Below find a simple framework for a database program.
You can use it in any way in your own projects, there is no usage limitation.

There are two versions available,

- Using SUB's
  - Sub's are an easy way to create reusable code.
  - The variable names inside of the sub's are different from the ones used outside.
  - You can use a sub in different projects.

- Using GOSUB's
  - This is for people who started to program in BASIC long ago.
  - Variable names must be the same for the whole program, which makes following their use difficult in bigger projects.

-
StPendl Oct 29, 2010

Using SUB's

```
[init]
    'define global variables
    global MaxItems

    'predefine item array
    dim items$(1), search$(1)

    'get database contents
    call OpenDB
    call ReadDB
    call CloseDB

[MainGUI]
    'Form created with the help of Freeform 3 v01-28-07
    'Generated on Jun 19, 2007 at 22:50:13

    nomainwin
    WindowWidth = 440
    WindowHeight = 230
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)
```

```
    listbox #main.itemlist, items$(, [DisplayItem],    5,   5, 175,
185
    statictext #main.NumberTxt,  "Item Number:", 200,   7,  80,  25
    statictext #main.NumberDisp, "",             300,   7,  95,  25
    statictext #main.NameTxt,    "Item Name:",   200,  32,  80,  25
    statictext #main.NameDisp,   "",             300,  32,  95,  25
    statictext #main.PrizeTxt,   "Item Prize:",  200,  57,  80,  25
    statictext #main.PrizeDisp,  "",             300,  57,  95,  25
    button #main.add,   "Add Item",   CheckButton, UL, 200, 112,  63,
  25
    button #main.edit,  "Edit Item",  CheckButton, UL, 275, 112,  63,
  25
    button #main.delete,"Delete Item",CheckButton, UL, 350, 112,  75,
  25
    button #main.search,"Search",     [search],    UL, 200, 162,  63,
  25
    button #main.exit,  "EXIT",       [quit.main], UL, 350, 162,  39,
  25

    open "Simple Database Framework" for window as #main
    print #main, "font ms_sans_serif 10"
    print #main, "trapclose [quit.main]"
    #main.itemlist "singleclickselect"
    wait

[DisplayItem]
    'get index of selected item
    #main.itemlist "selectionindex? index"

    #main.NameDisp word$(items$(index), 1, chr$(0))
    #main.NumberDisp word$(items$(index), 2, chr$(0))
    #main.PrizeDisp word$(items$(index), 3, chr$(0))
    wait

[search]
    'search in the database
    WindowWidth = 430
    WindowHeight = 190

    'position of dialogs are relative to previous open window
    UpperLeftX=1
    UpperLeftY=1

    textbox    #search.String,   5,   5, 175,  25
    button     #search.default,    "Search", [doSearch],  UL, 200,
```

```
   5,  75,  25
    listbox    #search.itemlist,   search$(,[doDisplay],   5,  35,
175, 120
    statictext #search.NumberTxt,  "Item Number:", 200,  35,  80,  25
    statictext #search.NumberDisp, "",             300,  35,  95,  25
    statictext #search.NameTxt,    "Item Name:",   200,  60,  80,  25
    statictext #search.NameDisp,   "",             300,  60,  95,  25
    statictext #search.PrizeTxt,   "Item Prize:",  200,  85,  80,  25
    statictext #search.PrizeDisp,  "",             300,  85,  95,  25
    button     #search.cancel,     "Close",[quit.search],  UL, 300,
127,  63,  25

    'modal windows block access to the previous window
    open "Search Database for Name" for dialog_modal as #search
    print #search, "font ms_sans_serif 10"
    print #search, "trapclose [quit.search]"
    #search.itemlist "singleclickselect"
    wait

[doSearch]
    redim search$(MaxItems)
    foundItem = 0

    ' search by name = field 1
    FieldNumber = 1

    #search.String "!contents? SearchString$"

    for Count = 1 to MaxItems
        'ignore case using LOWER$()
        if instr(lower$(word$(items$(Count), FieldNumber, chr$(0))),
lower$(SearchString$)) > 0 then
            foundItem = foundItem + 1
            search$(foundItem) = items$(Count)
        end if
    next

    #search.itemlist "reload"
    #search.itemlist "selectindex 0"
    wait

[doDisplay]
    'get index of selected item
    #search.itemlist "selectionindex? index"

    #search.NameDisp word$(search$(index), 1, chr$(0))
```

```
    #search.NumberDisp word$(search$(index), 2, chr$(0))
    #search.PrizeDisp word$(search$(index), 3, chr$(0))
    wait

[quit.search]
    close #search
    wait

[quit.main]
    close #main
    END

sub CheckButton handle$
    'get extension of button
    extension$ = word$(handle$, 2, ".")

