IBasic Monthly

Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

In This Issue

IBasic Monthly -Volume 1, Issue 2
Editor – Tony Jones - editor@ibasicmonthly.net
Assistant Editor – Rick Lett -rickl@ibasicmonthly.net

Submission Guidelines
If you would like to have your work considered for publication in IBasic Monthly, please forward your submission in RTF, DOC, HTML or TXT format to: submissions@ibasicmonthly.net. IBasic Monthly is published on the 15th of every month. All submissions must be in by the 8th of the month in order to be considered for the current issue.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Editors REMarks

Hello and welcome to the 2nd issue of IBasic Monthly! Judging from the response to the first issue, I think we are off to a fantastic start! The response was simply overwhelming and more than I could have ever imagined. Thanks to all of you for your support, kind comments and of course your submissions.

The response to the first issue was so positive that IBasic Monthly now has its own home on the web at www.ibasicmonthly.net, where you'll always be able to find the latest issue, along with the accompanying source code, the online edition of IBasic Monthly and back issues.

In addition to having our own home on the web, we've had to expand our staff as well. I would like to take this opportunity to introduce and welcome Rick Lett to IBasic Monthly. Rick will be taking on the role of Assistant Editor. He's been a valuable asset and great help with this month's issue. So when you see him on the forums or in the chat room, give him a pat on the back for a job well done.

You'll also notice that we are sporting a new logo this month, courteousy of Dan Silverman. He did an outstanding job on it and we would like to send a big thank you to him for donating his time and work to IBasic Monthly!

We have a huge issue for you this month, packed with articles, code and information. Bizzy continues his tutorial on using Linked List and takes us step by step to finishing up his ftp program in part 2 of FTP Now! John Sylvester gives us a different perspective on what it takes to write a program in Soul In Torment and we continue our journey with Rick Lett in Adventures In IBasic. Matt Cox demonstrates how to determine the version of Windows in Inside The Windows API and Jerry Muelver provides us with a way to use use hashes with ibHash: Faking Associative Arrays In IBasic. Paul Love has a puzzle for you in Of Jigsaw Puzzles, Mice And Me, plus he shows us how we can use Javascript in our IBasic programs in Javascript Jukebox. And if you've ever had trouble keeping your controls aligned in a window, then you'll definitely want yo read Graham Sutton's article, Re-Sizing Windows To Fit The Screen. And you don't want to miss our new features, including Developers Notes by Paul Turley, along with Tips And Tricks, Freeware Reviews and The IBasic User Profile Page.

Sincerely,
Tony Jones
Tony Jones
Editor – IBasic Monthly

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Developers Notes

By

Paul Turley

This is the first developer's notes column and this month I am going to just ramble on a bit. When the editor of IBasic Monthly first aproached me with the idea of a regular monthly column I was going to decline. But after thinking about if for a while I figured it would be a good place to share my thoughts and ideas on IBasic and the world in general.

The end of the year is fast approaching and development of IBasic Pro is proceeding at a steady pace. As I had mentioned on the Pyxia Forums I hope to have the first beta version ready by Christmas. The general release is slated for sometime in January, depending on how the beta release shapes up between now and then. Everyone that has pre-registered IBasic Pro will have a chance to try out the beta release.

For those that may be wondering about the future of IBasic Standard, which is the current software with a new name, you can put your fears to rest. Development on IBasic Standard will resume as soon as Pro is released. I have a lot of exciting new plans for IBasic Standard that will keep you happily coding along for years to come.

As the holidays approach we turn our thoughts away from programming and computers. Family regains its importance and many of us take a step back to consider all of the events that have shaped our lives over the past year. My family is no different and relatives I have not heard from since last Christmas begin sending me mail, in the form of greeting cards or nice letters with pictures. A few technologically adept members of my family have resorted to sending electronic cards and greetings. While very nice I still prefer the time honored tradition of licking a stamp and mailing the card. I actually save every card ever sent to me, collecting dust in some shoe box.

What does this have to do with IBasic or programming? The IBasic community is one very big family now. And while I can't send a greeting card to all of you this year, I will be giving away some registrations to IBasic and IBasic Pro to those that might not otherwise be able to afford it. I invite all reading this to follow along and give of yourself and your time in some small way to help others. Perhaps next year when you look at the events that shaped your lives you will remember the happiness you were able to share with someone in need.

Happy holidays one and all,
Paul Turley
Pyxia Development

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Opinion

Programming Guru's and Eclectic Thoughts

By

Rick Lett

Hi, just thought I would like to say a few words about some stuff that's been on my mind lately and may be fodder for discussion. Mostly about my search for a programming Language and peoples fierce loyalty to them.

I'm sure most of you did as I did and looked around at other forms of Basic before making your choice and may have tried others. I for one bought 2 along with IBasic and tried them both but for some reason that I can't explain I kept coming back to IBasic, I just like it. It's like a puzzle to me that I need to figure out how to solve, I guess.

Anyway, one of the things that I did do was look at the forums and support of the other languages and while those things are really not part of what I want to write about, that in the end is what made my decision for me. The thing that struck me in a lot of those other forums was the amount of bashing going on of other programming languages and that was kind of disturbing to me in a way, and I thought about why people do that.

What are we programming for, is one Basic better than another, and what as individuals do we want to get from doing this? For beginners it can be frustrating and for experienced users I'm sure very satisfying but you still see the discussion of which is better and which one stinks!

Paraphrasing Paul Turley a little, any good programmer should be able to accomplish a needed task with whatever tools are provided him.Thats the Pros though, how bout others? The not Pros? Maybe compelled to it like I was or maybe could better understand whatever you're using over the others?

Why your using what you're using is not as important I would think as much as what are you trying to accomplish. The purpose of programming to allow a user to interface with a computer in a useful or entertaining way so as to facilitate its ease of interface. Programming is not an end to itself, something useful has to be created. And a good programmer has to be well versed in the subject that he's trying to create. That to me is what separates the hobbyist from the

Pro I guess. How many of us on the IB forums could even conceive of writing a programming language, or any of the other forums? I'm sure there are some who could write massive code for whatever purpose they envision, but not all can.

I for one am just learning, and I can't tell you how much I admire the skill and use of IB or any other language being used by anyone who is passionate about that language, let alone someone who can create a tool that so many others use and enjoy. But I for one don't see it as one upmanship or elitism that one language is used and better than another. I see it as a comfort issue for the folks out there that are still learning. Someone who is well versed in programming knows that there is no perfect single language for all users. And can adapt with the situation as needed to accomplish what needs to be done.

So there I would think is the crux of the issue. In my opinion (and humble it is!) I believe the bashing is done mostly by people who are just starting to become comfortable with what their using but not quite ready to become serious about it. Or are uncomfortable with the perceived limitations of what they're using. I would think that in this medium any tool that will do the job is the first place Pros go to. IB is written in a mix and is done so for a reason, because that was the way to best accomplish the goal. And other Basics I'm sure have a mix that they are written in. I for one am glad that there are others to choose from so that people have a choice as to what's more comfortable to use. But as much as I like IB I hope to someday learn other methods of programming so I can be more versatile when I need to be.

So, while others are bashing various Basic languages I'm going to say Thank You Paul for offering IB so I can take that first step in programming and thanks for your help, and I would suggest others thank the developers for their preferred language also.

The views and ideas expressed in this article are those of the author and may not represent the views of IBasic Monthly. If you have any comments or suggestions you may email the author rickl@ibasicmonthly.com. By submitting a response you give IBasic Monthly permission to publish in a letters to the editor format in a subsequent issue and to edit for space and content. IBasic Monthly is intended for G type viewing so please make all comments constructive.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Soul In Torment

By

John P. 'Tuddypat' Sylvester

To be slightly different, I thought it would be good to try to explain the thought processes and decisions and dark forces at work during the development of a project rather than the technical details of programming. Am I the only one to suffer this way?

Let's start with a brief history; I trained as an Electronics Fitter and now work as a Field Service Engineer in Computers. I've been dipping in and out of programming in various languages over the last 25 years, using machine code on Raytheon computers (well actually binary, on a CE Panel, for the uninitiated, that's a box with lots of switches and lots of pretty lights). I've also done machine code on Z80, 6509. I've had a go at assembly on PDP-11 micro as well as 6809 and 8086 processors. At the higher end, I started on BASIC, MSBASIC, C, C++, Pascal, Java++ and Smalltalk and probably a few others on the way. I achieved a reasonable understanding of the languages and their use but never achieved expert status in any, I haven't even mastered English that well either. I have completed courses covering software engineering and software development. Old habits die hard, I still tend to plan at the keyboard and solve problems on the fly, besides it's a lot more fun thinking on your feet than planning on paper then committing to coding. One day I will have the resolve to do it properly, until then, this is how it goes…

I attempted at several stages to enter into Windows programming but it never seemed to gel for me until, yes, you've guessed it, IBASIC came along, it was like a fog being lifted. It took only a little while to achieve a degree of familiarity with the language and syntax it was not long before I was plunging headlong where previously I feared to go. I started writing little test programs to try out my new knowledge and plaything. After numerous modifications to the sample programs I decided the time had come to start a proper project.

The project was a Stock Control program. This was needed on a personal basis, since I carry a fair amount of parts with me and I have about thirty parts a week going in and out. At times more comes in than goes out and it takes some doing to keep track of them, like what I have and what job they're for. With previous companies they always managed to ask where a part was that I had had several months before and it meant going through a mountain of paperwork to track it down. Basically all the program had to do was allow me to record receipts, usage and disposal of parts, easy!

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

The Stock Control project started well but soon I had a source file that was getting a little unwieldy, so I decided to break it up into smaller files, each file holding a single entity of the program. This meant that I could list the program and any changes would only require re-listing the changed file.

Once I had spent a few nights reorganising things I was left with around sixty files to manage. This in itself was proving a task just managing the update of the files. However, that is another problem and maybe another article, (that will keep Tony happy). Having made life easier in one respect I now had sixty files to join together before I could run the source. Well you can imagine the command line:

’copy file1.txt+file2.text+file3.txt+file4.text etc. etc. etc’

So I wrote a batch file, but if I added a file I had to edit the batch file, this was starting to be a nightmare. So the Stock Control went on hold as I started to develop another program to make things easier. Of course I was deluding myself, this started well until I realised I had to have someway of marking the files and ensuring they were in the right order. I started entering the world of directory listings.

What I wanted was a routine that would list files, allow me to select them and output the list of selected files in a string, ready to process. It so happens that IBASIC has two very handy instructions, FINDOPEN and FINDNEXT.

With these two instructions one can open a directory and list all files within that directory, as a start I used a directory containing the source files for the Stock Control program. The routine worked like a well-oiled machine. However, it was not long before dissatisfaction set in and I realised it would be better if I could traverse through directories. So I modified it based on the ’dirselector' example that comes with IBASIC. It was then I realised the problem … FINDOPEN lists everything in a directory including sub-directories, and there was no way to differentiate between a directory and a file with no extension. I'd reached a dead end.

There was only one thing to do give up and do something useful. So I went and read the forum, surfed the internet, played games and everything else one does. However, fate would not let go that easily, one of the forum users asked the question if anyone knew about directory listings and I mentioned that I had such a beast, although not perfect, it could be useful or at least show the way to go about it. I emailed it to him. I also remarked on the forum that if anyone else was interested let me know. That was the mistake! It's like jumping off a cliff, as your feet leave Terra Firma, you wish you had thought a little more of the consequences as you plummet headlong towards the unknown. Thanks to a kind offer by one of the guru's of the forum, 'Fletchie', he offered to post it on his website for all to download.

It wasn't long before a couple of users reported problems, mainly under XP. There had been some comments about the unreliability of the SetCurrentDirectory API, which for some inexplicable reason, I had used. Looking through the code I found that it was redundant along with some other bits of code. This is what happens when you work from the keyboard. Anyway I decided at that point to redesign the whole thing and of course having already displayed the ability to open my mouth and insert both feet, I put this on the forum. Well that meant I had to do something now.

So I started with a complete blank sheet and proceeded to build it from scratch.

I decided simplicity was the key; it took the form of three list boxes, one for the drives, one for the directories and one for the files. Well of course having done most of the experimentation on the original version, it only needed refining. I decided not to use FINDOPEN/FINDNEXT functions and rely on sending messages to the list box using flags to control the listings. These flags had been gleaned from several sources the Microsoft web site and the 'include' files for Borland C++ (winuser.h). Of course, when implemented the lists behaved as expected, but they had short names and were unsorted, still it was a start. The sorting was easy to get round, change the flag on the list box to include the sort ability. It worked fine until I decided to list the \windows\system directory. Well it was fun, the file names started doing a Bossa Nova movement or was it the Cha-Cha? As each new name was entered the file names shuffled around and the slide bar at the side grew smaller and smaller as the shuffling intensified. Eventually it stopped the dancing motion and the file names were displayed and sorted but my eyes were dancing to the rhythm…this was not good.

I added two hidden list boxes with sort enabled (removing the sort from the main box), this would allow the directories and files to be sorted quietly and out of the way. Once the directories and files were listed in the hidden boxes it was just a matter of copying the names from the hidden boxes to the view boxes. I have always been a great believer that no matter what the program is if a finite time is needed to process, the user should be aware that something was happening. As I selected the \windows\system directory the two main boxes went grey for a while only seconds in reality but in electronics, computing and waiting, seconds often seem like hours, in fact, it often is hours when the thing hangs and you have no indication what's going on especially where grey boxes are concerned.

Anyway the result was worth waiting for, it finally displayed, all in order, the added advantage was the directories were delimited by square brackets, easily identifiable as a directory, the drives were shown delimited by square brackets and hyphens ([-a-]), this was getting better.

The selection and changing of drives, directories and files was simple it was the formatting and appearance of the results that was the challenge; the only thing to overcome now was the short file names! To this end I delved into the mysteries of API. Using that wonderful program API Viewer I found the GetLongPathNameA function call, so I incorporated into the code. The sequence of events now went, read the directory into the sorted field then for each directory and file as the entry was being copied from the hidden box to the visible box find the long name and display that. To tidy things up, the directories were converted to upper case and files to lower case during this transfer/conversion operation. Well you have to take the prettiness factor into account! This worked until I tried to change the directory, then nothing happened, no directories, no file names, only drives. After some thought and panic laden activity I discovered that the long path names were not accepted by the directory list messages, so I now had to convert the pathname back to the short version to be able to list the directories. Eventually I managed to get the whole thing working, an elated sense of achievement arose inside me giving me a nice warm feeling…oops back in a minute!

Of course it was not to last long, that little voice inside my head, no, not the 'Kill them all…' one, the other one, the 'Why don't you make it do…' one, spoke.

'It would be much better if it were a single window!' and so I thought well instead of copying the files and directories to their respective windows just copy them to the drive windows, which I did…you should have seen the mess! I forgot the list box had been set to sort. A quick adjustment and there it was working, of course I had to adjust all the selection routines to cater for the new box. Anyway I was happy the project had been completed. Now why was I so naïve? I posted the completed code onto the forum. It was received well, but there's always someone, I shall not name them, your secret is safe with me 'Fletchie' suggested 'Why not make it a component?' plus as a side comment 'Could add check boxes to allow the selection of drives, directories, system and hidden files.'

Well always one to rise to a challenge, I had no idea about components, but what the hell, the best way is by doing it. Any way before that I had to modify the code to allow the selection, this wasn't too difficult since the code already existed and just needed so form of control. So the dialog was modified to include the check boxes the code altered to read them and set up the required messages. Within a while it was working like a dream…or nightmare…depending on your point of view. Now was the time to look at it as a component. So I read up on the use of components and converted it to a component, the actual conversion was straightforward; most of the boring work was converting the names as agreed in the component code of practice that had been generated within the forum. Once it was done, before making it into a component, I ran it as a standard program, well my world crashed about me. No directories or files were displayed, it was then I realised that my base path was not available to all the subroutines. Its surprising how simple restrictions like, 'no global variables' causes a problem. I had all the subroutines

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

written but how could I pass the base path around without the use of a global variable? I couldn't! I thought about passing the path as a parameter but the path was needed within the dialog handler, there was no way I could think of passing that parameter to the handler. I also found I could not access the dialog from within the subroutines; I kept getting the message that the dialog was not open. I had hit that brick wall…again!

I decided that this challenge had me beat and posted on the forum that I could get no further and I had no way of making this a component and added if anyone could help, it would be appreciated? In true IBasic fashion a shining knight rode out of the mist and threw at my feet a rope of hope…one Jerry Muelver, another Guru of the forums! He pointed me to his Wiki website and a page about components. I searched through it to no avail, there was no more information than I had already, and obviously components were not meant to be the complex items I wanted them to be. At that moment I spotted a component by another veteran of the forum, 'Kludge'.

In it he was giving a demo of using a custom control within a component, and there was the answer I had been looking for. 'Kludge' had created a pseudo global variable by defining a user type and storing it in memory, all that was needed was to be able to pass the pointer. This was achieved by storing the pointer in the parent window's user data area, and all that was required (not really that simple) was to find the parent and extract the pointer to the data in memory, there also was the solution to my problem of the dialog not being open, a system variable giving the present dialogs id!!! It took me a little while to understand what it did and how it fitted together but after a short while it had all come together, it worked like a dream, I completed the component packing and had tested it successfully, at last I had achieved my goal. I announced the to the world that I had completed the project and now I was free to regain control of my life and resume some sense of normality.

I think it only fair to give thanks to those who helped, to Paul (Pyxia) without whom this would never have happened, to 'Fletchie' for the challenge, Jerry for the wiki web site and of course 'Kludge' who provided the solution. All these guys bear some responsibility for my torment and without who, the project would never have started or finished and of course the other four hundred odd users who unwittingly helped to solve problems with their entries on the forum.

Now where was I, oh yes! The Stock control program, but first I wonder if I can…………..

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Tips And Tricks

Simple Indexing Scheme

By

David “entiretech Harrison

I have been putting together a small program, partly to learn the syntax and possibilities and to work out a couple of ideas and also to write a simple relational database in native IBasic code. The program is a diary with space for an unlimited number of appointments for any given day. The Diary dates are saved using a simple hashing algorithm, this needs a record for every date so could have a number of blank entries, this is acceptable were it is reasonable to expect that most records will contain data - Diary dates, invoice numbers etc. it is the quickest possible way of retrieving a specific record, a simple calculation and you have the record number.

Any number of appointments can be added in a seperate table "related" to the main table by the date (the key).

The problem was with keeping the list of appointments sorted this can be pretty slow. It occurred to me that you could use a listbox as an array with automatic (quick) sorting. To collect the appointment details for all dates I load a listbox from a file, the file contains the date and the record number of the appointment data file, a binary search is used to find a starting position for the search.

