = Nodes(NodeIndex).RightNode
        Else
            Stop
        End If
    Next
    Nodes(NodeIndex).Value = Char
End Sub

Public Sub EncodeByte(ByteArray() As Byte, ByteLen As Long)
    Dim i As Long
    Dim j As Long
    Dim Char As Byte
    Dim BitPos As Byte
    Dim lNode1 As Long
    Dim lNode2 As Long
    Dim lNodes As Long
    Dim lLength As Long
    Dim Count As Integer
    Dim lWeight1 As Long
    Dim lWeight2 As Long
    Dim Result() As Byte
    Dim ByteValue As Byte
    Dim ResultLen As Long
    Dim Bytes As ByteArray
    Dim NodesCount As Integer
    Dim NewProgress As Integer
    Dim CurrProgress As Integer
    Dim BitValue(0 To 7) As Byte
    Dim CharCount(0 To 255) As Long
    Dim Nodes(0 To 511) As HUFFMANTREE
    Dim CharValue(0 To 255) As ByteArray
If (ByteLen = 0) Then
    ReDim Preserve ByteArray(0 To ByteLen + 3)
If (ByteLen > 0) Then
    Call CopyMem(ByteArray(4), ByteArray(0), ByteLen)
End If
ByteArray(0) = 72 "H"
ByteArray(1) = 69 "E"
ByteArray(2) = 48 "0"
ByteArray(3) = 13 'vbCr
Exit Sub
End If

ReDim Result(0 To 522)
Result(0) = 72
Result(1) = 69
Result(2) = 51
Result(3) = 13
ResultLen = 4
For i = 0 To (ByteLen - 1)
    CharCount(ByteArray(i)) = CharCount(ByteArray(i)) + 1
    If (i Mod 1000 = 0) Then
        NewProgress = i / ByteLen * PROGRESS_CALC_FREQUENCY
        If (NewProgress <> CurrProgress) Then
            CurrProgress = NewProgress
            RaiseEvent Progress(CurrProgress)
        End If
    End If
Next

'Create a leaf for each character
For i = 0 To 255
    If (CharCount(i) > 0) Then
        With Nodes(NodesCount)
            .Weight = CharCount(i)
            .Value = i
            .LeftNode = -1
            .RightNode = -1
            .ParentNode = -1
        End With
        NodesCount = NodesCount + 1
    End If
Next

'Create the Huffman Tree
For lNodes = NodesCount To 2 Step -1
    'Get the two leafs with the smallest weights
    lNode1 = -1: lNode2 = -1
    For i = 0 To (NodesCount - 1)
        If (Nodes(i).ParentNode = -1) Then
            If (lNode1 = -1) Then
                lWeight1 = Nodes(i).Weight
                lNode1 = i
            ElseIf (lNode2 = -1) Then
                lWeight2 = Nodes(i).Weight
                lNode2 = i
            ElseIf (Nodes(i).Weight < lWeight1) Then
                If (Nodes(i).Weight < lWeight2) Then
                    If (lWeight1 < lWeight2) Then
                        lWeight2 = Nodes(i).Weight
                        lNode2 = i
                    Else
                        lWeight1 = Nodes(i).Weight
                        lNode1 = i
                    End If
                Else
                    lWeight2 = Nodes(i).Weight
                    lNode2 = i
                End If
            End If
        End If
    Next
Else
    lWeight1 = Nodes(i).Weight
    lNode1 = i
End If
Else
    lWeight1 = Nodes(i).Weight
    lNode1 = i
End If
ElseIf (Nodes(i).Weight < lWeight2) Then
    lWeight2 = Nodes(i).Weight
    lNode2 = i
End If
End If
Next

With Nodes(NodesCount)
    .Weight = lWeight1 + lWeight2
    .LeftNode = lNode1
    .RightNode = lNode2
    .ParentNode = -1
    .Value = -1
End With

Nodes(lNode1).ParentNode = NodesCount
Nodes(lNode2).ParentNode = NodesCount
NodesCount = NodesCount + 1

Next

ReDim Bytes.Data(0 To 255)
Call CreateBitSequences(Nodes(), NodesCount - 1, Bytes, CharValue)
For i = 0 To 255
    If (CharCount(i) > 0) Then
        lLength = lLength + CharValue(i).Count * CharCount(i)
    End If
Next

lLength = IIf(lLength Mod 8 = 0, lLength \ 8, lLength \ 8 + 1)
If ((lLength = 0) Or (lLength > ByteLen)) Then
    ReDim Preserve ByteArray(0 To ByteLen + 3)
    Call CopyMem(ByteArray(4), ByteArray(0), ByteLen)
    ByteArray(0) = 72
    ByteArray(1) = 69
    ByteArray(2) = 48
    ByteArray(3) = 13
    Exit Sub
End If