    'get index of selected item
    #main.itemlist "selectionindex? index"

    'select action based on pushed button
    select case extension$
        case "add"
        call DisplayDialog "Add Item", MaxItems

        case "edit"
        if index > 0 then call DisplayDialog "Edit Item", index

        case "delete"
        if index > 0 then call DeleteItem index
    end select

    'refresh listbox contents
    #main.itemlist "reload"

    'cancel selection to allow reselection of currently selected item
    #main.itemlist "selectindex 0"
end sub

sub DisplayDialog Caption$, ItemNumber
    'Form created with the help of Freeform 3 v01-28-07
    'Generated on Jun 19, 2007 at 22:59:56

    WindowWidth = 275
    WindowHeight = 195

    'position of dialogs are relative to previous open window
```

```
    UpperLeftX=1
    UpperLeftY=1

    statictext #item.NumberTxt, "Item Number:",  10,   7,  80,  25
    statictext #item.NameTxt,   "Item Name:",    10,  42,  80,  25
    statictext #item.PrizeTxt,  "Item Prize:",   10,  77,  80,  25
    textbox #item.Number, 105,   7, 150,  25
    textbox #item.Name,   105,  42, 150,  25
    textbox #item.Prize,  105,  77, 150,  25
    button #item.cancel,  "Close",[quit.item], UL,  95, 127,  63,  25
    button #item.default, "Apply",[apply],     UL, 180, 127,  75,  25

    'modal windows block access to the previous window
    open Caption$; " - "; ItemNumber for dialog_modal as #item
    print #item, "font ms_sans_serif 10"
    print #item, "trapclose [quit.item]"

    if ItemNumber <> MaxItems then
        #item.Name word$(items$(ItemNumber), 1, chr$(0))
        #item.Number word$(items$(ItemNumber), 2, chr$(0))
        #item.Prize word$(items$(ItemNumber), 3, chr$(0))
    end if
    #item.Number "!setfocus"
    wait


[apply]
    ' apply changes
    #item.Number "!contents? Temp1$"
    #item.Name "!contents? Name$"
    #item.Prize "!contents? Temp2$"


' Make sure info in boxes is the proper type of data (number/string)
    if Temp1$ = str$(val(Temp1$)) then
        Number = val(Temp1$)
    else
        ' Item entered in the Number box is not a number !
        notice "Item Number must be numeric only."
        wait
    end if
    if Temp2$ = str$(val(Temp2$)) then
        Prize = val(Temp2$)
    else
        ' Item entered in the Prize box is not a number !
        notice "Item Prize must be numeric only."
```

```
        wait
    end if

    'fill the array element with the data

'separate fields by CHR$(0) to display only the first field in the lis
tbox
    items$(ItemNumber) = trim$(Name$); chr$(0); Number; chr$(0); Prize

    call ApplyItemData
    wait

[quit.item]
    'exit dialog
    close #item
end sub

sub ApplyItemData
    call BackupDB
    call OpenDB
    call WriteDB
    call ReadDB
    call CloseDB
end sub

sub DeleteItem ItemIndex
    confirm "Delete Item ... "+str$(ItemIndex)+chr$(13)+_
        "Name ..... "+word$(items$(ItemIndex), 1, chr$(0))+chr$(13)+_
        "Number ... "+word$(items$(ItemIndex), 2, chr$(0))+chr$(13)+_
        "Prize .... "+word$(items$(ItemIndex), 3, chr$(0)); answer

    if answer then
        items$(ItemIndex) = ""

        call BackupDB
        call OpenDB
        call WriteDB
        call ReadDB
        call CloseDB
    end if
end sub

sub OpenDB
    'open database and define record length
    open "database.dat" for random as #db len=150
```

```
    'set the fields, include some extra space for future use
    field #db,_
        40 as ItemName$,_
        10 as ItemNumber,_
        10 as ItemPrize,_
        90 as Reserve$
end sub

sub CloseDB
    close #db
end sub

sub ReadDB
    'get the number of records in the database
    '= length of database file divided by the record length
    TotalRecords = lof(#db)/150

    'check if the database is corrupted
    if TotalRecords <> int(TotalRecords) then
        notice "Database corrupted"; chr$(13);
"Please check its contents!"