The basic idea is shown in the part program submitted; it is a simple indexing scheme.
The program loads a lot of pseudo data for testing, a sequential sort is used this can be slow depending on the size of the list and the position of the target. Binary search is used to find a starting point, which reduces the search time and gives a consistent search time regardless of the size of the list.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

The found items are removed from the list - the idea is that they are moved to a second list and the relevant records from the data file retrieved for modifications and additions, when they are completed (you move to a different day or key) the new and modified items are added to the original list, this list is in effect an index.

If you are testing this try the sequential search first, use the save button, then the load button to restore the list and see the difference using the binary search. This is just for the testing stage if this were used in a real situation the list would remain in memory to save a lot of time. Essentialy when adding a record to the data file you would also add the key (in this case the date) and the record number to the list. (index). When searching for a given key this routine would find all matches and their relevant record numbers.

This description is a bit convoluted (could use more time to re-write), but the code is commented and should be self-explanatory.

I also have a routine for allowing deletions and additions to the data file using a linked list of available record numbers, unfortunately the comments are mainly in my head, I will work on it and submit next month.

```
' searching a list box for a sequence of identical values

autodefine "off"
declare BSearch(lBound as int,uBound as int,winuse as window,id as int,Target
as string)
declare SSearch(lBound as int,uBound as int,winuse as window,id as int,Target
as string)
declare "kernel32", GetTickCount(),int

def maxsize as int
' vary this number for testing
maxsize = 1000
def  numelements as int
def start,fini as int
def run as int
def retval as int

' file definitions
def idxfile:BFILE
def fileLoc:string
def temp as string
fileloc = getstartpath
def fName:string
fName = "test.txt"

' the index file would be smaller if used integers, but the conversion for
loading into the list box
' creates a noticeable slowing down, the eternal tradeoff between space and
time
```

```
type testit
   def a[8] as istring
   def b[6] as istring
endtype
def t as testit
def searchitem[8] as istring

' setid for controls
setid "indexlist",100
setid "keylist",101
setid "recnolist",102
setid "SSearchBtn",110
setid "BSearchBtn",111
setid "SaveBtn",112
setid "LoadBtn",113
def MainWin:window
window Mainwin,0,0,300,250,0x80C80080,0,"Caption",main
control mainwin,"L,ListBox1,6,1,130,119,0x50A00142,@indexlist"
control mainwin,"L,ListBox1,6,127,130,95,0x50A00142,@keylist"
control mainwin,"L,ListBox1,140,127,125,95,0x50A00142,@recnolist"
control mainwin,"B,Sequential,222,0,70,20,0x50000000,@SSearchBtn"
control mainwin,"B,Binary,222,20,70,20,0x50000000,@BSearchBtn"
control mainwin,"B,save,222,40,70,20,0x50000000,@SaveBtn"
control mainwin,"B,load,222,60,70,20,0x50000000,@LoadBtn"

newlist

run = 1
do
wait
until run = 0
closewindow MainWin
end

sub main
'work code goes here.
select @CLASS
   case @IDCONTROL
      select @controlid
'           case 1
'           if @notifycode = @lbndblclk
'           endif
            case @SSearchBtn
                  ' sequential search
                  ' this demonstrates the speed improvement by starting the
search close to the
                  ' search value by using a binary search to find the start
index ( case @BSearchBtn)
                  def retval as int
                  start = GetTickCount()
                  numelements = getstringcount (mainwin,@indexlist)
                  if numelements > 0
                        Ssearch(0,numelements-1,mainwin,@indexlist,searchitem)
                        fini = GetTickCount()
                        messagebox mainwin,"finished in"+str$(fini-start),""
                  endif
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
            case @BSearchBtn
            ' Binary search
            ' Find all matching strings in a sorted list box (easily converted
for an array or sorted disk file).
            ' If less than ten items use a simple sequential search
            ' more than ten use a binary search to find the starting point to
speed up the search.
            ' Binary search exits with the first string position that it
finds, to allow for multiple matches
            ' step back until the returned value is less than the search
string and then
            ' use this value as the starting point in the search.
            numelements = getstringcount (mainwin,@indexlist)
            start = GetTickCount()
            if  numelements > 0
                  if numelements > 10
                        ' use a binary search to find an instance of the
search string
                        retval = bsearch(0,numelements-
1,mainwin,@indexlist,searchitem)
                        ' if not found BSearch returns -1 (can't use zero as
the list is zero based)
                        if retval >= 0
                              ' step back through the list until search string
and list string no longer match
                              while left$(searchitem,8) =
left$(getstring(mainwin, @indexlist,retval),8)
                                    retval = retval - 1
                              endwhile
                              ' use new retval as starting point for
sequential search.
                              ' the search routine also moves values to
CurrApps list box.
                              Ssearch(retval,numelements-
1,mainwin,@indexlist,searchitem)
                        endif
                  else
                        ' less than ten then simple sequential search
                              Ssearch(0,numelements-
1,mainwin,@indexlist,searchitem)
                  endif
                  fini = GetTickCount()
                  messagebox mainwin,"finished in"+str$(fini-start),""
            endif
            case @SaveBtn
            ' save
            ' CurrApps  are the appointments currently being modified
            ' Add them back to index listbox
            def CurrApps as int
            def newstr as string
            ' copy the listbox back to the index listbox
            CurrApps = getstringcount(mainwin,@keylist)
            if CurrApps > 0
                  for i = 0 to CurrApps -1
                        t.a = getstring(mainwin,@keylist,i)
                        t.b = getstring(mainwin,@recnolist,i)
                        newstr = append$(t.a," ", t.b)
                        addstring mainwin,@indexlist,newstr
                  next i
                  for i = 0 to CurrApps
                        deletestring mainwin,@keylist,0
                        deletestring mainwin,@recnolist,0
                  next i
            endif
            putall
            case @LoadBtn
            ’ loadfrom file to index listbox
            getall
   endselect
’ put these two at the end of the case structure, after all they will only be
called once each.
   case @idcreate
      centerwindow mainwin
    case @idclosewindow
      run = 0
endselect
return


’subs and functions after here
sub SSearch(lBound,uBound,winuse,id,Target)
’ simple sequential search of a sorted listbox to find matching strings
’ and process them as needed
   def i,alldone as int
   i = lbound
   alldone = 0
   ’ set value for target, only do calculation once
   def tt[8] as istring
   tt =left$(Target,8)
   def ss[8] as istring
   do
      ’ only calculate value for compare once for each iteratation. ss is the
string to search for.
      ss = left$(getstring(winuse, id,i),8)
      if tt= ss
            addstring winuse,@keylist,ss
            addstring
winuse,@recnolist,mid$(getstring(winuse,id,i),@keylist,4)
            ’ strings are transferred to the other listbox ready for
modification or deletion.
            ’ copied back before change day or save.
            deletestring winuse,id,i
            ’ step back 1 as next string will have same position due to
deletion
            i = i-1
      endif
      ’ when the searchitem value is less than the value returned from the
string number
      ’ we have gone past all possible matching values and so can save time
and exit the loop
      if tt < ss then alldone = 1
      i = i+1
      ' must trap last entry or "endless loop" is possible
      if (i = numelements) then alldone = 1
   until alldone
return

sub BSearch(lBound,uBound,winuse,id,Target)
' the binary search is very efficient, it requires a sorted list, array or
file.
' we compare the target to the value at the middle of the array/list/record
number
' if the target is higher than the midlle value set new boundaries
' from the midpoint to the upperbound ie the midpoint becomes the new
' lowerboundary - compare the Target to the value at the middle of the
' new list (now half the size).
' if the Target is lower we use the lower half of the new list
' continue this division until the Target value is found or
' until the upperboundary the lowerboundary and the midpoint are the
' same indicating that the Target value is not in the list
'
' If the Target is first or last there is no point searching
' This also saves problems with the typing of midPoint as int
' causing last item in the list being skipped
' ie. (8+9)/2 = 8.5 and will always return 8

' set value for target, only do calculation once
def tt[8] as istring
tt =left$(Target,8)
if tt = left$(getstring(winuse, id,lbound),8)
   return lBound
endif
if  tt = left$(getstring(winuse, id,uBound),8)
   return uBound
endif

def ss[8] as istring
def foundit, retval,midPoint as int
' set midpoint
midPoint = (uBound+lBound)/2)
' set initial values - returns retVal = -1 if Target is not found
' can't return zero for false if numeric - as zero based array.
retVal = -1
foundit = 0
while foundit = 0
   ' only calculate compare value once for the loop, always a good idea.
   ss=left$(getstring(winuse, id,midpoint),8)
   'search until found or is not in list
        If tt > ss
      ' Too high so calculate new midpoint
      ' from old midpoint to upper boundary
      lBound = midPoint
      midPoint = (midPoint+uBound) / 2
   endif
        endif
   if tt < ss
      'calc new midpoint using lower half of list
      uBound = midPoint
      midPoint = (midPoint+lBound) / 2
        endif
   if tt = ss
      'foundit
         foundit = 1
         retval = midPoint
   endif
   If midPoint = lBound
      ' Doesn't exist
      foundit = 1
   endif
endwhile
return retval

sub putall
setcursor mainwin,@cswait
start = GetTickCount()
' new file, replaces original with updated listbox strings
IF(openfile(idxfile,fileloc+fName,"w") = 0)
   def count as int
   count = getstringcount (mainwin,@indexlist)
   ' make sure we have something to put
   if count > 0
      for i = 1 to count
            temp = getstring(mainwin,@indexlist,i-1)
            t.a = left$(temp,8)
            t.b = mid$(temp,10,4)
            put idxfile,i,t
      next i
      ' delete from end of list upwards, save a bit for re-sorting the list
      ' this should be replaced with Fletchie et.al's API call to clear the
list box.
      ' and probably accounts for the difference between saving and loading
data.
      for i = 0 to count
            deletestring mainwin,@indexlist,count - i
      next i
   endif
   closefile idxfile
else
   mesagebox winmain,"Unable open file for saving index",""
endif
fini = GetTickCount()
messagebox mainwin,"saved "+str$(count)+" records in "+str$(fini-start)+"
milliseconds",""

setcursor mainwin,@csarrow
return

sub getall
def numrecs as int
setcursor mainwin,@cswait
start = GetTickCount()
' open file for reading
IF(openfile(idxfile,fileloc+fName,"r") = 0)
   ' using a for loop is quicker than testing for EOF
   numrecs =len(idxfile)/len(t)
   if numrecs > 0
      for i = 1 to numrecs
      get idxfile,i,t
      addstring mainwin,@indexlist,t.a+" "+t.b
      next i
   endif
   closefile idxfile
else
   messagebox mainwin,"unable to open index file",""
endif
fini = GetTickCount()
messagebox mainwin,"retrieved "+str$(numrecs)+" records in "+str$(fini-
start)+" milliseconds",""
setcursor mainwin,@csarrow
return

sub newlist
' build a number of pseudo dates for testing (t.a)
' please ignore the strange number of months/days in a year/month
' and the fact that the so called record numbers (t.b) are extremely
artificial
def newstr[14] as istring
t.a = "20020101"
t.b = "1"
newstr = "            "
if maxsize > 0
for i = 0 to maxsize
   t.a = left$(ltrim$(str$(val(t.a)+ int(rnd(10)))),8)
   t.b = left$(ltrim$(str$(val(t.b)+ int(rnd(2)))),4)
   newstr = append$(t.a," ", t.b)
   addstring mainwin,@indexlist,newstr
   ' make sure we at least 2 pseudo dates for the search
   if i =  int((maxsize/1.5))
      addstring mainwin,@indexlist,newstr
      searchitem = t.a
   endif
next i
endif
messagebox mainwin,"looking for "+searchitem,""
return
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

INC & DEC Routines

BY

Fidcal

If your program does a lot of... megaIndexCounter_LEFT = megaIndexCounter_LEFT + 1 and someThingImportant = someThingImportant - 1 then you might like to consider including one or both of the functions listed below. Then you can replace the above with...

inc megaIndexCounter_LEFT
and
dec someThingImportant

Ah! Reminds me of my Z80 assembler days!

```
declare inc(varPTR:pointer)
declare dec(varPTR:pointer)
openconsole
color 0,15
cls
n=99
print n
inc n
print n
dec n
dec n
print n
do:until inkey$<>""

closeconsole
end

sub inc(varPTR:pointer)
  #varPTR=#varPTR+1
return

sub dec(varPTR:pointer)
  #varPTR=#varPTR-1
return
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Editors Note: AlGonzalez followed up Fidcal's orinignal routines with this modified version.

```
Declare Inc(varPTR As Pointer, incValue As Int)
Declare Dec(varPTR As Pointer, decValue As Int)

Dim n As Int

OpenConsole
Color 0,15
Cls

n = 99
Print n
Inc n
Print n
Dec n
Dec n
Print n

Print "-----"
Inc n, 2
Print n
Dec n, 50
Print n

Do:Until Inkey$ <> ""

CloseConsole
End

Sub Inc(varPTR As Pointer, incValue As Int)
    If incValue < 1: incValue = 1: EndIf
    #varPTR = #varPTR + incValue
Return

Sub Dec(varPTR As Pointer, decValue As Int)
    If decValue < 1: decValue = 1: EndIf
    #varPTR = #varPTR - decValue
Return
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Inside The Windows API

How To Use The GetVersionExA API Call To Get The Operating System Details.

By

Matt "Lucifer" Cox

This month Matt Cox gives us a step by step approach to determining what version of Windows your application may be running under. Knowing what version of Windows your program is running under can be useful in a number of ways. There may be a time that your program uses a feature that is only available under say Windows 98SE or higher. Rather than having your program crash on Windows 95, you could check for the Windows version and have your program exit gracefully. You could also use the version routine as part of a larger program to generate diagnostic or system information for the end-user to submit with possible bug reports. All in all, a very useful routine to add to your API toolbox.

STEP 1 - You need to add the following TYPE to the top of your code within the same location as all the rest of your declarations.

```
TYPE OSVERSIONINFO
      DEF InfoSize:INT
      DEF MajVer:INT
      DEF MinVer:INT
      DEF Build:INT
      DEF Platform:INT
      DEF PSS[128]:CHAR
ENDTYPE
```

STEP 2 - You need to add a global variable to the top of your code within the same location as all the rest of your declarations. This will allow you to access the OS data from any point within your application.

```
DEF vertext$[5]:STRING
```

STEP 3 - You need to declare the GetVersionExA API call to the top of your code within the same location as all the rest of your declarations.

```
DECLARE "Kernel32", GetVersionExA(VersionInformation:OSVERSIONINFO),INT
```

STEP 4 - At the end of your code you need to add the following subroutine. This subroutine is what does the work and produces the OS details.

```
SUB OSVersion
      DEF ver$, ver2$:string
      DEF VersionInformation:OSVERSIONINFO
      VersionInformation.InfoSize=LEN(VersionInformation)

      IF GetVersionExA(VersionInformation) = 0
            vertext$ = "Error"
      ELSE
            ver$ =
str$(VersionInformation.MajVer)+","+str$(VersionInformation.MinVer)+","+str$(
VersionInformation.Platform)
            SELECT ver$
                  CASE " 4, 0, 1"
                        ver2$ = "Win 95/OSR2"
                  CASE " 4, 10, 1"
                        ver2$ = "Win 98/SE"
                  CASE " 4, 90, 1"
                        ver2$ = "Win ME"
                  CASE " 4, 0, 2"
                        ver2$ = "Win NT4"
                  CASE " 5, 0, 2"
                        ver2$ = "Win 2000"
                  CASE " 5, 1, 2"
                        ver2$ = "Win XP"
                  DEFAULT
                        ver2$ = "Undifined"
            ENDSELECT
            vertext$[0] = ltrim$(STR$(VersionInformation.MajVer))
            vertext$[1] = ltrim$(STR$(VersionInformation.MinVer))
            vertext$[2] = ltrim$(STR$(VersionInformation.Build))
            vertext$[3] = ltrim$(STR$(VersionInformation.Platform))
            vertext$[4] = ver2$
      ENDIF
return vertext$
```

Let’s put a small demo together to get it working.

Example –

```
TYPE OSVERSIONINFO
      DEF InfoSize:INT
      DEF MajVer:INT
      DEF MinVer:INT
      DEF Build:INT
      DEF Platform:INT
      DEF PSS[128]:CHAR
ENDTYPE

DEF vertext$[5]:STRING
DEF a$:STRING

DECLARE "Kernel32",GetVersionExA(VersionInformation:OSVERSIONINFO),INT

OPENCONSOLE
OSVersion
PRINT "OPERATING SYSTEM DETAILS"
PRINT "========================"
PRINT
PRINT "Major Version    : "+vertext$[0]
PRINT "Minor Version    : "+vertext$[1]
PRINT "Platform         : "+vertext$[3]
PRINT "Build            : "+vertext$[2]
PRINT "Operating System : "+vertext$[4]
PRINT
INPUT "Press ENTER exit",a$
CLOSECONSOLE

END

SUB OSVersion
      DEF ver$, ver2$:string
      DEF VersionInformation:OSVERSIONINFO
      VersionInformation.InfoSize=LEN(VersionInformation)

      IF GetVersionExA(VersionInformation) = 0
            vertext$ = "Error"
      ELSE
            ver$ =
str$(VersionInformation.MajVer)+","+str$(VersionInformation.MinVer)+","+str$(
VersionInformation.Platform)
            SELECT ver$
                  CASE " 4, 0, 1"
                        ver2$ = "Win 95/OSR2"
                  CASE " 4, 10, 1"
                        ver2$ = "Win 98/SE"
                  CASE " 4, 90, 1"
                        ver2$ = "Win ME"
                  CASE " 4, 0, 2"
                        ver2$ = "Win NT4"
                  CASE " 5, 0, 2"
```

```
                        ver2$ = "Win 2000"
                  CASE " 5, 1, 2"
                        ver2$ = "Win XP"
                  DEFAULT
                        ver2$ = "Undifined"
            ENDSELECT
            vertext$[0] = ltrim$(STR$(VersionInformation.MajVer))
            vertext$[1] = ltrim$(STR$(VersionInformation.MinVer))
            vertext$[2] = ltrim$(STR$(VersionInformation.Build))
            vertext$[3] = ltrim$(STR$(VersionInformation.Platform))
            vertext$[4] = ver2$
      ENDIF
return vertext$
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

My Adventures With IBasic

(Or how to frustrate yourself for fun!!)

By

RICK LETT
(Newbie)

The Mystery of <u>Select Case</u>

Well hello again all you (...Including me!)Newbie programmers! Well it looks like we get another month together, and hopefully many more (at least until I'm found out anyway!).

