This package provided as is, for evaluation purposes only, with no warranties of any kind.

5/8/92

HUGE is a Windows 3.1 source code huge string array manager for Microsoft Visual Basic that allows the creation of fixed length string arrays of up to 1MB on 80286 system and up to 16MB on 80386 systems, within Visual Basic, without the need for a DLL.

This "package" will be included in a product I am developing, and is included here on CIS for evaluation use and feedback.

Please use (at your own risk, I warranty *nothing*) and provide feedback to me about its performance. Informal test indicate it operates very quickly and safely. Please provide feedack to Hank Marquis, CIS 76120,2413.

HUGE requires (and checks for) the version of Windows to be 3.1 or higher. HUGE is implemented to operate on fixed length strings.

HUGE consists of

>   HUGE.BAS - huge array engine
>   HUGE.FRM - huge array simple demo
>
>   HugeDIM - create a huge array
>   HugeZAP - closes huge array
>
>   HugeGET - get an huge array element
>   HugePUT - store an huge array element
>
>   HugeERR - return english error message

You can create up to 10 (arbitray limit really, changed by setting ha() DIM statement in HUGE.BAS main code section) huge arrays only limited by Windows system memory available. HUGE arrays do not use VB memory for storage and thus allow greated program size. Also, you can create much larger string arrays that you can in basic using HUGE.

Each call returns a status indicating sucess of failure. status% = 0 indicates the call executed with no errors. status% > 0 indicates the call failed, and status% contains a failure code. Codes from 170 to 174 are HUGE failure codes. Other values are Windows or VB failure codes. For HUGE failure codes you can use HugeERR() to return an english result.

Sub HugeDIM (handle%, status%, recordsize%, elements%)

HugeDIM creates a huge fixed-length string array and returns a handle which you use to access the array with HugeGET() and HugePUT().

handle% is returned as the logical handle to the huge array created.
status% is returned indicating the sucess or failure of the call (>0 indicates an error occured)
recordsize% is passed indicating the size of each array element
elements% is passed as the number of elements this huge array will contain

Sub HugeGET (handle%, status%, element%, recdata$)

HugeGET returns a string stored in the huge array.

handle% is passed as the logical handle to the huge array.
status% is returned indicating the sucess or failure of the call (>0 indicates an error occured)
element% is passed as the element to retrieve
recdata$ is returned as the string stored in element%

Sub HugePUT (handle%, status%, element%, recdata$)

HugePUT stores a string into the huge array.

handle% is passed as the logical handle to the huge array.
status% is returned indicating the sucess or failure of the call (>0 indicates an error occured)
element% is passed as the element to store into
recdata$ is passed as the string to store in element%

Sub HugeZAP (handle%, status%)
HugeZAP erases a huge array, and de-allocates its memory back to Windows.

handle% is passed as the logical handle to the huge to erase.
status% is returned indicating the sucess or failure of the call (>0 indicates an error occured)

It is very important to use the HugeZAP to close each array created using HugeDIM. HUGE
arrays are Windows elements - not VB elements. Simply ending your VB program will not
deallocate the memory!

Function HugeERR (status%) As String

HugeERR returns an eglish error message of the any error that occured, as indicated in the value
of the status% variable passed.

status% is a non-zero status% returned from any of the HUGE calls.

Usage

See the HUGE.FRM, form_load() procedure for an example is using HUGE.

Use HugeDIM to create an array
Use HugePUT/HugeGET to access/store strings into the array
Use HugeZAP to erase the array at the end of the program
Use HugeERR to return an error message if any HUGE routine returns a status > 0.

Techical Notes

HUGE uses Windows API calls to create and manage huge fixed length string arrays. The API
calls used are:

GlobalSize - returns size of allocated memory block
GlobalAlloc - allocated Windows memory
GlobalFree - releases memory allocated by GlobalAlloc
GlobalLock - returns handle to memory allocated by GlobalAlloc and locks it in place
GlobalUnlock - releases use count on memory
lstrcpy - used in HUGE to return a windows long pointer to a Visual Basic string
hmemcpy - used to move memory from huge array to Visual Basic
GetVersion - used to return Windows version running (checks for 3.1)