        TotalRecords = int(TotalRecords + .5)
    end if

    'dimension array to enable adding one record
    MaxItems = TotalRecords + 1
    redim items$(MaxItems)

    for Record = 1 to TotalRecords
        get #db, Record

        'fill the array with the data

'separate fields by CHR$(0) to display only the first field in the lis
tbox
        items$(Record) = trim$(ItemName$); chr$(0)
; ItemNumber; chr$(0); ItemPrize
    next
end sub

sub WriteDB
    Record = 1

    for Count = 1 to MaxItems
        if items$(Count) <> "" then
            ItemName$ = word$(items$(Count), 1, chr$(0))
```

```
            ItemNumber = val(word$(items$(Count), 2, chr$(0)))
            ItemPrize = val(word$(items$(Count), 3, chr$(0)))

            put #db, Record
            Record = Record + 1
        end if
    next
end sub

sub BackupDB
    if FileExists("database.bak") then kill "database.bak"

    name "database.dat" as "database.bak"
end sub

function FileExists(FilePath$)
    ' returns zero if file does not exist
    ' returns one if file exists
    dim FileExistsInfo$(1,1)

    files "", FilePath$, FileExistsInfo$(

    FileExists = val(FileExistsInfo$(0,0))
end function
```

Back to Top

Using GOSUB's

```
[init]
    'predefine item array
    dim items$(1)

    'get database contents
    gosub [OpenDB]
    gosub [ReadDB]
    gosub [CloseDB]

[MainGUI]
    'Form created with the help of Freeform 3 v01-28-07
    'Generated on Jun 19, 2007 at 22:50:13

    nomainwin
```

```
    WindowWidth = 440
    WindowHeight = 230
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)

    listbox #main.itemlist, items$(, [DisplayItem],    5,   5, 175,
185
    statictext #main.NumberTxt,  "Item Number:", 200,   7,  80,  25
    statictext #main.NumberDisp, "",             300,   7,  95,  25
    statictext #main.NameTxt,    "Item Name:",   200,  32,  80,  25
    statictext #main.NameDisp,   "",             300,  32,  95,  25
    statictext #main.PrizeTxt,   "Item Prize:",  200,  57,  80,  25
    statictext #main.PrizeDisp,  "",             300,  57,  95,  25
    button #main.add,   "Add Item",   [add],       UL, 200, 112,  63,
  25
    button #main.edit,  "Edit Item",  [edit],      UL, 275, 112,  63,
  25
    button #main.delete,"Delete Item",[delete],    UL, 350, 112,  75,
  25
    button #main.search,"Search",     [search],    UL, 200, 162,  63,
  25
    button #main.exit,  "EXIT",       [quit.main], UL, 350, 162,  39,
  25

    open "Simple Database Framework" for window as #main
    print #main, "font ms_sans_serif 10"
    print #main, "trapclose [quit.main]"
    #main.itemlist "singleclickselect"
    wait

[add]
    extension$ = "add"
    gosub [CheckButton]
    wait

[edit]
    extension$ = "edit"
    gosub [CheckButton]
    wait

[delete]
    extension$ = "delete"
    gosub [CheckButton]
    wait

[DisplayItem]
```

```
    'get index of selected item
    #main.itemlist "selectionindex? SelectedItem"

    #main.NameDisp word$(items$(SelectedItem), 1, chr$(0))
    #main.NumberDisp word$(items$(SelectedItem), 2, chr$(0))
    #main.PrizeDisp word$(items$(SelectedItem), 3, chr$(0))
    wait

[search]
    'search in the database
    WindowWidth = 430
    WindowHeight = 190

    'position of dialogs are relative to previous open window
    UpperLeftX=1
    UpperLeftY=1

    textbox    #search.String,   5,   5, 175,  25
    button     #search.default,    "Search", [doSearch],  UL, 200,
   5,  75,  25
    listbox    #search.itemlist,   search$(,[doDisplay],   5,  35,
175, 120
    statictext #search.NumberTxt,  "Item Number:", 200,  35,  80,  25
    statictext #search.NumberDisp, "",             300,  35,  95,  25