Last month we talked about our first program and did something a little different than a "Hello World "program. Fun *wasn't it!!*

Well this month we're going to expand on that and introduce a few more concepts. (Sorry folks, gotta move this along because I'll never be a programmer of IBasic by lollygagging around you know.)
Now since you can find your way around in Windows go ahead and cut and paste the program below into the IBasic editor can't miss it, it's that big white screen in front of ya.

Go ahead I'll wait.

```
Rem***************************************************************************

DECLARE "kernel32",Sleep(dwMilliseconds:INT),INT
Def random, time:int
openconsole
print"Demo of select/case to randomly"
print"print 3 colors to console screen using "
print"random numbers and random timer intervals"
locate 18,5:print"Notice that case represents each case selected"
locate 20,5:print"Timer shows the randomly selected timer amount"
locate 22,5:print"(in milliseconds)"
locate 24,5:color 9,0:print"<press any key to close>"
do
random = rnd(3) + 1
time = rnd(500) + 300
```

```
select random

      case 1

            locate 10,10
            color 0,12
            print "  ","case ",random,"timer ",time
            sleep(time)
            locate 10,10
            color 0,0
            print"                       "

      case 2

            locate 12,10
            color 0,14
            print "  ","case ",random,"timer ",time
            sleep(time)
            locate 12,10
            color 0,0
            print"                       "

      case 3

            locate 14,10
            color 0,2
            print "  ","case ",random,"timer ",time
            sleep(time)
            locate 14,10
            color 0,0
            print "                      "


endselect

until inkey$<>""
closeconsole
end

Rem*****************************************************************************
```

Well here we are on the other side, go ahead hit **run** and *look at her goooo!!*
Pretty neat huh? Took me all afternoon to come up with that one. Now we got to figure out how it works.

Just hit any key and read on.

Ok first off remember when I *alluded to* a better way to *time out your program last month?* Well it's time to talk about that a bit.

Notice the first line.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

DECLARE"kernel32",Sleep(dwMilliseconds:INT),INT

Now unless you've been reading a lot since last month this is a **DECLARE** statement from the windows **API or Applications Programmer Interface,** some of you knew that, some of you didn't.

By the way, you would do well to down load the **ApiViewer 2002**, an excellent tool with an IBasic plug in that shows these statements in IBasic syntax, *get it, and use it.* And by-by the way, you can read an excellent article from last months issue on the window **API**.

Now I know your asking (this is you but not whinny)*"hey, why do I need this any way, that for-next thing worked just fine to time out last time"* .Ya know I'm glad you asked that.

The thing is if you look at that statement you'll see in the parenthesis **dwMilliseconds**, now what that means is that a millisecond is a millisecond no matter what clock speed your computer is running at. How fast a for-next loop counts is determined by how fast your computers clock is. That means this program will run at the same speed no matter what clock speed your computer is running at. That's important so that it doesn't run too slow or too fast but at the rate you intended it to.

Ok, next we **DEF**ined our variables, very important, as **INT**egers then opened the console then **PRINT**ed some purty words to the console screen saying what we're doing. And it does help to know that ya know!

If your having trouble remembering about **Print, Locate, Color,** then shame on you, and you can review those commands in the users guide.

Next we need to create our random set of numbers. Since in our example we are going to check the condition of random 3 times we want to randomly generate any number between 1 and 3 so we say **random = rnd (3) + 1** (good thing IBasic isn't case sensitive). The + 1 is there so there will be no zero (0) value for random, for this program it wouldn't hurt because zero would be ignored, but that could be an important consideration later on (and no I can't think of an example right now so trust me ok).

We also need to set a random time value so our program will stop long enough for us to see the colors on our console screen. Now a millisecond is *one thousandth of a second,* which is pretty fast to us. So we need to give it a value that is useful to us, to see what is happening. So:
time = rnd(500) + 300. The + 300 will ensure that our time is always at least 300 milliseconds long by adding 300 to whatever value that is randomly generated.... And guess what, that is an

example of why setting that value is an important consideration, because we don't want that value to be zero because that is not useful to our program. Boy I learn something new all the time ya know that!
By the way in case you haven't figured it out yet 1 sec. is 1000 milliseconds.

Now with that out of the way we can talk about what this here article is all about, **Select / case**.
I have thought a long time how best to explain **Select / case,** so being the semi illiterate guy that I am (yes, I did graduate, thank you!) I looked those words up in the Dictionary and came up with this; we are going to **select** or *choose from a group a particular occurrence or* case of the variable random. Hey we're making decisions again!!
We're testing or selecting the condition of the variable random to see which state it is in and then executing the instructions for that condition, and that's all there is to it.
As each random number comes up the program looks at the value, and if it is say = to 1then it stops there and moves to the designated location sets the **Color, Prints** the message
then **Sleeps (n)** for the amount of time provided by the variable time. Using **Sleep** is as simple as providing a number between the parentheses, then the instructions stop for that amount of time then returns control to the program. Easy huh?

Now of course all things got to end so after setting up our **select / case** we need to **endselect** so our program will know when to go back and look at the next condition of random.
We do that **until inkey$** is not equal to nothing (a key press) and that ends our program.
As it runs you'll be able to see each time it **selects** each **case** and executes that part and the time that it pauses before resuming.

That's all there is to it folks! And I hope it helps you out understanding how to use **select / case** in your programs. And remember if you're programming and not having fun then your not using **IBasic.**

See ya next month!!!!

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

JavaScript Jukebox

By

Paul “pel” Love

I was working on a project recently and it occurred to me that I could use a freeware javascript routine I had lying around to let the user do some loan payment calculations (using IBasic’s embedded browser window) without having to write any code myself. That, in turn, got me thinking about how many free javascripts are available on the net and how easily they can be integrated into IBasic programs. The program "Javascript Jukebox" is a quick example of making use of javascript routines; it also includes an ’editor’ screen to let you modify the javascript code or copy pieces of it to use in a new script.

Basically, in order to run each javascript in the browser control, you can start with a web page "skeleton" and place the appropriate parts of the script in the head or body sections (most scripts indicate clearly what goes where). For example, start with the basic HTML page:

```
<html>
<head>
<title>Basic Web Page</title>
</head>
<body>
</body>
</html>
```

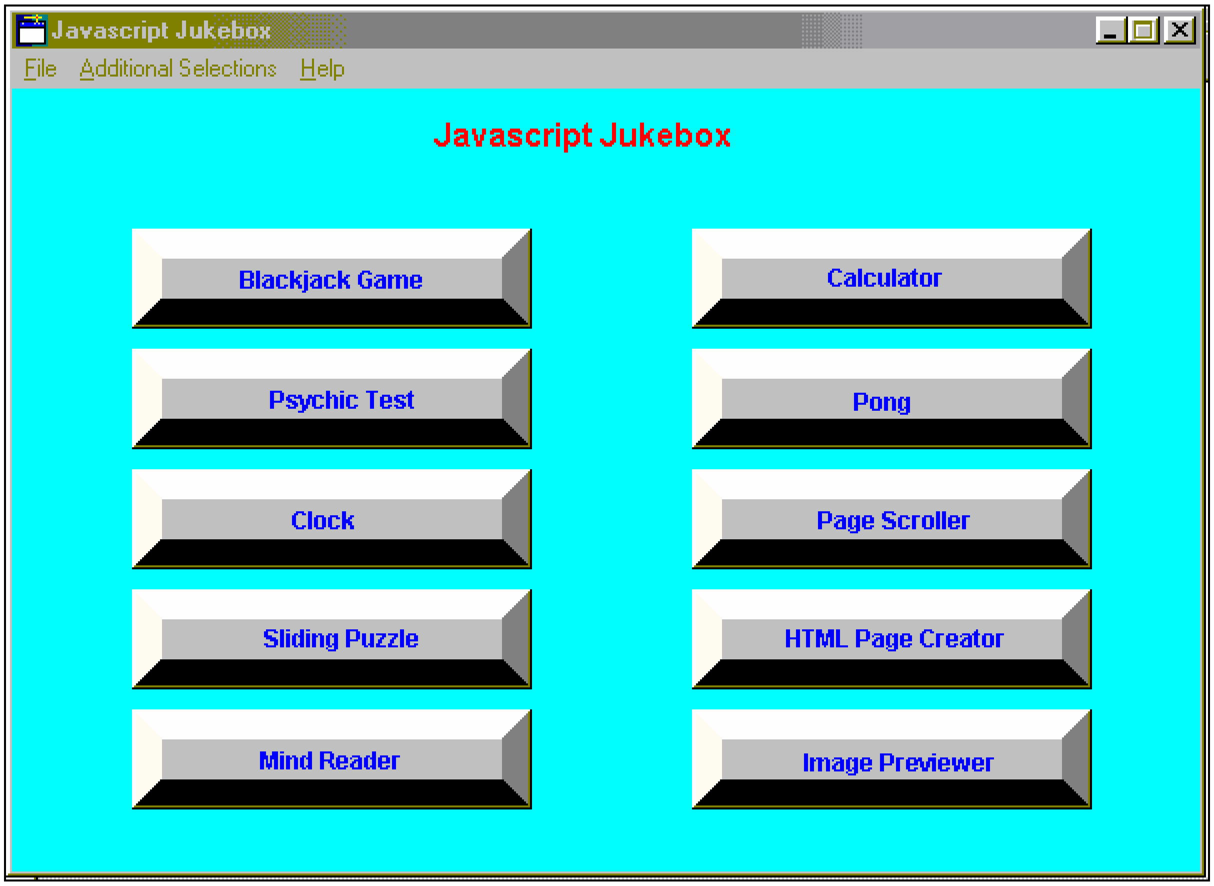
Then insert the "head" portion of the script between the <head> and </head> tags and the "body" portion between the <body> and </body> tags.

Of course you can also make use of VBScript, DHTML routines and Java applets -- for example, there’s a freeware applet named "ADraw" available from http://www.javaside.com/ that’s a neat "Paint" type program and can be easily called from a web page with about four lines of HMTL code.

Getting back to Javascript Jukebox though, here’s a picture of the main screen:

IBasic Monthly
Dedicated to everything IBasic!

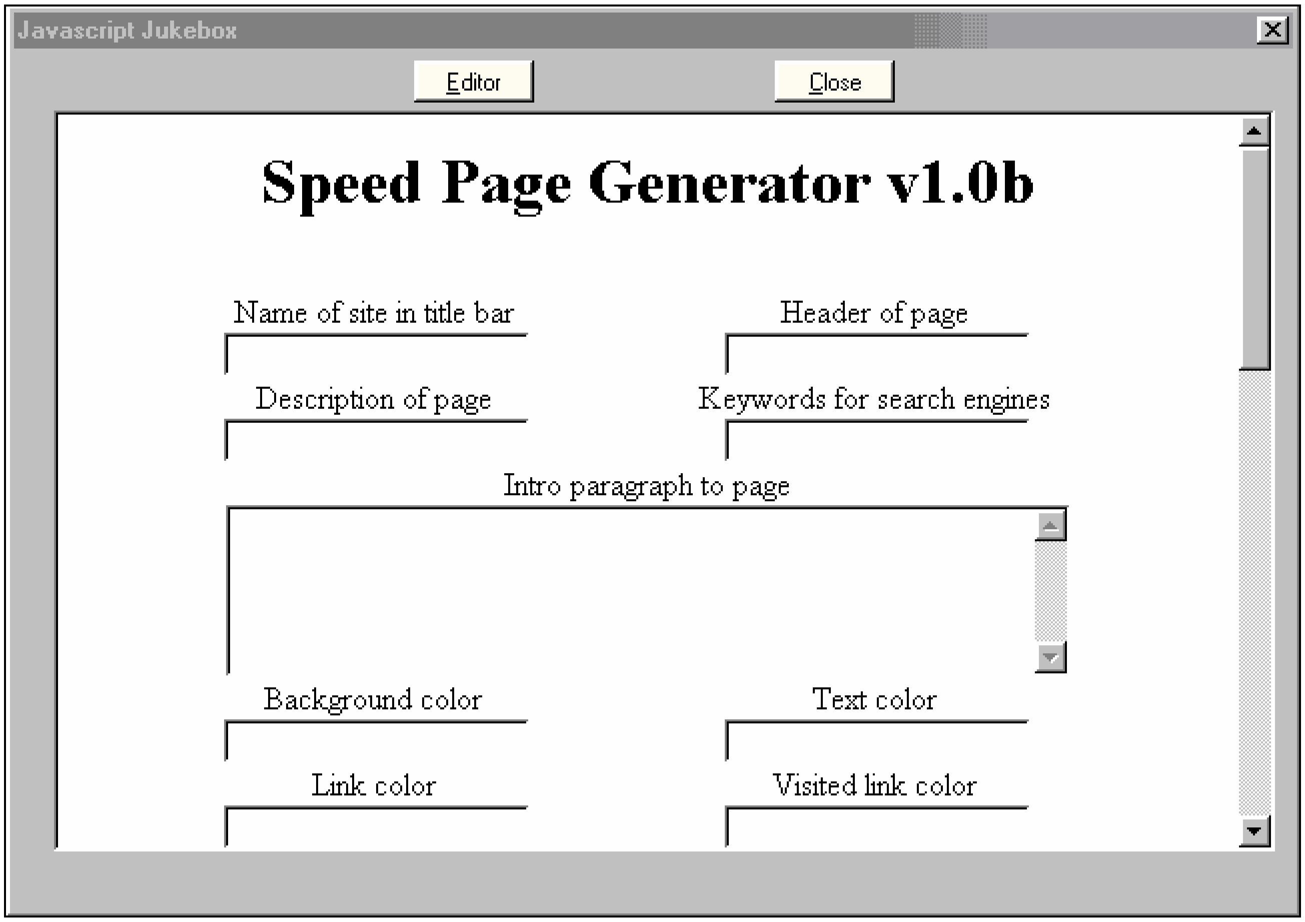
December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2



Clicking on any of the buttons starts that particular script running in a browser window:

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

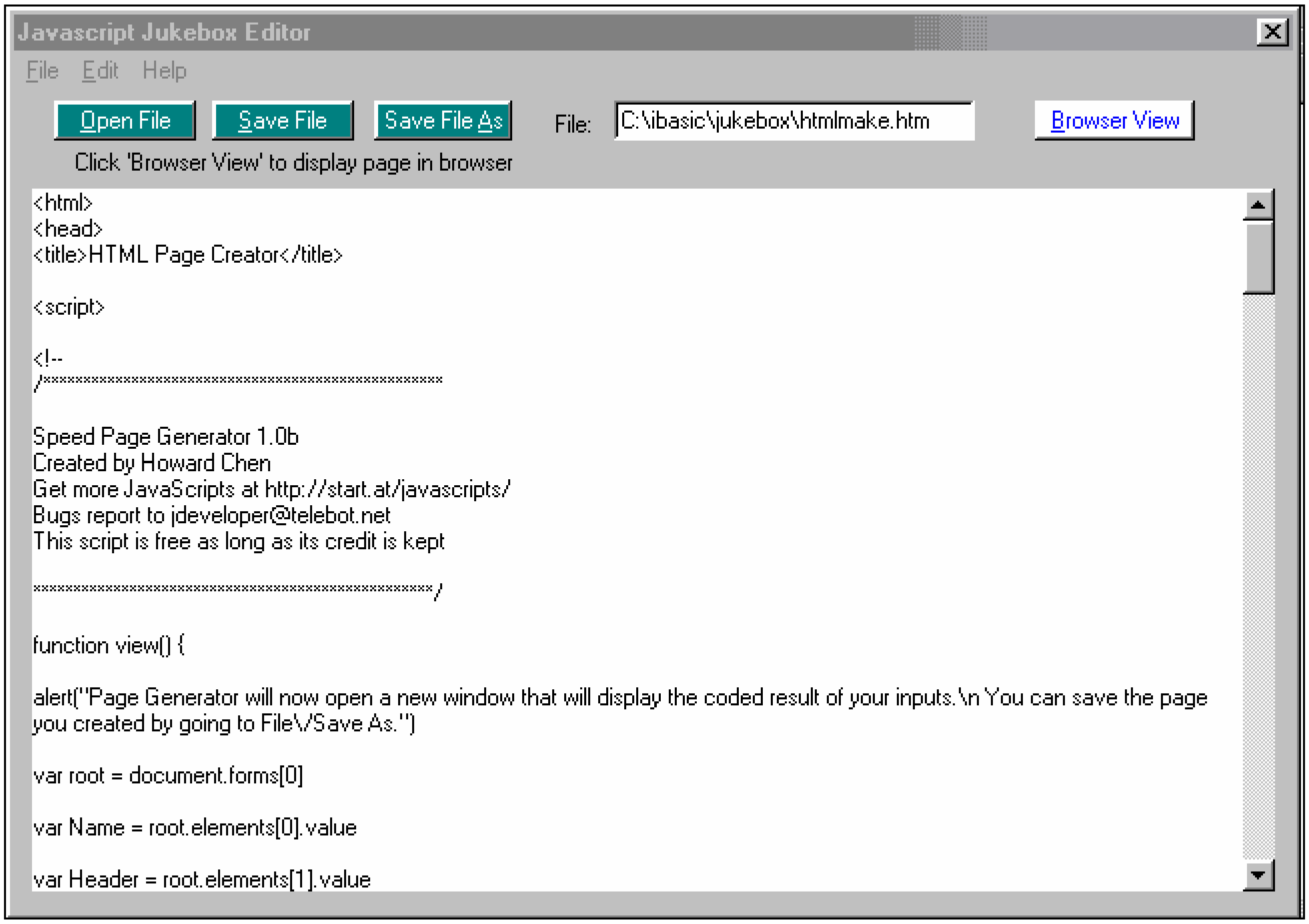


Speed Page Generator v1.0b

Name of site in title bar
Header of page
Description of page
Keywords for search engines
Intro paragraph to page
Background color
Text color
Link color
Visited link color

And clicking on the 'Editor' button over the browser window copies the web page code into an edit screen:

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2


Javascript Jukebox Editor
File Edit Help
Open File
Save File
Save File As
File:
C:\ibasic\jukebox\htmlmake.htm
Browser View
Click 'Browser View' to display page in browser

```
<html>
<head>
<title>HTML Page Creator</title>

<script>

<!--
/*****************************************************

Speed Page Generator 1.0b
Created by Howard Chen
Get more JavaScripts at http://start.at/javascripts/
Bugs report to jdeveloper@telebot.net
This script is free as long as its credit is kept

*****************************************************/

function view() {

alert("Page Generator will now open a new window that will display the coded result of your inputs.\n You can save the page
you created by going to File\/Save As.")

var root = document.forms[0]

var Name = root.elements[0].value

var Header = root.elements[1].value
```