Char = 0
For i = 0 To (ByteLen - 1)
    Char = Char Xor ByteArray(i)
    If (i Mod 10000 = 0) Then
        NewProgress = i/ByteLen*PROGRESS_CALCCRC+ PROGRESS_CALCFREQUENCY
        If (NewProgress <> CurrProgress) Then
            CurrProgress = NewProgress
            RaiseEvent Progress(CurrProgress)
        End If
    End If
End If
Next
Result(ResultLen) = Char
ResultLen = ResultLen + 1
Call CopyMem(Result(ResultLen), ByteLen, 4)
ResultLen = ResultLen + 4
For i = 0 To 7
    BitValue(i) = 2 ^ i
Next

Count = 0
For i = 0 To 255
    If (CharValue(i).Count > 0) Then
        Count = Count + 1
    End If
Next
Call CopyMem(Result(ResultLen), Count, 2)
ResultLen = ResultLen + 2
Count = 0
For i = 0 To 255
    If (CharValue(i).Count > 0) Then
        Result(ResultLen) = i
        ResultLen = ResultLen + 1
        Result(ResultLen) = CharValue(i).Count
        ResultLen = ResultLen + 1
        Count = Count + 16 + CharValue(i).Count
    End If
Next
ReDim Preserve Result(0 To ResultLen + Count \ 8)
BitPos = 0
ByteValue = 0
For i = 0 To 255
    With CharValue(i)
        If (.Count > 0) Then
            For j = 0 To (.Count - 1)
                If (.Data(j)) Then ByteValue = ByteValue + BitValue(BitPos)
                BitPos = BitPos + 1
                If (BitPos = 8) Then
                    Result(ResultLen) = ByteValue
                    ResultLen = ResultLen + 1
                    ByteValue = 0
                    BitPos = 0
                End If
            Next
        End If
    End With
Next
If (BitPos > 0) Then
    Result(ResultLen) = ByteValue
    ResultLen = ResultLen + 1
End If
ReDim Preserve Result(0 To ResultLen - 1 + lLength)
Char = 0
BitPos = 0
For i = 0 To (ByteLen - 1)
    With CharValue(ByteArray(i))
        For j = 0 To (.Count - 1)
            If (.Data(j) = 1) Then Char = Char + BitValue(BitPos)
            BitPos = BitPos + 1
            If (BitPos = 8) Then
                Result(ResultLen) = Char
                ResultLen = ResultLen + 1
                BitPos = 0
                Char = 0
            End If
        Next
    End With
Next
End If
Next
End With
If (i Mod 10000 = 0) Then
NewProgress = i / ByteLen * PROGRESS_ENCODING + PROGRESS_CALCCRC_ + PROGRESS_CALCFREQUENCY
If (NewProgress <> CurrProgress) Then
CurrProgress = NewProgress
RaiseEvent Progress(CurrProgress)
End If
End If
Next
If (BitPos > 0) Then
Result(ResultLen) = Char
ResultLen = ResultLen + 1
End If
ReDim ByteArray(0 To ResultLen - 1)
Call CopyMem(ByteArray(0), Result(0), ResultLen)
If (CurrProgress <> 100) Then
RaiseEvent Progress(100)
End If
End Sub
Public Function DecodeString(Text As String) As String
Dim ByteArray() As Byte
ByteArray() = StrConv(Text, vbFromUnicode)
Call DecodeByte(ByteArray, Len(Text))
DecodeString = StrConv(ByteArray(), vbUnicode)
End Function
Public Function EncodeString(Text As String) As String
Dim ByteArray() As Byte
ByteArray() = StrConv(Text, vbFromUnicode)
Call EncodeByte(ByteArray, Len(Text))
EncodeString = StrConv(ByteArray(), vbUnicode)
End Function
Public Sub DecodeByte(ByteArray() As Byte, ByteLen As Long)
Dim i As Long
Dim j As Long
Dim Pos As Long
Dim Char As Byte
Dim CurrPos As Long
Dim Count As Integer
Dim CheckSum As Byte
Dim Result() As Byte
Dim BitPos As Integer
Dim NodeIndex As Long
Dim ByteValue As Byte
Dim ResultLen As Long
Dim NodesCount As Long
Dim lResultLen As Long
Dim NewProgress As Integer
Dim CurrProgress As Integer
Dim BitValue(0 To 7) As Byte
Dim Nodes(0 To 511) As HUFFMANTREE
Dim CharValue(0 To 255) As ByteArray
If(ByteArray(0) <> 72) Or (ByteArray(1) <> 69) Or (ByteArray(3) <> _
13) Then
  ElseIf (ByteArray(2) = 48) Then
    Call CopyMem(ByteArray(0), ByteArray(4), ByteLen - 4)
    ReDim Preserve ByteArray(0 To ByteLen - 5)
    Exit Sub
  ElseIf (ByteArray(2) <> 51) Then
    Exit Sub
  End If
End If
CurrPos = 5
CheckSum = ByteArray(CurrPos - 1)