    statictext #search.NameTxt,    "Item Name:",   200,  60,  80,  25
    statictext #search.NameDisp,   "",             300,  60,  95,  25
    statictext #search.PrizeTxt,   "Item Prize:",  200,  85,  80,  25
    statictext #search.PrizeDisp,  "",             300,  85,  95,  25
    button     #search.cancel,     "Close",[quit.search],  UL, 300,
127,  63,  25

    'modal windows block access to the previous window
    open "Search Database for Name" for dialog_modal as #search
    print #search, "font ms_sans_serif 10"
    print #search, "trapclose [quit.search]"
    #search.itemlist "singleclickselect"
    wait

[doSearch]
    redim search$(MaxItems)
    foundItem = 0

    ' search by name = field 1
    FieldNumber = 1

    #search.String "!contents? SearchString$"
```

```
    for Count = 1 to MaxItems
        'ignore case using LOWER$()
        if instr(lower$(word$(items$(Count), FieldNumber, chr$(0))),
lower$(SearchString$)) > 0 then
            foundItem = foundItem + 1
            search$(foundItem) = items$(Count)
        end if
    next

    #search.itemlist "reload"
    #search.itemlist "selectindex 0"
    wait

[doDisplay]
    'get index of selected item
    #search.itemlist "selectionindex? index"

    #search.NameDisp word$(search$(index), 1, chr$(0))
    #search.NumberDisp word$(search$(index), 2, chr$(0))
    #search.PrizeDisp word$(search$(index), 3, chr$(0))
    wait

[quit.search]
    close #search
    wait

[quit.main]
    close #main
    END

[CheckButton]
    'select action based on pushed button
    select case extension$
        case "add"
        SelectedItem = MaxItems
        DialogCaption$ = "Add Item"
        gosub [DisplayDialog]

        case "edit"
        DialogCaption$ = "Edit Item"
        if SelectedItem > 0 then gosub [DisplayDialog]

        case "delete"
        if SelectedItem > 0 then gosub [DeleteItem]
    end select
```

```
    'refresh listbox contents
    #main.itemlist "reload"

    'cancel selection to allow reselection of currently selected item
    #main.itemlist "selectindex 0"
    return

[DisplayDialog]
    'Form created with the help of Freeform 3 v01-28-07
    'Generated on Jun 19, 2007 at 22:59:56

    WindowWidth = 275
    WindowHeight = 195

    'position of dialogs is relative to previous open window
    UpperLeftX=1
    UpperLeftY=1

    statictext #item.NumberTxt, "Item Number:",  10,   7,  80,  25
    statictext #item.NameTxt,   "Item Name:",    10,  42,  80,  25
    statictext #item.PrizeTxt,  "Item Prize:",   10,  77,  80,  25
    textbox #item.Number, 105,   7, 150,  25
    textbox #item.Name,   105,  42, 150,  25
    textbox #item.Prize,  105,  77, 150,  25
    button #item.cancel,  "Close",[quit.item], UL,  95, 127,  63,  25
    button #item.default, "Apply",[apply],     UL, 180, 127,  75,  25

    'modal windows block access to the previous window
    open DialogCaption$; " - "; SelectedItem for dialog_modal as #item
    print #item, "font ms_sans_serif 10"
    print #item, "trapclose [quit.item]"

    if SelectedItem <> MaxItems then
        #item.Name word$(items$(SelectedItem), 1, chr$(0))
        #item.Number word$(items$(SelectedItem), 2, chr$(0))
        #item.Prize word$(items$(SelectedItem), 3, chr$(0))
    end if
    #item.Number "!setfocus"
    wait

[apply]
    ' apply changes
    #item.Number "!contents? Temp1$"
    #item.Name "!contents? Name$"
    #item.Prize "!contents? Temp2$"
```

```
' Make sure info in boxes is the proper type of data (number/string)
    if Temp1$ = str$(val(Temp1$)) then
        Number = val(Temp1$)
    else
        ' Item entered in the Number box is not a number !