And finally, here’s the IBasic program listing:

```
REM Javascript Jukebox

'Trap keyboard and mouse events
SETID "ENMKEYEVENTS",0x10000
SETID "ENMOUSEEVENTS",0x20000
SETID "ENMSGFILTER",0x700
TYPE MSGFILTER
  def hwndFrom:INT
  def idFrom:INT
  def code:INT
  def msg:INT
  def wparam:INT
  def lparam:INT
ENDTYPE
DEF mf:MSGFILTER
DEF mem:MEMORY
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
DEF win,wb,wb1:WINDOW
DEF dlg,dlg2:DIALOG
DEF run,x,y,w,h,IEflag,answer,answer2,bitmap1,pos,rtflag:INT
DEF htmlfile$,null$,text$:STRING
DEF savetext$[64000],savetext2$[64000]:ISTRING
DEF htmlfile:FILE

window win,0,0,600,435,@CAPTION|@SYSMENU|@MINBOX,0,"Javascript
Jukebox",winhandler
setwindowcolor win,RGB(0,180,255)
menu win,"T,&File,0,0","I,&Quit,0,3"
insertmenu win,1,"T,&Additional Selections,0,0","I,Graph maker,0,21"
insertmenu win,2,"T,&Help,0,0","I,Contents,0,91"
CONTROL win,"B,,60,70,200,50,@CTLBTNBITMAP,100"
CONTROL win,"B,,60,130,200,50,@CTLBTNBITMAP,120"
CONTROL win,"B,,60,190,200,50,@CTLBTNBITMAP,140"
CONTROL win,"B,,60,250,200,50,@CTLBTNBITMAP,160"
CONTROL win,"B,,60,310,200,50,@CTLBTNBITMAP,180"
CONTROL win,"B,,340,70,200,50,@CTLBTNBITMAP,200"
CONTROL win,"B,,340,130,200,50,@CTLBTNBITMAP,220"
CONTROL win,"B,,340,190,200,50,@CTLBTNBITMAP,240"
CONTROL win,"B,,340,250,200,50,@CTLBTNBITMAP,260"
CONTROL win,"B,,340,310,200,50,@CTLBTNBITMAP,280"
SETCONTROLTEXT win,100,GETSTARTPATH+"blackjck.bmp"
SETCONTROLTEXT win,120,GETSTARTPATH+"psytest.bmp"
SETCONTROLTEXT win,140,GETSTARTPATH+"clock.bmp"
SETCONTROLTEXT win,160,GETSTARTPATH+"puzzle.bmp"
SETCONTROLTEXT win,180,GETSTARTPATH+"mreader.bmp"
SETCONTROLTEXT win,200,GETSTARTPATH+"calc.bmp"
SETCONTROLTEXT win,220,GETSTARTPATH+"pong.bmp"
SETCONTROLTEXT win,240,GETSTARTPATH+"pgscroll.bmp"
SETCONTROLTEXT win,260,GETSTARTPATH+"htmlmake.bmp"
SETCONTROLTEXT win,280,GETSTARTPATH+"imagepre.bmp"

DIALOG dlg,0,0,640,430,0x80C00080|@SYSMENU|@SIZE,0,"Javascript
Jukebox",dlghandler
CONTROL dlg,"B,&Editor,200,5,60,22,@TABSTOP,10"
CONTROL dlg,"B,&Close,380,5,60,22,@TABSTOP,20"

DIALOG dlg2,0,0,640,430,0x80C00080|@SYSMENU|@SIZE,0,"Javascript Jukebox
Editor",dlg2handler
CONTROL
dlg2,"RE,,10,50,620,350,@TABSTOP|@CTEDITMULTI|@CTEDITRETURN|@VSCROLL|@HSCROLL,
10"
CONTROL dlg2,"T,Click 'Browser View' to display page in
browser,30,30,250,18,,15"
CONTROL dlg2,"E,,300,5,180,20,@TABSTOP|@CTEDITAUTOH,20"
CONTROL dlg2,"T,File:,270,10,30,20,0x5000010B,30"
CONTROL dlg2,"B,&Browser View,510,5,80,20,@TABSTOP|@CTLBTNDEFAULT,70"
CONTROL dlg2,"B,&Open File,20,5,70,20,@TABSTOP,100"
CONTROL dlg2,"B,&Save File,100,5,70,20,@TABSTOP,110"
CONTROL dlg2,"B,Save File &As,180,5,70,20,@TABSTOP,120"

bitmap1 = LoadImage (GETSTARTPATH+"jukebox.bmp",@IMGBITMAP)
SHOWIMAGE win,bitmap1,@IMGBITMAP,40,5,500,50
```

```
IEflag = 1: ' assume IE4+ is installed
null$ = ""

run = 1
WAITUNTIL run = 0
deleteimage bitmap1,@IMGBITMAP
CLOSEWINDOW win
END

'------------------------------------------------------------------------------
SUB winhandler
SELECT @CLASS
      CASE @IDCREATE
       centerwindow win

    CASE @IDCLOSEWINDOW
     run = 0

    CASE @IDMENUPICK
         select @MENUNUM
            case 3
               run = 0
            case 21: ' graphit
               htmlfile$ = GETSTARTPATH + "graphit1.htm"
               showwindow win,@SWHIDE
               answer = domodal dlg
               if wb > 0 then closewindow wb
               showwindow win,@SWRESTORE

            case 91: ' help file
               htmlfile$ = GETSTARTPATH + "jboxhelp.htm"
               showwindow win,@SWHIDE
               answer = domodal dlg
               if wb > 0 then closewindow wb
               showwindow win,@SWRESTORE
         endselect

    CASE @IDCONTROL
         select @CONTROLID
            case 100: ' blackjack game
               htmlfile$ = GETSTARTPATH + "blackjck.htm"
               showwindow win,@SWHIDE
               answer = domodal dlg
               if wb > 0 then closewindow wb
               showwindow win,@SWRESTORE

            case 120: ' psychic test
               htmlfile$ = GETSTARTPATH + "psytest.htm"
               showwindow win,@SWHIDE
               answer = domodal dlg
               if wb > 0 then closewindow wb
               showwindow win,@SWRESTORE

            case 140: ' clock
               htmlfile$ = GETSTARTPATH + "clock.htm"
```

```
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 160: ' sliding puzzle
          htmlfile$ = GETSTARTPATH + "puzzle.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 180: ' mind reader
          htmlfile$ = GETSTARTPATH + "mreader.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 200: ' calculator
          htmlfile$ = GETSTARTPATH + "calc.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 220: ' pong
          htmlfile$ = GETSTARTPATH + "pong.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 240: ' page scroller
          htmlfile$ = GETSTARTPATH + "pgscroll.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 260: ' HTML page creator
          htmlfile$ = GETSTARTPATH + "htmlmake.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

       case 280: '
          htmlfile$ = GETSTARTPATH + "imagepre.htm"
          showwindow win,@SWHIDE
          answer = domodal dlg
          if wb > 0 then closewindow wb
          showwindow win,@SWRESTORE

   endselect
```

```
ENDSELECT
RETURN

sub dlghandler
SELECT @CLASS
      CASE @IDCLOSEWINDOW
       closedialog dlg,@IDOK

    CASE @IDCONTROL
            select @CONTROLID
           case 10:  ' editor
              answer2 = domodal dlg2
           case 20:  ' close
              closedialog dlg,@IDOK
        endselect

      'size the embedded browser when the dialog is resized
      CASE @IDSIZE
          GETSIZE(dlg,x,y,w,h)
            IF (wb <> 0)
            x = 20:  y = 30:  w = w - 30:  h = h - 60
                  SETSIZE(wb,x,y,w,h)
            ENDIF

    CASE @IDINITDIALOG
        setcontrolcolor dlg,10,RGB(0,0,0),RGB(255,255,208)
        setcontrolcolor dlg,20,RGB(0,0,0),RGB(255,255,208)
        if IEflag = 0
           messagebox dlg,"ABORT - Internet Explorer 4.0+ not
installed","Javascript Jukebox"
        else
           'err = SetCurrentDirectoryA(GETSTARTPATH)
           WINDOW
wb,20,30,610,370,@BROWSER|@NOAUTODRAW|@NOCAPTION,dlg,"Jukebox Browser",wbmain
           if(openfile(htmlfile,htmlfile$,"R") = 0)
              BROWSECMD wb,@NAVIGATE,htmlfile$
              closefile htmlfile
           endif
        endif
ENDSELECT
RETURN

wbmain:
' browser window handler
SELECT @CLASS
      CASE @IDCLOSEWINDOW
            'run = 0


ENDSELECT
RETURN

wb1main:
' main browser window handler
SELECT @CLASS
      CASE @IDCLOSEWINDOW
```

```
            'run = 0

ENDSELECT
RETURN

sub dlg2handler
SELECT @CLASS
      CASE @IDCLOSEWINDOW
       closedialog dlg2,@IDOK

    case @IDMENUPICK
         select @MENUNUM
            case 1: 'Close
               closedialog dlg2,@IDOK
            case 2: 'New file
               ret = CONTROLCMD (dlg2,10,@RTLOAD,null$,0)
            case 3: 'Print
               CONTROLCMD dlg2,10,@RTPRINT
            case 21:'Select All
               setfocus dlg2,70
               setfocus dlg2,10
               CONTROLCMD dlg2,10,@RTSETSELECTION,0,-1
                case 22:'Copy
                     CONTROLCMD dlg2,10,@RTCOPY
                case 23:'Paste
                     CONTROLCMD dlg2,10,@RTPASTE
            case 41:  ' load basic template
               htmlfile$ = GETSTARTPATH+"base.htm"
               gosub doopen
            case 71: 'help file

         endselect

    CASE @IDCONTROL
            select @CONTROLID
           case 10: ' RE control
               if @NOTIFYCODE = @ENMSGFILTER
                  mem = @QUAL
                  READMEM mem,1,mf
                  select mf.msg
                     case @IDRBUTTONUP
                        mx = mf.lparam&0xffff
                        my = mf.lparam/0x10000
                        CONTEXTMENU dlg2,mx+10,my+50,"I,Select
All,0,21","I,Copy,0,22","I,Paste,0,23"
                  endselect
               endif

           case 70: ' go
              closedialog dlg2,@IDOK

           case 100: ' open file
              htmlfile$ = "":  gosub doopen
           case 110: ' save to file
              gosub dosave
           case 120: ' save file as
```

```
              htmlfile$ = "":  gosub dosave
        endselect

    CASE @IDINITDIALOG
        menu dlg2,"T,&File,0,0","I,New File,0,2","I,Print,0,3","I,Close,0,1"
        insertmenu dlg2,1,"T,&Edit,0,0","I,Select
All,0,21","I,Copy,0,22","I,Paste,0,23","I,Find,0,25"
        insertmenu dlg2,2,"T,Help,0,0","I,Content,0,71"
        SETCONTROLCOLOR dlg2,100,RGB(255,255,255),RGB(0,140,152)
        SETCONTROLCOLOR dlg2,110,RGB(255,255,255),RGB(0,140,152)
        SETCONTROLCOLOR dlg2,120,RGB(255,255,255),RGB(0,140,152)
        setcontrolcolor dlg2,70,RGB(0,0,255),RGB(255,255,255)
        CONTROLCMD dlg2,10,@RTSETLIMITTEXT,512000
        CONTROLCMD dlg2,10,@RTSETEVENTMASK,@ENMKEYEVENTS
        CONTROLCMD dlg2,10,@RTSETEVENTMASK,@ENMMOUSEEVENTS

        if (htmlfile$ <> "") & (len(htmlfile$)>0) then gosub doopen

ENDSELECT
RETURN

SUB doopen
      if htmlfile$ = "" then htmlfile$ = filerequest("Load File",dlg2,1)
      if(openfile(htmlfile,htmlfile$,"R") = 0)
        ret = CONTROLCMD (dlg2,10,@RTLOAD,htmlfile,0)
        setfocus dlg2,10
        CONTROLCMD dlg2,10,@RTHIDESEL,1
        if ret = 0 then rtflag = 1
        closefile htmlfile
    endif
    setcontroltext dlg2,20,htmlfile$
RETURN

SUB dosave
    CONTROLCMD dlg2,10,@RTSETSELECTION,0,-1
    savetext$ = CONTROLCMD(dlg2,10,@RTGETSELTEXT)
    do
      pos = instr(savetext$,chr$(13))
          if pos <> 0
             if mid$(savetext$,pos,1) = chr$(13)
                savetext2$ = mid$(savetext$,1,pos-1)
                savetext2$ = savetext2$ + mid$(savetext$,pos+1)
                savetext$ = savetext2$
             endif
          endif
       until pos = 0
    if (CONTROLCMD (dlg2,10,@RTLOAD,savetext$,0)) = 0
    endif

      if htmlfile$ = "" then htmlfile$ = filerequest("Save File",dlg2,0)
      if(len(htmlfile$) > 0)
            if(openfile(htmlfile,htmlfile$,"W") = 0)
                  ret = CONTROLCMD (dlg2,10,@RTSAVE,htmlfile,0)
                  closefile htmlfile
            endif
      endif
```

```
RETURN
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

ibHash: Faking Associative Arrays With IStrings

By

Jerry Muelver, HyText Consulting

An array of strings is a powerful tool, the "Programmer's Friend" for sure. You can store, retrieve, add, delete, modify, sort, slice, and dice data every which way once you've got it into an array.

Numerical Index: Array

To get a hold of your data, all you have to know is the numerical index for the array element holding the data:

```
def fruit[5]:string
fruit = "apple","banana","cherry","mango","orange"
```

Now we know that the array looks like this:

```
fruit[0] = "apple"
fruit[1] = "banana"
fruit[2] = "cherry"
fruit[3] = "mango"
fruit[4] = "orange"
```

So if someone asks, "Do you like fruit[2] pie?", we can answer, "Yes, fruit[2] pie is good, but I really like fruit[0] pie!" and everyone will know what we're talking about... as long as everyone knows what numerical index stands for which fruit... and no one added any that we don't know about... and the array was originally defined big enough to hold the total number of kinds of fruit we ever want to list... and no one inserts a new fruit near the beginning of the list and changes all the index numbers we used to know and love....

So, there are some organizational and structural problems with numerically-indexed arrays. We typically overcome those problems by extensive index bookkeeping to keep track of what's where in the array, so we know how to find data when we need it.

Suppose I have an array of my "favorites", like this:

```
favorites = "burgandy","apple","chocolate almond","banana","linguini"
```

Well, that's nice and tidy. Now quick -- what's my favorite pasta? That's not so hard to figure out. How about my favorite fruit? Pie? Color? Would "chocolate almond" be my favorite candy bar... or ice cream? The numerical index for this array is not going to be much help answering those questions, because you have to know not only what item is at each index, but also what kind of item it is.

String Index: Hash

Let me introduce you to the associative array, also called a "hash". Here's another way of looking at my favorites:

```
favorite["color"]     = "burgandy"
favorite["pie"]       = "apple"
favorite["ice cream"] = "chocolate almond"
favorite["fruit"]     = "banana"
favorite["pasta"]     = "linguini"
```

Suddenly, everything's clear! Instead of numbers for the index, I used strings. For each string I associated a value -- another string. Now, quick -- what's my favorite fruit? Pie? Color?

Elements in an associative array have two components -- the string index ("key"), and the associated content ("value").

An associative array is also called a "hash" because the key (in languages like Perl, Java, Python, awk, JavaScript, Ruby, Smalltalk, etc.) is processed into a memory address and stored in an internal database that does all the bookkeeping for you. Ask for favorite["pie"] and "pie" gets hashed into an address, and the program coughs up the contents of that address through direct access -- no searching or reading or comparing or other fiddling around.

With very little effort, you can come up with scads of applications for hashes -- username and password, name and phone number, style and color, paragraph and font, room and contents, disease and symptoms, action and life-credits -- the list is endless. There's only one small problem -- IBasic does not support hashes.

Arrays as Workaround Hashes

There are several options for doing hash-like work in IBasic. One way is to create an array of User-Defined Types (UDT):

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
Type hash
    key: string
    value: string
Endtype

def favorite[5]: hash
favorite[0].key = "color"
favorite[0].value = "burgundy"
favorite[1].key = "pie"
...
```

That's a little better. It still has a numerical index to juggle around, but at least we can search for a particular key and get that key's value, and we could have numbers in their native form instead of strings for the value part of the UDT if needed.

You could also skip the UDT and go directly to an array structure like this:

```
def favorite[10]:string
favorite = "color","burgundy","pie","apple","ice cream","chocolate
almond"
favorite[3] = "fruit","bananna","pasta","linguini"
```

Then, to find the value for a particular key, search the odd-number indexes:

```
for i = 0 to 9 step 2
    if (favorite[i] = key) then found = i
next i
```

or (my preference)

```
numelements = 10
i = -2:
do
    i = i + 2
until ((i > numelements) | (favorites[i] = key))
```

To change the value for that key, put the new value into favorite[i+1]. To delete a key, move each of the following array keys and values up two steps to fill the preceding key and value with the current key and value.... And don't forget to subtract two from the total number of array elements to keep track of the number of keys plus the number of values!

But we're still juggling numerical array indexes to get the job done. Isn't there a better way?

ibHash to the Rescue!

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Yes! There ***is*** a better way -- the elusive, mysterious, hitherto-unknown but soon-to-be-famous ibHash! An ibHash looks like this:

```
favorite[2000]:istring
favorite = "{color}burgundy{pie}apple{ice cream}chocolate almond"
favorite = favorite + "{fruit}banana{pasta}linguini"
```

The rules for building an ibHash are simple:

1. Define an ISTRING big enough to hold all your keys and values.
2. Put each key into curly braces.
3. Put the value for a key right after the key in the ISTRING.

That's pretty straightforward. But how do we find a key in that ISTRING? Or add a key and value? Change a value? Delete something? The companion demo file shows the gritty details of ibHash manipulation with ibHash subroutines (which call some included string-manipulation subroutines for execution). Let me explain the general ideas here.

- **Find a key** -- To find a key in an ibHash, put the key into curly braces, then search the ibHash with INSTR(ibHash,key) -- ibHashGet(ibHash,key), uses extractstr(str,openmarker,closemarker)
- **Add a key and value** -- To add something to an ibHash, put the key in curly braces, add the value to the end, and add the combination to the ibHash -- ibHashPut(ibHash,key,value), uses extractstr(str,openmarker,closemarker) and replstr(str,target,source)
- **Delete a key and value** -- to delete a key-value pair from the ibHash, find the key, copy everything from the key to the next opening curly brace or end of file to get the key's value, and replace the key + value combination in the ibHash with a null -- ibHashDel(ibHash,key), uses extractstr(str,openmarker,closemarker) and replstr(str,target,source)
- **Change the value for a key** -- To change a key's value, find the key, copy the value, and replace the old key + value combination in the ibHash with the new key + value combination -- ibHashPut(ibHash,key,value), if the key is already in the ibHash also uses extractstr(str,openmarker,closemarker) and replstr(str,target,source) to update the existing key.
- **List key-value pairs** -- To list all the key-value pairs in an ibHash, chop each key-value off a copy of the ibHash with splitstr(str,marker) with marker="{", then chop the pair apart at the closing curly brace "}".