CurrPos = CurrPos + 1
Call CopyMem(ResultLen, ByteArray(CurrPos - 1), 4)
CurrPos = CurrPos + 4
lResultLen = ResultLen
If (ResultLen = 0) Then Exit Sub
ReDim Result(0 To ResultLen - 1)
Call CopyMem(Count, ByteArray(CurrPos - 1), 2)
CurrPos = CurrPos + 2
For i = 1 To Count
  With CharValue(ByteArray(CurrPos - 1))
    CurrPos = CurrPos + 1
    .Count = ByteArray(CurrPos - 1)
    CurrPos = CurrPos + 1
    ReDim .Data(0 To .Count - 1)
  End With
Next
For i = 0 To 7
  BitValue(i) = 2 ^ i
Next
ByteValue = ByteArray(CurrPos - 1)
CurrPos = CurrPos + 1
BitPos = 0
For i = 0 To 255
  With CharValue(i)
    If (.Count > 0) Then
      For j = 0 To (.Count - 1)
        If (ByteValue And BitValue(BitPos)) Then .Data(j) = 1
        BitPos = BitPos + 1
        If (BitPos = 8) Then
          ByteValue = ByteArray(CurrPos - 1)
          CurrPos = CurrPos + 1
          BitPos = 0
          End If
        Next
      End If
    End With
  Next
  End If
  End With
Next
If (BitPos = 0) Then CurrPos = CurrPos - 1
NodesCount = 1
Nodes(0).LeftNode = -1
Nodes(0).RightNode = -1
Nodes(0).ParentNode = -1
Nodes(0).Value = -1
For i = 0 To 255
  Call CreateTree(Nodes(), NodesCount, i, CharValue(i))
Next
ResultLen = 0
For CurrPos = CurrPos To ByteLen
  ByteValue = ByteArray(CurrPos - 1)
For BitPos = 0 To 7
    If (ByteValue And BitValue(BitPos)) Then
        NodeIndex = Nodes(NodeIndex).RightNode
    Else
        NodeIndex = Nodes(NodeIndex).LeftNode
    End If
    If (Nodes(NodeIndex).Value > -1) Then
        Result(ResultLen) = Nodes(NodeIndex).Value
        ResultLen = ResultLen + 1
        If (ResultLen = lResultLen) Then GoTo DecodeFinished
    End If
    Next
If (CurrPos Mod 10000 = 0) Then
    NewProgress = CurrPos / ByteLen * PROGRESS_DECODING
    If (NewProgress <> CurrProgress) Then
        CurrProgress = NewProgress
        RaiseEvent Progress(CurrProgress)
    End If
End If
Next
DecodeFinished:
Char = 0
For i = 0 To (ResultLen - 1)
    Char = Char Xor Result(i)
    If (i Mod 10000 = 0) Then
        NewProgress = i / ResultLen * PROGRESS_CHECKCRC + PROGRESS_DECODING
        If (NewProgress <> CurrProgress) Then
            CurrProgress = NewProgress
            RaiseEvent Progress(CurrProgress)
        End If
    End If
Next
If (Char <> CheckSum) Then
    Err.Raise vbObjectError, "clsHuffman.Decode()", "The data might be corrupted (checksum did not match expected value)"
End If
ReDim ByteArray(0 To ResultLen - 1)
Call CopyMem(ByteArray(0), Result(0), ResultLen)
If (CurrProgress <> 100) Then
    RaiseEvent Progress(100)
End If
End Sub
Private Sub CreateBitSequences(Nodes() As HUFFMANTREE, ByVal NodeIndex As Integer, ByVal Bytes As ByteArray, ByVal CharValue() As ByteArray)
    Dim NewBytes As ByteArray
    If (Nodes(NodeIndex).Value > -1) Then
        CharValue(Nodes(NodeIndex).Value) = Bytes
        Exit Sub
    End If
    If (Nodes(NodeIndex).LeftNode > -1) Then
        NewBytes = Bytes
        NewBytes.Data(NewBytes.Count) = 0
        NewBytes.Count = NewBytes.Count + 1
        Call CreateBitSequences(Nodes(), Nodes(NodeIndex).LeftNode, NewBytes, CharValue)
    End If
    If (Nodes(NodeIndex).RightNode > -1) Then
        NewBytes = Bytes
        Call CreateBitSequences(Nodes(), Nodes(NodeIndex).RightNode, NewBytes, CharValue)
    End If
End Sub
Private Sub CreateBitSequences(Nodes As Nodes, NodeIndex As Integer, NewBytes As Byte) As Boolean
    Dim CharValue As Integer
    Dim NewBytesCount As Integer

    NewBytes.Data(NewBytes.Count) = 1
    NewBytes.Count = NewBytes.Count + 1
    Call CreateBitSequences(Nodes(), Nodes(NodeIndex).RightNode, NewBytes, CharValue)
End Sub

Private Function FileExist(Filename As String) As Boolean
    On Error GoTo FileDoesNotExist
    Call FileLen(Filename)
    FileExist = True
    Exit Function

FileDoesNotExist:
    FileExist = False
End Function