        notice "Item Number must be numeric only."
        wait
    end if
    if Temp2$ = str$(val(Temp2$)) then
        Prize = val(Temp2$)
    else
        ' Item entered in the Prize box is not a number !
        notice "Item Prize must be numeric only."
        wait
    end if

    'fill the array element with the data

'separate fields by CHR$(0) to display only the first field in the lis
tbox
    items$(SelectedItem) = trim$(Name$); chr$(0); Number; chr$(0)
; Prize

    gosub [ApplyItemData]
    wait

[quit.item]
    'exit dialog
    close #item
    return

[ApplyItemData]
    gosub [BackupDB]
    gosub [OpenDB]
    gosub [WriteDB]
    gosub [ReadDB]
    gosub [CloseDB]
    return

[DeleteItem]
    confirm "Delete Item ... "+str$(SelectedItem)+chr$(13)+_
        "Name ..... "+word$(items$(SelectedItem), 1, chr$(0))+chr$(
13)+_
        "Number ... "+word$(items$(SelectedItem), 2, chr$(0))+chr$(
```

```
13)+_
        "Prize .... "+word$(items$(SelectedItem), 3, chr$(0)); answer

    if answer then
        items$(SelectedItem) = ""

        gosub [BackupDB]
        gosub [OpenDB]
        gosub [WriteDB]
        gosub [ReadDB]
        gosub [CloseDB]
    end if
    return

[OpenDB]
    'open database and define record length
    open "database.dat" for random as #db len=150

    'set the fields, include some extra space for future use
    field #db,_
        40 as ItemName$,_
        10 as ItemNumber,_
        10 as ItemPrize,_
        90 as Reserve$
    return

[CloseDB]
    close #db
    return

[ReadDB]
    'get the number of records in the database
    '= length of database file divided by the record length
    TotalRecords = lof(#db)/150

    'check if the database is corrupted
    if TotalRecords <> int(TotalRecords) then
        notice "Database corrupted"; chr$(13);
"Please check its contents!"
        TotalRecords = int(TotalRecords + .5)
    end if

    'dimension array to enable adding one record
    MaxItems = TotalRecords + 1
    redim items$(MaxItems)
```

```
    for Record = 1 to TotalRecords
        get #db, Record

        'fill the array with the data

'separate fields by CHR$(0) to display only the first field in the lis
tbox
        items$(Record) = trim$(ItemName$); chr$(0)
; ItemNumber; chr$(0); ItemPrize
    next
    return

[WriteDB]
    Record = 1

    for Count = 1 to MaxItems
        if items$(Count) <> "" then
            ItemName$ = word$(items$(Count), 1, chr$(0))
            ItemNumber = val(word$(items$(Count), 2, chr$(0)))
            ItemPrize = val(word$(items$(Count), 3, chr$(0)))

            put #db, Record
            Record = Record + 1
        end if
    next
    return

[BackupDB]
    if FileExists("database.bak") then kill "database.bak"

    name "database.dat" as "database.bak"
    return

function FileExists(FilePath$)
    ' returns zero if file does not exist
    ' returns one if file exists
    dim FileExistsInfo$(1,1)

    files "", FilePath$, FileExistsInfo$(

    FileExists = val(FileExistsInfo$(0,0))
end function
```

[Back to Top]