The companion demo program for this article shows how to put an ibHash together entirely from user input. You could add file-saving and -reading functions to build a database application quite easily. For extreme flexibility, you tie your ibHash together with Fletchie's DynaStrings instead of istrings and hash away to the limits of the computer's memory.

When to use ibHashes

Because it is string-based, an ibHash offers flexibility that arrays and UDTs can't match. You can change the number of fields in a record as easily as changing the number of records in a file. Since the index keys are strings, you can make them up on the fly, even from user input. Free form in structure, there are no restrictions on number or types of fields in records. to handle numbers, all you need to do is drop the string value into a val(str) function, and you're off to the calculation races.

I use ibHashes for color combinations, formatting styles, user input storage and updating, and free-form text base applications. New uses crop up with every new project. Try ibHashes the next time you need to keep track of odd-ball collections of objects, or input from forms, or free-form record sets for adventure game characters or quiz scores. You'll love 'em!

"When you've got a good hammer, everything looks like a nail."

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

LINKED LISTS Made Easy

CREATING A SINGLE LINKED LIST - PART II

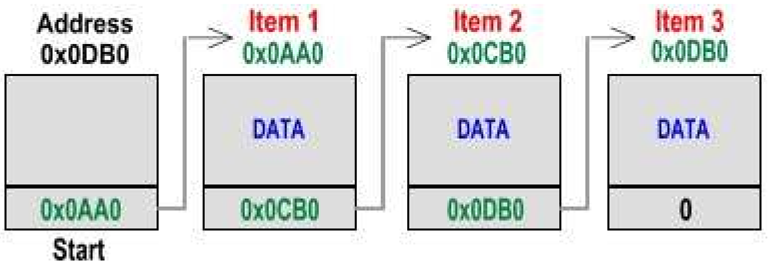
By

Bizzy

In Part I - The Foundations, we examined the many different types of variables, such as, Integer, Float, Double, String, UDT (User Defined Type), and Arrays, and how they work in RAM. We also saw how Arrays were built and UDT's made into Arrays. Pointers were explained in simple terms to enable us to understand how they work in general. And finally a Single Linked List and its Rules were examined.

In this article - Creating a Single Linked List - Part II - we will now design and build the software to do just that - build a Single Linked List.

The complete software code is listed at the end of this article.



THE FOUR STEPS TO BUILDING THE SINGLE LINKED LIST

STEPS 1, 2 and 3 CONSIST OF THE DEFINITION AND SETUP CODE WHICH IS DONE ONCE AT THE START OF THE PROGRAM

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

STEP 1

DEFINE THE USER DEFINE TYPE (UDT)

The first thing to do is to define the UDT that we will use to save our Data into. We gave it a TYPE name of **list** for this UDT definition.

FName is where we will save the First Name of a person. **LName** will have their Last Name. **Age** will contain their Age and **AreaCode** will contain the Area Code they live in. **UniqueID** will be a number which we must supply for each **Item**. It must be a different number for each **Item** we put into the **Linked List**. We will use it as an identifier later in the program.
Nxt:POINTER is where we keep the ADDRESS of the next **Item** in the **Linked List**.

EACH ITEM IN OUR LINKED LIST WILL USE THIS list UDT TO STORE THE DATA

```
TYPE list
   DEF FName:STRING
   DEF LName:STRING
   DEF Age:INT
   DEF AreaCode:INT
   DEF UniqueID:INT
   DEF Nxt:POINTER
ENDTYPE
```

STEP 2

DEFINE THE VARIABLES TO USE IN THE LINKED LIST

The variable **start** is defined as a **TYPE** of **list**. We will use **start** to hold the address of the first **Item** in our **Linked List**. This address will be assigned to **start.Nxt**.

The **node** variable is defined as a **POINTER** which we will use to add our **Items** into the **Linked List**.

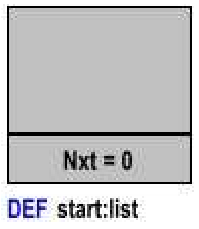
VARIABLES USED IN BUILDING OUR LINKED LIST

```
DEF start:list
DEF node:POINTER
```

STEP 3

ASSIGN start.Nxt A NULL VALUE

When we start out building a **Linked List** we need to signify that it is an EMPTY LIST by assigning start.Nxt a **0** (zero) value.



START WITH AN EMPTY LIST

```
start.Nxt = 0
```

THE FOURTH STEP IS REPEATED EACH TIME YOU ADD A NEW ITEM TO THE LINKED LIST

PS - There are two ways to put an Item into a Linked List - (1) Append - which places the new Item at the end of the List; (2) Insert - which places the new Item at the beginning of the List. The method shown here is the Append mode, though the code is in the program to do the Insert Mode.

STEP 4 (Append Mode)

(a) FIND END OF LINKED LIST

The Program is to go through a **WHILE** loop examining each **#node.Nxt** to see if it has an ADDRESS which points to the next **Item** or if it has a **0** (zero) which of course indicates it has reached the last **Item** in the List.
First set **node** to point to the **start** Address - **node** = **start**.
Now **node** and **start** can access the same address. ***PART I has more info on POINTERS if you wish to look it up.***

```
node = start
WHILE (#node.Nxt <> 0)
    node = #node.Nxt
ENDWHILE
```

(b) ALLOCATE RAM FOR THE NEW ITEM

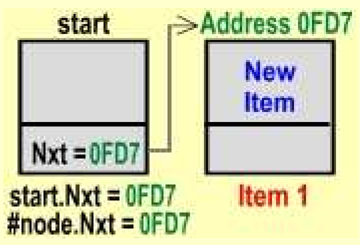
The program has exited the **WHILE** loop because it has found

```
#node.Nxt = NEW(list, 1)
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

#node.Nxt = 0 and is in position to add the next **Item**.
So we need to allocate RAM for the new **Item** and save the new **Item's** address into **#node.Nxt** which now equals **0.**
NEW allocates RAM for the TYPE list and returns the Address to #node.Nxt



(c) ASSIGN NODE THE VALUE OF #NODE.NXT

Now that we have created the new **Item** and have its address in **#node.Nxt** we need to put data into it.
To put data into the new **Item** we need to assign the **Pointer node** the address of the new Item. **node = #node.Nxt**
The variable **Pointer node** is now pointing to the new **Item** we just allocated RAM for.

node = #node.Nxt

node is now pointing to address 0FD7 (see Image Above)

(d) ASSIGN VALUES TO NODE VARIABLES

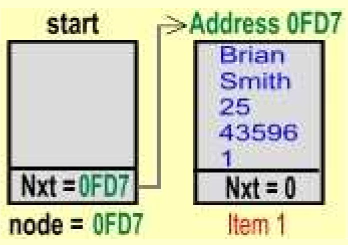
All the Variables are now assigned their values. The program we are using has a **SUB** which sends the values to the **Linked List**. ***(See Code at bottom of article).***
Note the dereferencing # sign is used to put Data into the node.

#node.FName = "Brian"
#node.LName = "Smith"
#node.Age = 25
#node.AreaCode = 43596
#node.UniqueID = 1

(e) ASSIGN #NODE.NXT = 0

The current **node** is the last **Item** in the **Linked List** so **#node.Nxt** is assigned **0**. The next **Item** to be added will read through the **Linked List** until it finds this **#node.Nxt = 0 (our First Item)** before it adds the next **Item** to the **Linked List**.
The diagram, at right, shows the values inserted into the **Item** in the **Linked List**.
The new **Item** just added is in position one of the **Linked List** and the next **Item** will follow in position two in the **Linked List**.

#node.Nxt = 0



IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

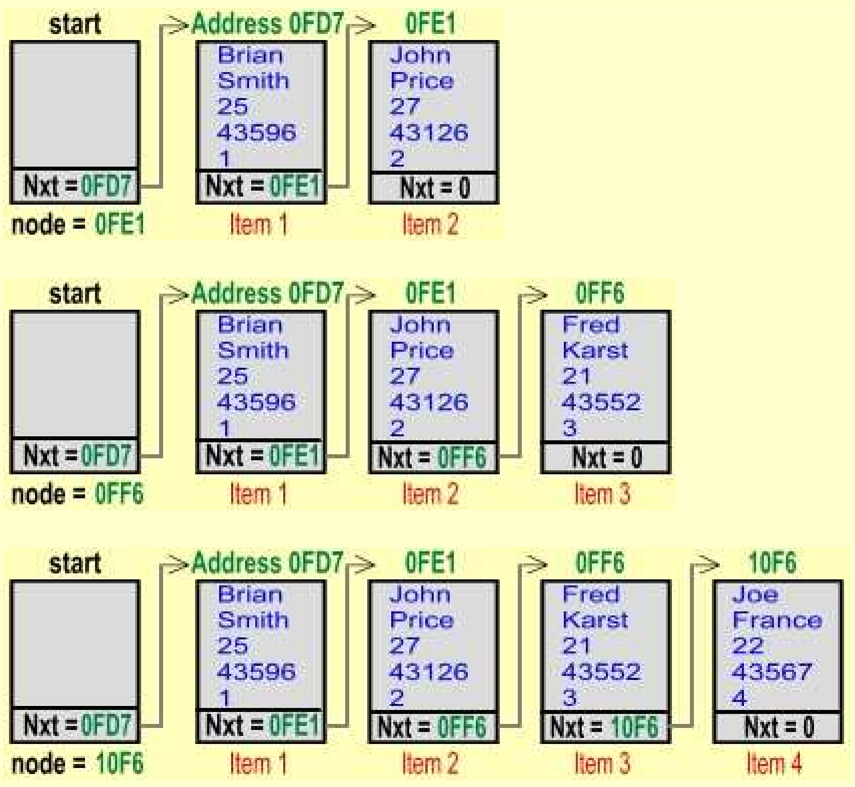
That is the First Item in the Linked List entered.

To enter more Items into the Linked List go to STEP 4 and do (a), (b), (c), (d) and (e) again.

<u>NOTE:</u> There is an easy way to know when to use the dereferencing sign #. It is used whenever you wish to put Data in or get Data out of an Item. No matter whether it is an address, a string or an integer. These Data variables were declared <u>in</u> the UDT.

PUTTING IT ALL TOGETHER

(a)	**node = start** **WHILE** **(#node.Nxt <> 0)** **node = #node.Nxt** **ENDWHILE**
(b)	**#node.Nxt = NEW(list, 1)**
(c)	**node = #node.Nxt**
(d)	**#node.FName = "John"** **#node.LName = "Price"** **#node.Age = 27** **#node.AreaCode = 43126** **#node.UniqueID = 2**
(e)	**#node.Nxt = 0**



IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

STEP 4 - USING THE INSERT MODE

```
DEF s:INT

s = GETSTATE(Main,8)
IF s = 1
  node = NEW(list,1)
  #node.FName = FN
  #node.LName = LN
  #node.Age = Age
  #node.AreaCode = AC
  #node.UniqueID = ID
  #node.Nxt = start.Nxt
  start.nxt = node
ELSE
 ...
 ...
ENDIF
```

FOUR STEPS TO INSERT AN ITEM AT BEGINNING OF LINKED LIST

The **Insert Mode** adds the **Item** at the beginning of the **Linked List** each time a new **Item** is added.

(a) Create a new **Item** with the NEW command and let the Address be stored in the **node**Pointer.
(b) Now assign the **Item** all of its Data
(c) Assign the **#node.Nxt** the value in **start.Nxt**
(d) Assign the **start.Nxt** variable the Address in **node**

To put it simply the **new node** will always be at the start of the **Linked List** and will always take the **Address from start.nxt***(where the last Item was Inserted)* **into its #node.Nxt**. Also **start.Nxt** receives the **New node Address** created from the NEW command.

PRINTING THE LINKED LIST

```
node = start.Nxt
WHILE node
   ADDSTRING(Main, 5, APPEND$(#node.FName," ",#node.LName,", Age
",STR$(#node.Age),
                                      ", Area Code ",STR$(#node.AreaCode)))
   node = #node.Nxt
ENDWHILE
```

The Linked List can be printed by iterating through the List with a WHILE loop.

1. So that we commence at the beginning of the list we assign the address of the **First Item** in **start.Nxt** to the **node** pointer.
node = start.Nxt

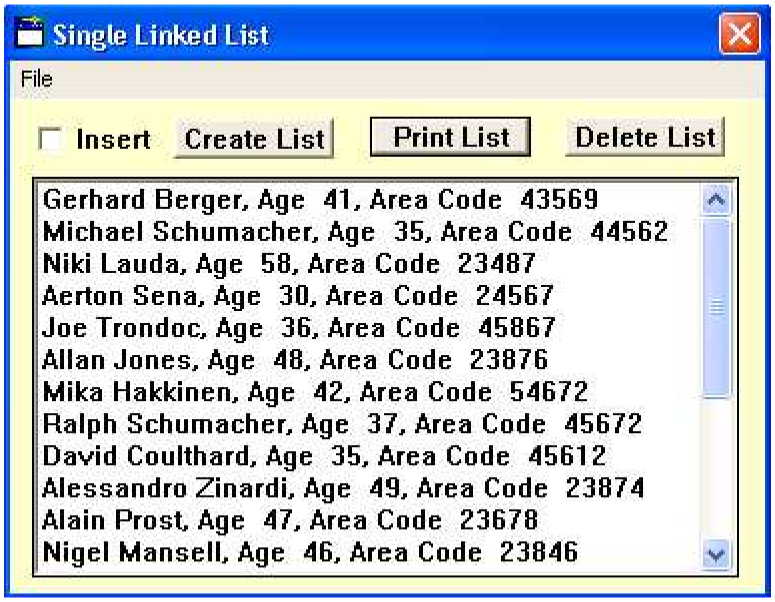
2. A **WHILE** loop is used to iterate through the **Linked List**.
WHILE node

3. ADDSTRING function is used to collect data from each **node** and put the data into a **Listbox**.

4. After the data is retrieved from the **node** we then assign the address of the next **node** in the **Linked List** to **node**. This gets the next **Item's** data into the **node** pointer. **node = #node.Nxt**

5. The **WHILE** loop is ended with **ENDWHILE**

THE LINKED LIST PRINTED TO A LISTBOX



```
Gerhard Berger, Age  41, Area Code  43569
Michael Schumacher, Age  35, Area Code  44562
Niki Lauda, Age  58, Area Code  23487
Aerton Sena, Age  30, Area Code  24567
Joe Trondoc, Age  36, Area Code  45867
Allan Jones, Age  48, Area Code  23876
Mika Hakkinen, Age  42, Area Code  54672
Ralph Schumacher, Age  37, Area Code  45672
David Coulthard, Age  35, Area Code  45612
Alessandro Zinardi, Age  49, Area Code  23874
Alain Prost, Age  47, Area Code  23678
Nigel Mansell, Age  46, Area Code  23846
```

DELETING THE LINKED LIST

1. DEF ref, temp:**POINTER** - Two Pointers	**We need to start at the beginning of the**

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

are defined.

2. **ref** = **start.Nxt** - The **ref** Pointer is set to the start of the List.
3. IF(**ref**) - Test to see that the List is not EMPTY.
4. WHILE **#ref.Nxt** - Start WHILE Loop - which checks for **#ref.Nxt** containing an address. If a **0** is found then WHILE loop has reached the end of the List.
5. **temp** = **ref** - Assign the address of **ref** to **temp.** We cannot DELETE **ref** as it contains the next address, so **temp** holds the address to be deleted.
6. **ref** = **#ref.Nxt** - Get the next address into **ref**.
7. DELETE **temp** - Delete the **temp** address from the Linked List.
8. ENDWHILE - Exit the WHILE loop when **#ref.Nxt = 0**
   ENDIF - Exit the IF function
9. **start.nxt = 0** - List is empty so set **start.nxt = 0**
   node = start - Set **node** Pointer to same address as **start.** This sets variables up for empty list.
10. SENDMESSAGE Main,@LB_RESETCONTENT,0,0,5 Clears the Listbox.

Linked List and iterate through the list to delete the Item's.

```
SUB DeleteLinkedList
   DEF ref, temp:POINTER

   ref = start.Nxt

   IF(ref)
      WHILE #ref.Nxt
         temp = ref
         ref = #ref.Nxt
         DELETEtemp
      ENDWHILE
   ENDIF

   start.Nxt = 0
   node = start
   SENDMESSAGE
       Main,@LB_RESETCONTENT,0,0,5
RETURN
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Single Linked List – Append Mode

IBASIC CODE FOR THE SINGLE LINKED LIST PROGRAM

```
SETID "LB_RESETCONTENT", 388

DEF Main:WINDOW
TYPE list
 DEF FName:STRING
 DEF LName:STRING
 DEF Age:INT
 DEF AreaCode:INT
 DEF UniqueID:INT
 DEF Nxt:POINTER
ENDTYPE

DEF start:list
DEF node:POINTER
DECLARE AddList(FN:STRING,LN:STRING,Age:INT,AC:INT,ID:INT)

WINDOW Main,150,150,385,300,@CAPTION|@SYSMENU|@SIZE,0,"Single Linked
List",MainWindow
MENU Main,"T,&File,0,0","I,&Quit,0,1"
CONTROL Main,"B,Create List,84,8,90,20,0x50010000,1"
CONTROL Main,"B,Print List,175,8,90,20,0x50010000,2"
CONTROL Main,"B,Delete List,267,8,90,20,0x50010000,3"
CONTROL Main,"B,,90,1,20,21,0x40080001,40"
CONTROL Main,"L,,11,38,353,200,0x50B00140,5"
CONTROL Main,"C,Insert,13,8,70,20,0x50000003,8"

SETWINDOWCOLOR Main, RGB(255,255,200)

start.Nxt = 0

run = 1
WAITUNTIL run=0
CLOSEWINDOW Main
END

SUB MainWindow
```

```
SELECT @CLASS
 CASE @IDCLOSEWINDOW
  run = 0
 CASE @IDMENUPICK
  SELECT @MENUNUM
   CASE 1
    run = 0
  ENDSELECT
 CASE @IDCONTROL
  SELECT @CONTROLID
   CASE 1
    AddLinkedList
   CASE 2
    PrintLinkedList
   CASE 3
    DeleteLinkedList
  ENDSELECT
 CASE @IDCREATE
  ENABLETABS Main, 1
  CENTERWINDOW Main
ENDSELECT
RETURN

SUB AddLinkedList
 AddList("Gerhard","Berger",41,43569,1)
 AddList("Michael","Schumacher",35,44562,2)
 AddList("Niki","Lauda",58,23487,3)
 AddList("Aerton","Sena",30,24567,4)
 AddList("Joe","Trondoc",36,45867,5)
 AddList("Allan","Jones",48,23876,6)
 AddList("Mika","Hakkinen",42,54672,7)
 AddList("Ralph","Schumacher",37,45672,8)
 AddList("David","Coulthard",35,45612,9)
 AddList("Alessandro","Zinardi",49,23874,10)
 AddList("Alain","Prost",47,23678,11)
 AddList("Nigel","Mansell",46,23846,12)
 AddList("Michael","Andretti",40,45230,13)
 AddList("Eddie","Irvine",43,43298,14)
 AddList("Damon","Hill",45,43278,15)
 AddList("Jaques","Villeneuve",45,43278,15)
 AddList("Nelson","Piquet",41,33278,15)
 AddList("Keke","Rosberg",47,33671,16)
 AddList("Jackie","Stewart",62,38674,17)
 AddList("James","Hunt",57,33678,18)
 AddList("Jack","Brabham",66,33678,19)
RETURN


SUB PrintLinkedList
```

```
 SENDMESSAGE Main,@LB_RESETCONTENT,0,0,5
 node = start.nxt
 WHILE node
  ADDSTRING(Main, 5, APPEND$(#node.FName," ",#node.LName,", Age
",STR$(#node.Age),", Area Code ",STR$(#node.AreaCode)))
  node = #node.nxt
 ENDWHILE
RETURN

SUB AddList(FN,LN,Age,AC,ID)
  DEF s:INT

  s = GETSTATE(Main,8)
  IF s = 1
    node = NEW(list,1)
    #node.FName = FN
    #node.LName = LN
    #node.Age = Age
    #node.AreaCode = AC
    #node.UniqueID = ID
    #node.Nxt = start.nxt
    start.nxt = node
  ELSE
    node = start
    WHILE (#node.Nxt <> 0)
    node = #node.Nxt
    ENDWHILE
    #node.Nxt = NEW(list,1)
    node = #node.Nxt
    #node.FName = FN
    #node.LName = LN
    #node.Age = Age
    #node.AreaCode = AC
    #node.UniqueID = ID
    #node.Nxt = 0
  ENDIF
RETURN

SUB DeleteLinkedList
 DEF ref, temp:POINTER

 ref = start.Nxt
 IF(ref)
  WHILE #ref.Nxt
   temp = ref
   ref = #ref.Nxt
   DELETE temp
  ENDWHILE
 ENDIF
 start.Nxt = 0
 node = start
 SENDMESSAGE Main,@LB_RESETCONTENT,0,0,5
RETURN
```

SUGGESTED PROJECT

(a) Create a Dialog to handle input of the data. When the Create List Button is clicked the dialog should open and allow you to type in the UDT data. You will also need to add a SAVE Button to the Dialog to save the data into the Linked List.

(b) Change the ListBox to a ListView Control.

That is all for this article covering building a Single Linked List, Printing the List, and Deleting the List. In the next article we will Delete an Item from the List, Sort the List and Save to Binary File. The Delete and Sort functions have their own set of Rules just as the Add to Linked List has its rules.

Happy programming until then - Bizzy

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

The IBasic Users Profile Page

Every month here at IBasic Monthly we'd like to profile one of our users so others can see the quality of people attracted to the IBasic programming language. These great users are also available on the forum to offer help and suggestions for beginner and experienced alike

This month we'd like to profile IBasicPower (formerly known as SoulTaker).
The IBasicPower web site may be found at the following link below the icon.

Joined: 26 Nov 2002

http://www.ibasicpower.com/

IBasicPower was formally schooled in programming using mainframe computers, and has been programming since the early 1980's.

Languages used: asm, basic, C, C++ and cobol

Experience: Wrote code samples for the book "Visual Basic Source Code Library. He's also a Microsoft Beta Tester and Developer. He currently works for Pitney Bowes DMT (Document Messaging Technologies)

IBasicPower has recently registered the domain name www.ibasicpower.com and is up and running and will offer many IB related tips and links and a program or two.

About the web site IBasicPower told IBasic Monthly "I'm doing this Web Site to help other's and to help promote the IBasic Programming Language."

Be sure to watch his web site for updates and useful bits from this experienced programmer.

IBasic Monthly
Dedicated to everything IBasic!

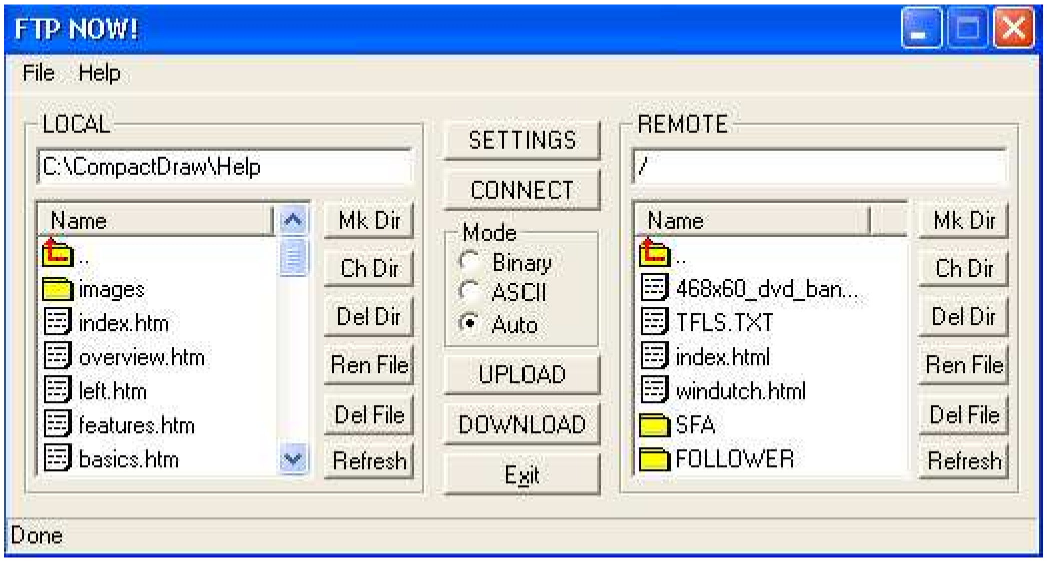
December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

FTP NOW!

Creating a Windows Internet Application with IBasic PART II

By

BIZZY



In the first article we started out by planning the Application and then building an outline of the FTP Now! program. In this article all the Code will be added to make a fully functional Internet Application.

The Menu Options, Button Controls and File Lists will all need Code associated with them. With the Code also we will need more Const, Declare API, UDTs (User Defined Type) and Def Variables.

The first thing we will put in the program will be the Const, Declares, UDTs and Defs and then work with the Windows Messaging in our Application to build all the Code needed for each Control.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Here is the list of all the SETID, CONST in the program. There are a few SETID that are not used in the program but are there for future development needs. As you read through the Code you can reference back to this list.

Most of the Constants can be obtained from API Viewer (Link on Pyxia Web Site)

```
’ Internet Flag Constants
SETID "FTP_TRANSFER_TYPE_UNKNOWN",0x0
SETID "FTP_TRANSFER_TYPE_ASCII",0x1
SETID "FTP_TRANSFER_TYPE_BINARY",0x2
SETID "INTERNET_DEFAULT_FTP_PORT",21
SETID "INTERNET_SERVICE_FTP",1
SETID "INTERNET_FLAG_PASSIVE", 0x8000000
SETID "INTERNET_OPEN_TYPE_PRECONFIG", 0
SETID "INTERNET_OPEN_TYPE_DIRECT", 1
SETID "INTERNET_OPEN_TYPE_PROXY", 3
SETID "INTERNET_OPEN_TYPE_PRECONFIG_WITH_NO_AUTOPROXY", 4
’ List View - Image List
SETID "SM_CXSMICON", 49
SETID "SM_CYSMICON", 50
’ List View
SETID "LVSIL_SMALL", 1
SETID "LVM_FIRST", 0x1000
SETID "LVM_SETIMAGELIST", (@LVM_FIRST + 3)
SETID "LVM_GETITEMA",(@LVM_FIRST + 5)
SETID "LVM_SETITEMA",(@LVM_FIRST + 6)
SETID "LVIS_OVERLAYMASK",0xF00
SETID "LVS_SHAREIMAGELISTS", 0x40
SETID "LVS_SHOWSELALWAYS", 0x8
’ List View and Local Files
SETID "FILE_ATTRIBUTE_DIRECTORY",0x10
’ List View Image
CONST LVIF_IMAGE = 0x2
CONST LVIF_STATE = 0x8
CONST LVIF_TEXT = 0x1
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
' Window Automation
SETID "AW_VER_NEGATIVE",0x8
SETID "AW_HIDE",0x10000
SETID "AW_ACTIVATE",0x20000
SETID "AW_BLEND",0x80000
' File String Length
CONST MAX_PATH = 260
```

All the User Defined Type's used in the program are listed below. You will see them used throughout the SUBs. They also can be found in the API Viewer (Link on Pyxia Web Site), except for Internet Params (params), which has been made for our Internet Parameter File.

```
' UDT - List View Item
TYPE LV_ITEM
    DEF mask:UINT
    DEF iItem:INT
    DEF iSubItem:INT
    DEF state:UINT
    DEF stateMask:UINT
    DEF pszText:STRING
    DEF cchTextMax:INT
    DEF iImage:INT
    DEF lParam:INT
ENDTYPE

' UDT - Select Folder - Local
TYPE BROWSEINFO
    DEF hOwner:INT
    DEF pidlRoot:INT
    DEF pszDisplayName:STRING
    DEF lpszTitle:STRING
    DEF ulFlags:INT
    DEF lpfn:INT
    DEF lParam:INT
    DEF iImage:INT
```

```
' UDT - List View
TYPE NMLISTVIEW
    DEF hwndFrom:INT
    DEF idFrom:INT
    DEF code:INT
    DEF iItem:INT
    DEF iSubItem:INT
    DEF uNewState:INT
    DEF uOldState:INT
    DEF uChanged:INT
    DEF ptActionx:INT
    DEF ptActiony:INT
    DEF lParam:INT
ENDTYPE

' UDT - Files
TYPE WIN32_FIND_DATA
    DEF dwFileAttributes:INT
    DEF ftCreationTimeLow:INT
    DEF ftCreationTimeHigh:INT
    DEF ftLastAccessTimeLow:INT
    DEF ftLastAccessTimeHigh:INT
    DEF ftLastWriteTimeLow:INT
    DEF ftLastWriteTimeHigh:INT
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
ENDTYPE

' UDT - Internet Params
TYPE params
    DEF profile:STRING
    DEF host:STRING
    DEF userid:STRING
    DEF pword:STRING
ENDTYPE
```

```
    DEF nFileSizeHigh:INT
    DEF nFileSizeLow:INT
    DEF dwReserved0:INT
    DEF deReserved1:INT
    DEF cFileName[259]:ISTRING
    DEF cAlternate[13]:ISTRING
ENDTYPE
```

The following list is all the Windows API functions that we will be using in the FTP NOW! program. You need the API Viewer (Link on Pyxia Web Site) as it has many of the APIs in IBasic format. Most of the names of the DLLs say what the function does.

Further Reference can be made by the Win32.Hlp File.

NOTE: All API DECLARE Functions must be on one line in IBasic Code Editor!

```
' Wininet is the Windows DLL that we use for the Internet Functions
DECLARE
"wininet",InternetConnectA(session:INT,server:STRING,port:WORD,username:STRING,pass:STRIN
G,
  service:INT,flags:INT,context:INT),INT
DECLARE
"wininet",InternetOpenA(agent:STRING,access:INT,proxyname:STRING,proxybypass:STRING,
  flags:INT),INT
DECLARE "wininet",InternetCloseHandle(handle:INT),INT
DECLARE "wininet",FtpSetCurrentDirectoryA(handle:INT,url:STRING),INT
DECLARE "wininet",FtpGetCurrentDirectoryA(hConnect:INT, lpszCurrentDirectory:STRING,
  lpdwCurrentDirectory:POINTER),INT
DECLARE "wininet",FtpCreateDirectoryA(hConnect:INT, lpszDirectory:STRING),INT
DECLARE "wininet",FtpRemoveDirectoryA(hConnect:INT, lpszDirectory:STRING),INT
DECLARE "wininet",FtpDeleteFileA(hConnect:INT, lpszFileName:STRING),INT
DECLARE "wininet",FtpRenameFileA(hConnect:INT, lpszExisting:STRING, lpszNew:STRING),INT
DECLARE "wininet",FtpGetFileA(hConnect:INT, lpszRemoteFile:STRING, lpszNewFile:STRING,
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
fFailIfExists:INT,  dwFlagsAndAttributes:INT, dwFlags:INT, dwContext:INT),INT
DECLARE "wininet",FtpPutFileA(hConnect:INT, lpszLocalFile:STRING,
lpszNewRemoteFile:STRING,dwFlags:INT, dwContext:INT),INT
DECLARE "wininet",FtpFindFirstFileA(hConnect:INT, lpszSearchFile:STRING,
lpFindFileData:MEMORY, dwFlags:INT, dwContext:INT),INT
DECLARE "wininet",InternetFindNextFileA(hFind:INT, lpvFindData:MEMORY),INT
' Used in the Get Folder Locally SUB for changing Folders
DECLARE "shell32.dll",SHGetPathFromIDList(pidl:INT,pszPath:STRING),INT
DECLARE "shell32.dll",SHBrowseForFolder(lpbi:BROWSEINFO),INT
DECLARE "ole32",CoTaskMemFree(pidl:INT)
' Used for File Attrributes Remote and Local
DECLARE "Kernel32",GetFileAttributesA(lpFileName:STRING),INT
' Used with Image List and List View
DECLARE "user32",GetDlgItem(hDlg:INT, nIDDlgItem:INT),INT
DECLARE "user32",SendMessageA(wnd:INT,message:INT,wparam:INT,lparam:LV_ITEM),INT
' Used with the Image List
DECLARE "comctl32",ImageList_AddIcon(himl:INT, hicon:INT),INT
DECLARE "comctl32",ImageList_Create(cx:INT, cy:INT, flags:UINT, cInitial:INT, cGrow:INT ),INT
DECLARE "user32",GetSystemMetrics(item:INT),INT
' Used to Animate the opening and closing of the FTP NOW! Main Window
DECLARE "user32.dll",AnimateWindow(hwnd:INT, dwTime:INT, dwFlags:INT),INT
```

Variables used in the FTP NOW! Program are listed below. You will find them used throughout the program Code. These Variables are all Global Variables and can be used from any part of the Program.

```
' Main Window of FTP NOW! dialog
DEF d1:DIALOG
' Internet Params Dialog
DEF d2:DIALOG
' Input Dialog to get User Input
DEF InpDia:DIALOG
' About Dialog for General Info
DEF about:DIALOG
' Help Dialog to display Help
DEF helper:DIALOG
```

```
' Variables used with the Internet Params
' Settings
DEF profile,site,userid,pword,ref:STRING

' Used for Input Dialog to get Folder name
DEF DirName:STRING
' For User Input Dialog - to set controls
DEF IDLabel,IDName:STRING
' For Status Bar
```

IBasic Monthly
Dedicated to everything IBasic!

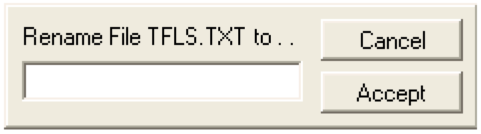
December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
' Internet Variables used with API
DEF hopen,hconnect,hhttp:INT
DEF infile,outfile:STRING
' Internet Params File
DEF paramfile:BFILE
' UDT for Internet Params
DEF settings:params
DEF panes[1]:INT

' List View UDT Variables
DEF lv:NMLISTVIEW
DEF lvi:LV_ITEM
' Handles to the List Views for inserting Icons
DEF hwndListViewLoc,hwndListViewRem:INT
' Main Program Run Variable
DEF run:INT
```

Well all the Variables and so on have been dealt with. There are three small Dialog Windows that still need to be created. They are the Input Dialog for User Input, Help Dialog to Display Help, and an About Dialog to show information about FTP NOW!

We will do the three small Dialog Windows first and then go on and Code the Program.

The Input Dialog (InpDia) which we will use to get User Input regarding Folders and Files needs to be done next as it will be used throughout the Code. The Dialog is created the same way as the Internet Params Dialog (d2) ***(See Part I - Animated Gif)*****.**

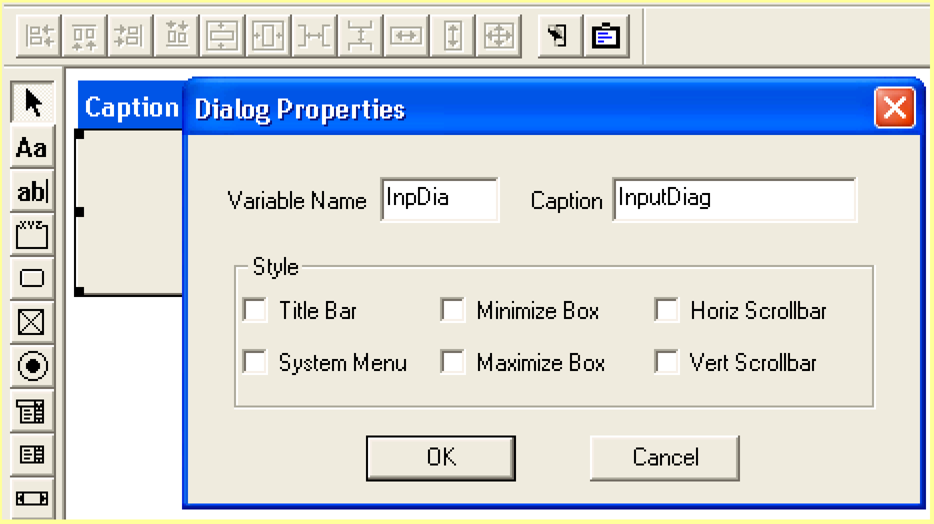


1.	Open the **Dialog Editor**	**CREATE INPUT DIALOG IN THE DIALOG EDITOR**
2.	Type in **Variable Name** and **Caption**. Turn off all Check Box items. ***(Image at Right)***	
3.	Place a **Static Text** Control on the Dialog and set its **Caption** to Name	
4.	Place an **Edit**	

IBasic Monthly
Dedicated to everything IBasic!

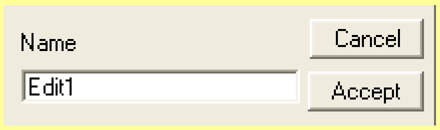
December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	Control on Dialog and set **Tabstop Check Box.**
5.	Place a **Cancel** Button and an **Accept** Button on the Dialog.
6.	Click **Generate Source** Icon
7.	Paste Dialog Code (**InpDia**) into Main Code Window.


Caption Dialog Properties

Variable Name: InpDia
Caption: InputDiag

Style
- [ ] Title Bar
- [ ] Minimize Box
- [ ] Horiz Scrollbar
- [ ] System Menu
- [ ] Maximize Box
- [ ] Vert Scrollbar



INPUT DIALOG WINDOWS MESSAGE SUB

1.	Make a **SELECT** Statement on **@CLASS**
2.	Check if it is **@IDCONTROL**
3.	Make another **SELECT** Statement on **@CONTROLID**
4.	**If CASE = 3 (Cancel** Button ID) which means the User has clicked the **Cancel** Button in the **Input Dialog**. Set **DirName** to an empty string and close the Dialog.

```
'-----------------------------INPUT DIALOG------------------------------

DIALOG InpDia,0,0,232,58,0x80000080,d1,"",InputDiag
CONTROL InpDia,"T,Name,6,8,140,14,0x5000010B,1"
CONTROL InpDia,"E,,6,26,140,21,0x50810080,2"
CONTROL InpDia,"B,Cancel,155,6,70,20,0x50010000,3"
CONTROL InpDia,"B,Accept,155,32,70,20,0x50010000,4"

'---------------------INPUT DIALOG MESSAGES---------------------
SUB InputDiag
   SELECT @CLASS
       CASE @IDCONTROL
         SELECT @CONTROLID
            CASE 3
                  DirName = ""
                 CLOSEDIALOG InpDia,@IDCANCEL
            CASE 4
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

5. **If** CASE **= 4** (**Accept** Button ID) which mean the User has clicked the **Accept** Button. Get the Text from the **Edit Control** (**ID 2**) and put it into the **DirName** String Variable. Close the Dialog Window.

CASE @IDINITDIALOG
When the Dialog is created the INITDIALOG is used to set up the Dialog before it is displayed.

```
                DirName = GETCONTROLTEXT(InpDia, 2)
                CLOSEDIALOG InpDia,@IDOK
        ENDSELECT
        CASE @IDINITDIALOG
        ' Center Dialog on Screen
        CENTERWINDOW InpDia
        ' Set Static Text Label (ID 1) Caption to content of IDLabel
        SETCONTROLTEXT(InpDia, 1, IDLabel)
        ' Set Edit Control Text to the content of IDName
        SETCONTROLTEXT(InpDia, 2, IDName)
    ENDSELECT
RETURN
```

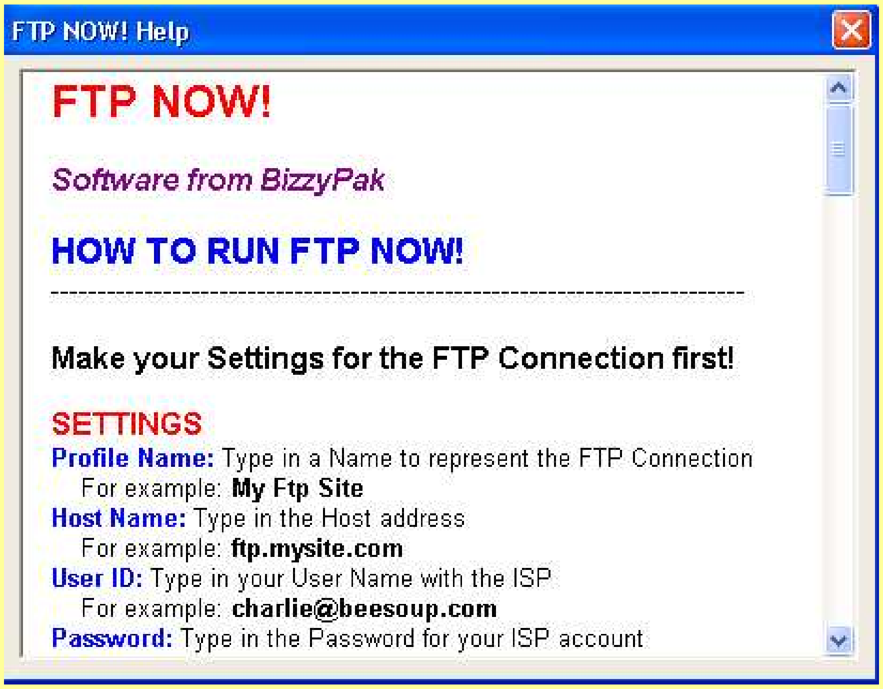
IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

HELP DIALOG WINDOW

The Help Dialog Window is created the same way as the Input Dialog (above) with the IBasic Dialog Editor. A Rich Edit is its only Control.

1.	The Windows Message Handler for the Dialog is **HelpHandler**
2.	Two Variables are Defined - one for a **File** and one for a **File Name**.
3.	**Filename** is set to the **Program's path** with the File **Help.rtf**
4.	A **Select Case Statement** receives the Window's Messages for this Dialog.
5.	The **CASE @IDTIMER** event when active causes the program to see if the **Rich Edit** Control has been created.
6.	If **Rich Edit** is created then the **Timer** is stopped, and
7.	The **File Help.rtf** is then opened . . .
8.	If the **File** opens then the contents of the **File Help.rtf** are loaded into the **Rich Edit.**


FTP NOW! Help

FTP NOW!

Software from BizzyPak

HOW TO RUN FTP NOW!

Make your Settings for the FTP Connection first!

SETTINGS
Profile Name: Type in a Name to represent the FTP Connection
For example: **My Ftp Site**
Host Name: Type in the Host address
For example: **ftp.mysite.com**
User ID: Type in your User Name with the ISP
For example: **charlie@beesoup.com**
Password: Type in the Password for your ISP account

```
'--------------------------------------HELP DIALOG--------------------
DIALOG helper,0,0,464,333,0x80C80080,d1,"FTP NOW!
Help",HelpHandler
CONTROL helper,"RE,,8,7,447,316,0x50A10804,1"

'----------------------------HELP DIALOG MESSAGES-----------
SUB HelpHandler
    DEF file1:FILE
    DEF filename:STRING

    filename = GETSTARTPATH + "Help.rtf"
    SELECT @CLASS
        CASE @IDTIMER
         IF CONTROLEXISTS(helper,1)
             STOPTIMER helper
             IF (OPENFILE(file1,filename,"R")=0)
                CONTROLCMD helper,1,@RTLOAD,file1,1
                CLOSEFILE file1
            ENDIF
         ENDIF
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

9.	The **File** is then Closed.
10.	If **@IDCLOSEWINDOW** - User has clicked the [X] to close the Dialog.
11.	When the **helper** Dialog Window is created the **CASE @IDINITDIALOG** is used to set up the Dialog Window before it is Displayed.
12.	The **Rich Edit** Margins are set to **15**.
13.	The **Time** is **Started** for 100 milliseconds.
14.	The **helper** Dialog Window is **centered**.

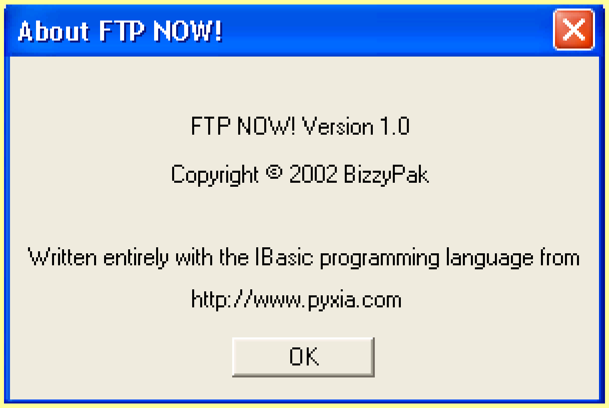
```
        CASE @IDCLOSEWINDOW
          CLOSEDIALOG helper,@IDOK
        CASE @IDINITDIALOG
          CONTROLCMD helper,1,@RTSETMARGINS,15,0
          STARTTIMER helper, 100
          CENTERWINDOW helper
    ENDSELECT
RETURN
```

The TIMER is turned on in the IDINITDIALOG so that we can load theHelp.rtf File after the Dialog is Displayed.

ABOUT DIALOG

The About Dialog is accessed from the Menu and consists of four Static Text Controls and one Button.

Create the **About Dialog** by opening the **IBasic Dialog Editor** and placing the **Controls** as shown in the Image to the right.


About FTP NOW!

FTP NOW! Version 1.0

Copyright © 2002 BizzyPak

Written entirely with the IBasic programming language from

http://www.pyxia.com

1.	The Messages from Windows is processed in the **SUB abouthandler**
2.	Use a **Select Case** Statement to

```
'------------------------------------ABOUT BOX-----------------
DIALOG about,0,0,295,172,0x80C80080,d1,"About FTP NOW!",abouthandler
CONTROL about,"T,FTP NOW! Version 1.0,90,29,115,22,0x50000100,1"
CONTROL about,"T,Copyright © 2002
```

	process the Message.
3.	**CASE @IDCLOSEWINDOW** - indicates the User clicked the **[X]** to close the **About Dialog**.
4.	If **CASE @IDCONTROL** event is active then check if the **@CONTROLID = 3** and if so close the **About Dialog**.
5.	**CASE @IDINITDIALOG** is used to initialise the **Dialog** before it is Displayed and sets the **Dialog** to **Center Screen**.

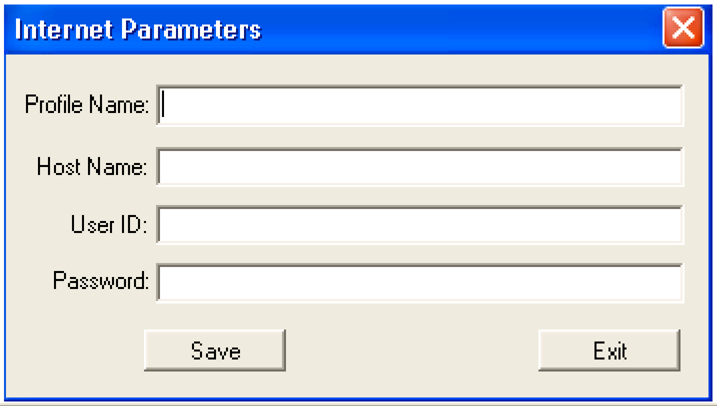
```
BizzyPak,80,53,182,19,0x50000100,2"
CONTROL about,"B,OK,112,140,70,20,0x50000001,3"
CONTROL about,"T,Written entirely with the IBasic
programming language from, 10,94,277,19,0x50000100,4"
CONTROL
about,"T,http://www.pyxia.com,91,115,113,18,0x50000100,5"

'-----------------------------ABOUT BOX MESSAGES--------
SUB abouthandler
   SELECT @CLASS
       CASE @IDCLOSEWINDOW
         CLOSEDIALOG about, @IDOK
       CASE @IDCONTROL
         IF @CONTROLID = 3 THEN
CLOSEDIALOG(about,@IDOK)
       CASE @IDINITDIALOG
         CENTERWINDOW about
   ENDSELECT
RETURN
```

CODE THE PROGRAM

INTERNET PARAMETERS

So far the **Internet Parameters Dialog** (**d2, DiagTwo**)
has been completed ***(in Part I)*** except for the two **SUBS** (**ObtainParamFile**, **WriteParamFile**) that it needs.

So we will complete the two **SUBs** now.



The ObtainParamFile SUB is used to read the Disc File PARAM.DAT and put the settings in the File into the Internet Parameters Display	Opens PARAM.DAT File on Disc and Reads File into Dialog Window Controls

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Window.

1.	Set Cursor to **WAIT**
2.	Openfile **PARAM.DAT** to **R**ead Mode
3.	Get Data from **File** into the UDT **settings**
4.	Close File **PARAM.DAT**
5.	Set Variables **profile, site, userid,** and **pword** from the **settings** UDT
6	Set Controls with the text in the variables.
7.	Set Cursor to **ARROW**

```
SUB ObtainParamFile
  SETCURSOR (d2, @CSWAIT)
  IF(OPENFILE(paramfile,GETSTARTPATH
+"PARAM.DAT","R") = 0)
    GET paramfile,1,settings
    CLOSEFILE paramfile
    ' Transfer to DiagTwo edit boxes Variables used by -
    ' FtpPut/GetFile
    profile = settings.profile
    site = settings.host
    userid = settings.userid
    pword = settings.pword
    SETCONTROLTEXT d2,5,profile
    SETCONTROLTEXT d2,6,site
    SETCONTROLTEXT d2,7,userid
    SETCONTROLTEXT d2,8,pword
  ENDIF
  SETCURSOR (d2, @CSARROW)
RETURN
```

The WriteParamFile SUB is used to write to the Disc File PARAM.DAT saving the settings from the Internet Parameters Display Window.

1.	Set Cursor to **WAIT**
2.	Get text from Controls into **settings** UDT variable.
3.	Openfile **PARAM.DAT** to **W**rite Mode
4.	Put **settings** UDT data into file.
5.	Close **PARAM.DAT** File

Opens PARAM.DAT File on Disc, Gets Control text into settings and Saves the Data

```
SUB WriteParamFile
  SETCURSOR (d2, @CSWAIT)
  settings.profile = GETCONTROLTEXT(d2,5)
  settings.host = GETCONTROLTEXT(d2,6)
  settings.userid = GETCONTROLTEXT(d2,7)
  settings.pword = GETCONTROLTEXT(d2,8)

  IF (OPENFILE(paramfile,GETSTARTPATH
+"PARAM.DAT","W") = 0)
    PUT paramfile,1,settings
    CLOSEFILE paramfile
  ENDIF
```

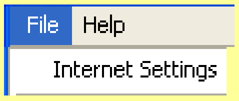
IBasic Monthly
Dedicated to everything IBasic!

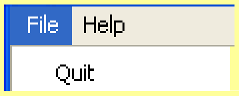
December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

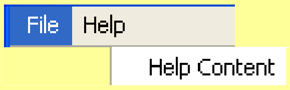
6. Set Cursor to **ARROW**	**SETCURSOR** (d2, **@CSARROW**) **RETURN**

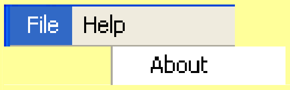
Main Window (d1) Windows Message Handler Sub is the heart of the program where most events are processed. The GREEN TEXT in the Panel below are comments on the Code and what it is used for. Where Controls are mentioned in the Code you will see the GREEN TEXT IS UNDERLINED. Each Area of the CODE in the SUB DiagOne has a Header Comment to show what part it relates to in the program. The Images in the left margin show which Controls activate each Message.

```
--------------------------------MAIN DIALOG MESSAGES---------------
SUB DiagOne
   DEF stattext:STRING                     String used with Status Bar
   DEF left,top,width,height:INT            Variables for Window Size
   DEF mem:MEMORY                          Variable for List View UDT

   SELECT @CLASS
 ------------------------------------------ MENU OPTIONS -----------------
      CASE@IDMENUPICK
        SELECT@MENUNUM
        CASE 1
              DoSettings          SUB Internet Params - PARAM.DAT
              BeginWithFile         SUB Read File - Internet Params,  "
        CASE 2
             Close Internet Connect - API
              InternetCloseHandle(hconnect)
              InternetCloseHandle(hopen)    Close Internet Open - API
              run = 0                        Close Main Dialog Window (d1)
        CASE 3
         DOMODAL helper         Help Contents - Display Help Dialog
        CASE 4
         DOMODAL about          About Box - Display About Dialog
        ENDSELECT
 -----------------------DIALOG CLOSE WINDOW [X] CLICKED------
```





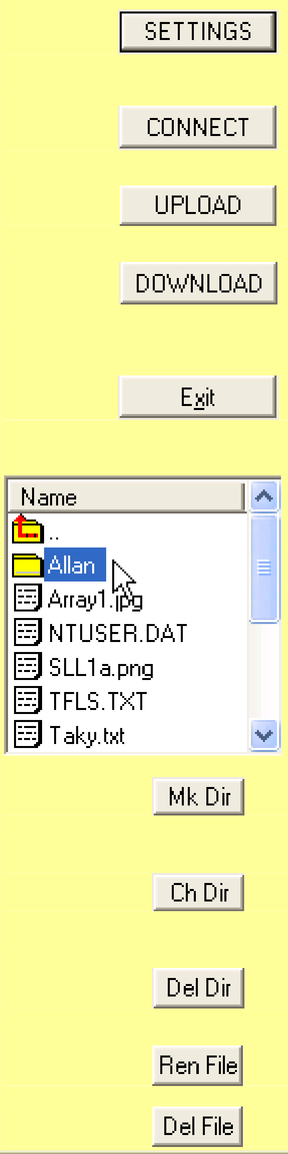




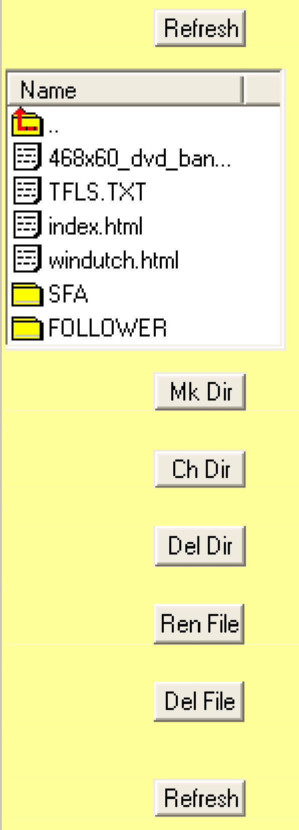
IBasic Monthly

Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

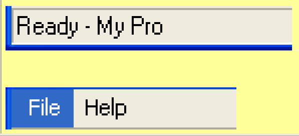


```
    CASE @IDCLOSEWINDOW
            Close Internet Connect - API
            InternetCloseHandle(hconnect)
            InternetCloseHandle(hopen)     Close Internet Open - API
            run = 0                        Close Main Dialog Window (d1)
-------------------- CONTROLS IN MAIN DIALOG WINDOW (d1)---
    CASE@IDCONTROL
      SELECT@CONTROLID
      CASE 1                             SETTINGS BUTTON
            DoSettings         SUB Internet Params - PARAM.DAT
            BeginWithFile      SUB Read File - Internet Params,  "
      CASE 2                             CONNECT BUTTON
            ConnectSite          SUB Connect  via Internet Parameter
      CASE 6                             UPLOAD BUTTON
            UploadFiles                  SUB Upload Files to Web
      CASE 7                             DOWNLOAD BUTTON
            DownloadFiles       SUB Download Files to Local Folder
      CASE 8                             Exit Button
            InternetCloseHandle(hconnect)  Close Internet Connect
            InternetCloseHandle(hopen)     Close Internet Open
            run = 0                      Close Main Dialog Window (d1)

---------------------------------- LOCAL GROUP AREA------------------
      CASE 10              Change Dir Local by Double Click in LV
       IF(@NOTIFYCODE = @NMDBLCLK)
          mem = @QUAL                  NMLISTVIEW UDT in Memory
         READMEM mem,1,lv               "           "   in lv Variable
          DClkChDirLoc                 SUB - Change Local Folder
       ENDIF

      CASE 11                          MK DIR BUTTON
            MakeDirLocal                SUB - Make Dir Local
      CASE 12                          CH DIR BUTTON
            SelectFolder       SUB - Select Folder to Change Dir Local
            GetLocalFolder     SUB - Reads Selected Folder into File List
      CASE 13                          DEL DIR BUTTON
             DeleteDirLocal             SUB - Delete Dir Local
      CASE 14                          REN FILE BUTTON
             RenameFileLocal            Rename File in Local Folder
      CASE 15                          DEL FILE BUTTON
```



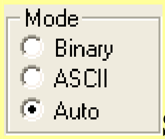
```
          DeleteFileLocal            Delete File in Local Folder
      CASE 16                        REFRESH BUTTON
          GetLocalFolder             Refresh Dir Local

------------------------------- REMOTE GROUP AREA--------------------
      CASE 20               Change Dir Remote - Double Click in LV
        IF(@NOTIFYCODE = @NMDBLCLK)
          mem = @QUAL                NMLISTVIEW UDT in Memory
          READMEM mem,1,lv           "          "   in lv Variable
          DClkChDirRem               SUB - Change Folder on Remote Site
        ENDIF

      CASE 21                        MK DIR BUTTON
           MakeDirRem                SUB - Make Dir on Remote Site
      CASE 22                        CH DIR BUTTON
           ChangeDirRemote           SUB - Change Dir on Remote Site
      CASE 23                        DEL DIR BUTTON
           DelDirRemote              SUB - Del Dir Remote Site
      CASE 24                        REN FILE BUTTON
           RenameFileRem             SUB - Rename File on Remote Site
      CASE 25                        DEL FILE BUTTON
           DeleteFileRem             Delete File on Remote Site
      CASE 26                        REFRESH BUTTON
          GetRemoteFolder            Refresh Remote Dir
      ENDSELECT
--------------------Indicated Size Change of Dialog Window (d1) -------
     CASE@IDSIZE
      ' Check to see if the Control (Status Bar - ID 40) Exists
      IF CONTROLEXISTS(d1,40)
         ' Tell the status window we are sizing
        CONTROLCMD d1,40,@SWRESIZE
         ' Get the client size of the window
        GETCLIENTSIZE d1,left,top,width,height
         panes = -1                         Set panes to full width  ( -1 )
        CONTROLCMD d1,40,@SWSETPANES,1,panes
         stattext = "Ready - " + settings.profile
        '  Put Text into Status Bar
        CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext
      ENDIF
-------------------------INITIALIZE MAIN DIALOG WINDOW-------
```

Status Bar



IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2



Set State

Status Bar

```
        CASE@IDINITDIALOG
          ' Put Menu Options into d1
         MENU d1, "T, File, 0, 0", "I, Internet Settings, 0, 1", "I,-,0, 0", "I, Quit, 0
,2"
         INSERTMENU d1,1, "T,Help,0,0", "I,Help Content,0,3", "I,About,0,4"
         CENTERWINDOW d1   Center the Main Dialog Window (d1)
         SETSTATE d1, 5, 1        Set Group Radio Buttons - Auto Mode
          BeginWithFile           SUB - Opens File for Parameter Settings
          ' Set up the List View Controls - Columns, Captions and Widths
          ' LOCAL List View Control
         CONTROLCMD d1,10,@LVINSERTCOLUMN,0,"Name"
         CONTROLCMD d1,10,@LVINSERTCOLUMN,1,""
         CONTROLCMD d1,10,@LVSETCOLWIDTH,0,118
         CONTROLCMD d1,10,@LVSETCOLWIDTH,1,0
          ' REMOTE List View Control
         CONTROLCMD d1,20,@LVINSERTCOLUMN,0,"Name"
         CONTROLCMD d1,20,@LVINSERTCOLUMN,1,""
         CONTROLCMD d1,20,@LVSETCOLWIDTH,0,118
         CONTROLCMD d1,20,@LVSETCOLWIDTH,1,0
          ref = "FTPNow"                        Variable used in Internet Connect
          IDLabel = "Name"                      Input Dialog Static Text Control
          IDName = ""                           Input Dialog Edit Control
          'Get the Windows client size and set up 1 pane for the Status Bar
         GETCLIENTSIZE d1,left,top,width,height
          panes = -1                            Set panes to full width  ( -1 )
         CONTROLCMD d1,40,@SWSETPANES,1,panes
          stattext = "Ready - " + settings.profile
         ' Set the initial pane text
         CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext
          ' Animate the Main Dialog Window Effect when it Opens
          AnimateWindow(d1, 500,
@AW_VER_NEGATIVE|@AW_ACTIVATE) :' API
   ENDSELECT
RETURN
```

THE CODE THAT STARTS THE FTP NOW! PROGRAM

The following code starts the FTP NOW! program running.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

----------------------DESCRIPTION------

SHOWDIALOG d1 - Displays the Main
Dialog
Window of **FTP NOW!**

CreateTheImage is a **SUB** that creates an
Image
List of the **Icons** used in the **List View**
Controls.

hwndListViewLoc and **hwndListViewRem**
are
Handles to the **LOCAL** and **REMOTE List**
View
Controls. The **Handles** are used to add the
Icons
to the **List View Controls**.

WAITUNTIL run=0 is a loop control to keep
the **Windows Messages getting processed**
while
run = 1

AnimateWindow is an **API Function** to end
the
Display of the **FTP NOW! Window.**

CLOSEDIALOG d1 - closes **FTP NOW!**
Window.
END - Ends the FTP NOW! Program.

------------------------------CODE-------------------

----------------------START PROGRAM----------

```
 SHOWDIALOG d1

CreateTheImage SUB - Creates Image List for Icons

hwndListViewLoc = GetDlgItem(d1, 2024+10)
hwndListViewRem = GetDlgItem(d1, 2024+20)

run = 1

WAITUNTIL run = 0

AnimateWindow(d1, 600,
@AW_BLEND|@AW_HIDE)

CLOSEDIALOG d1

END
```

THE SUBROUTINES THAT WORK FROM THE ABOVE START UP CODE AND THE MAIN MESSAGE SUB DiagOne ARE THE ONLY THING LEFT TO DO. We will work down through the DiagOne Sub to create the Subs.

The Image List is created to make the Icons available for the two List View Controls. Because the Image List is used by more than one List View it is necessary to set the

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

LVS_SHAREIMAGELISTS flag in both of our List View Controls. They are ID 10 and ID 20.

1.	**Create** Image List
2.	**Loadimage** from File
3.	Add loaded **Icon** to the **Image List**
4.	2. and 3. are done 3 times to load the 3 **Icons** needed.
5.	Sendmessage to **List View** (**ID 10**) to set **Image List** into the **List View**.
6.	Sendmessage to **List View** (**ID 20**) to set **Image List** into the **List View**.

```
---------------------------------------------Image List---------------------------
SUB CreateTheImage
   DEF hiItem1,hiItem2,hiItem3:INT
   DEF himlSmall:INT

   himlSmall=ImageList_Create(GetSystemMetrics(@SM_CXSMICON),
[Next Line]
     GetSystemMetrics(@SM_CYSMICON), 1, 3, 1)

   hiItem1 = LOADIMAGE
(GETSTARTPATH+"UpFold.ico",@IMGICON)
   ImageList_AddIcon(himlSmall, hiItem1)

   hiItem2 = LOADIMAGE
(GETSTARTPATH+"Fold1.ico",@IMGICON)
   ImageList_AddIcon(himlSmall, hiItem2)

   hiItem3 = LOADIMAGE (GETSTARTPATH+"Text.ico",@IMGICON)
   ImageList_AddIcon(himlSmall, hiItem3)

   SENDMESSAGE d1,@LVM_SETIMAGELIST,@LVSIL_SMALL,
himlSmall,10
   SENDMESSAGE d1,@LVM_SETIMAGELIST,@LVSIL_SMALL,
himlSmall,20
RETURN
```

The DoSettings SUB activates the Internet Parameters Dialog Window

```
------------------------------------------SETTINGS--------------------
SUB DoSettings
    ' Load the settings dialog
    DOMODAL d2
RETURN
```

The BeginWithFile Sub loads the

```
---------------------------------------GET PARAM.DAT FILE------
SUB BeginWithFile
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

PARAM.DAT File from Disc and sets the Internet Connect Variables.

1.	Set Cursor **WAIT**
2.	Open the File **PARAM.DAT**
3.	Read into **settings**
4.	Set Variables with **settings** data
5.	Set Cursor **ARROW**

```
    SETCURSOR (d2, @CSWAIT)
    IF (OPENFILE(paramfile,GETSTARTPATH
+"PARAM.DAT","R") = 0)
        GET paramfile,1,settings
        CLOSEFILE paramfile
' Transfer to DiagTwo edit boxes and Variables used by
FtpPutFile
        profile = settings.profile
        site = settings.host
        userid = settings.userid
        pword = settings.pword
    ENDIF
    SETCURSOR (d2, @CSARROW)
RETURN
```

The ConnectSite Sub connects the FTP NOW! Program to theInternet Site that is in the Variable site (settings.site)

CONNECT

1.	Set Cursor **WAIT**
2.	Open settings for Internet from the **Registery**
3.	Set **Status Bar** with "Connected"
4.	Connect to site with **userid** and **pword** with **FTP** Service
5.	Set **Status Bar** with "Connected" + **site**
6.	**GetRemoteFolder** SUB. Reads the Folder on the Internet Site

```
------------------------------------------CONNECT--------------------
SUB ConnectSite
    DEF stattext:STRING

    SETCURSOR (d2, @CSWAIT)   ' Open an internet connection
with settings in Registery (Preconfig)
    hopen =
InternetOpenA(ref,@INTERNET_OPEN_TYPE_PRECONFIG,"",
"",0)

    IF (hopen)
        stattext = "Connected"
        CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext  '
Connect to Internet Site with userid and pword using FTP Service
        hconnect = InternetConnectA(hopen,site,
@INTERNET_DEFAULT_FTP_PORT,userid,pword,
@INTERNET_SERVICE_FTP,@INTERNET_FLAG_PASSIVE,
0)
        IF (hconnect)
            stattext = "Connected - " + site
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

7.	**Status Bar** "Done"
8.	Set Cursor **ARROW**

```
        CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext
         GetRemoteFolder     :' SUB  Read Folder on Internet Site
into LV 20
        ENDIF
    ENDIF

    stattext = "Done"
    CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext
    SETCURSOR (d2, @CSARROW)
RETURN
```

The UploadFiles Sub Uploads Files from the Local Computer to the Web Site address which is set in the List View (ID 20)

1.	Set Cursor **WAIT**
2.	Check Mode for **Transfer Mode**
3.	Get number of entries in **List View**
4.	**WHILE** item <= **count**. Iterate through **List View**
5.	If **Selected** and the **Attribute** = '0' then it is a Selected Item in the **List View**
6.	Check if the **File Attrib** is **ASCII** (".HTM or .TXT") **Done to ensure the settings by User are correct.**

```
---------------UPLOAD FILES TO INTERNET SERVER
SUB UploadFiles
    DEF count, item, selected, Ans, error, AcFnd:INT
    DEF Str1, text, DirPth, Attrib, source, dest:STRING
    DEF TheState, Pos:INT

    SETCURSOR (d2, @CSWAIT)
    ' Check for type of Transfer Mode
    IF (GETSTATE(d1, 5) = 1) THEN TheState = 3
    IF (GETSTATE(d1, 4) = 1) THEN TheState = 2
    IF (GETSTATE(d1, 3) = 1) THEN TheState = 1
   ' Get the Folder Path from the Edit Box (ID 9)
    DirPth = GETCONTROLTEXT(d1, 9)
    item = 0
    ' Get the number of Folders and Files in the List View
    count = CONTROLCMD( d1, 10, @LVGETCOUNT)
    ' Iterate through the List View with the WHILE Function
    WHILE (item <= count)
        selected = CONTROLCMD(d1, 10,
@LVGETSELECTED, item)
     text = CONTROLCMD(d1, 10, @LVGETTEXT, item, 0)
     Attrib = CONTROLCMD(d1, 10, @LVGETTEXT, item, 1)
' Check if this item in the List View is selected and is a File
Attrib = 0
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

7. **If TheState = 3**
   Auto Mode
   Auto Mode can be both **ASCII** and **Binary Mode** Files
8. **If AcFnd** then send by **ASCII Mode** (HTM or TXT) extensions were found in Files . . .
9. . . . else send by **Binary Mode**
10. **If TheState = 2**
    ASCII Mode
11. Send by **ASCII**
12. **If TheState = 1**
    Binary Mode
13. Send by **Binary Mode**
14. Set Cursor to **ARROW**
15. Set the **Status Bar** to "Done"
16. **GetRemoteFolder Sub**. The **Sub** will read the Web Folder into the **List View**, displaying the Updated File List after the Up-load is complete.

```
          AcFnd = 0
          Str1 = UCASE$(text)
          Pos = INSTR(1, Str1, ".HTM")
        IF(Pos > 0) THEN AcFnd = 1
          Pos = INSTR(1, Str1, ".TXT")
        IF (Pos > 0) THEN AcFnd = 1
          source = DirPth + "\" + text
          dest = text
          Str1 = ""
        IF (TheState = 3)
          IF (AcFnd > 0)
                Str1 = "Uploading " + text + " file by ASCII"
               CONTROLCMD
d1,40,@SWSETPANETEXT,0,Str1
                error = FtpPutFileA(hconnect, source,
dest,          [Next Line]
                         @FTP_TRANSFER_TYPE_ASCII, 0)
          ELSE
                Str1 = "Uploading " + text + " file by Binary"
               CONTROLCMD
d1,40,@SWSETPANETEXT,0,Str1
                error = FtpPutFileA(hconnect, source,
dest,         [Next Line]
              @FTP_TRANSFER_TYPE_BINARY, 0)
          ENDIF
       ELSE
         IF (TheState = 2)
              Str1 = "Uploading " + text + " file by ASCII"
             CONTROLCMD
d1,40,@SWSETPANETEXT,0,Str1
              error = FtpPutFileA(hconnect, source, dest,
[Next Line]
                     @FTP_TRANSFER_TYPE_ASCII, 0)
         ENDIF
         IF (TheState = 1)
             Str1 = "Uploading " + text + " file by Binary"
            CONTROLCMD
d1,40,@SWSETPANETEXT,0,Str1
             error = FtpPutFileA(hconnect, source,
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
dest,              [Next Line]
                        @FTP_TRANSFER_TYPE_BINARY, 0)
           ENDIF
         ENDIF
     ENDIF
     item = item + 1
     ENDWHILE
     SETCURSOR (d2, @CSARROW)
     Str1 = "Done"
     CONTROLCMD d1,40,@SWSETPANETEXT,0,Str1
     GetRemoteFolder
RETURN
```

The DownloadFiles Sub Downloads Files from the Web Site to the Local Folder address which is set in the List View (ID 10)

DOWNLOAD

1.	Set Cursor **WAIT**
2.	Check Mode for **Transfer Mode**
3.	Get number of entries in **List View**
4.	**WHILE** item <= **count**. Iterate through **List View**
5.	If **Selected** and the **Attribute** = '0' it is a Selected Item in **List View**
6.	Check if the **File Attrib** is **ASCII** (".HTM or .TXT") **Done to ensure**

```
-------------DOWNLOAD FILES FROM REMOTE FOLDER


SUB DownloadFiles
    DEF count,item,selected,Ans,error,AsciFnd:INT
    DEF text,DirPth,Attrib,source,dest:STRING
    DEF Temp,Str1:STRING
    DEF TheState:INT

    SETCURSOR (d2, @CSWAIT)
    ' Check for type of Transfer Mode User has Selected
    IF (GETSTATE(d1, 5) = 1) THEN TheState = 3
    IF (GETSTATE(d1, 4) = 1) THEN TheState = 2
    IF (GETSTATE(d1, 3) = 1) THEN TheState = 1
    ' Get the Path to the Folder to Download to
    DirPth = GETCONTROLTEXT(d1, 9)
    item = 0
    ' Get the number of Folders and Files in the List View (ID 10)
    count = CONTROLCMD( d1, 20, @LVGETCOUNT)
    ' Iterate through List View with WHILE Function
    WHILE (item <= count)
  selected = CONTROLCMD( d1, 20, @LVGETSELECTED, item)
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	the settings by User are correct.
7.	**If TheState = 3** **Auto Mode** **Auto Mode** can be both **ASCII** and **Binary Mode** Files
8.	**If AsciFnd** then send by **ASCII Mode** (HTM or TXT) extensions were found in Files . . .
9.	. . . else send by **Binary Mode**
10.	**If TheState = 2** **ASCII Mode**
11.	Send by **ASCII**
12.	**If TheState = 1** **Binary Mode**
13.	Send by **Binary Mode**
14.	Set Cursor to **ARROW**
15.	Set the **Status Bar** to "Done"
16.	**GetLocalFolder Sub**. The **Sub** will read the **Local Folder** into the **List View**, displaying the Updated File List after the Down-load is complete.

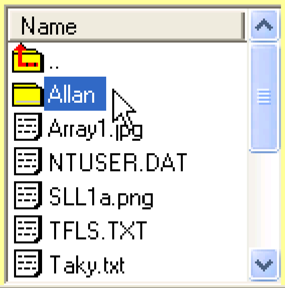
```
        Attrib = CONTROLCMD( d1, 20, @LVGETTEXT, item, 1 )
 ' Check if this item in the List View is selected and is a File Attrib
= 0
        IF ((selected > 0) &  (Attrib = "0"))
            source = text
            dest = DirPth + "\" + text
            AsciFnd = 0
            Str1 = UCASE$(text)
          IF (INSTR(Str1, ".HTM") > 0) THEN AsciFnd = 1
          IF (INSTR(Str1, ".TXT") > 0) THEN AsciFnd = 1
            ' Download the file
          IF (TheState = 3)
             IF (AsciFnd = 1)
                    Temp = "Downloading " + text + " file by ASCII"
                 CONTROLCMD
d1,40,@SWSETPANETEXT,0,Temp
                    error = FtpGetFileA(hconnect, source, dest, 1, 0,
[Next Line]
                              @FTP_TRANSFER_TYPE_ASCII, 0)
             ELSE
                    Temp = "Downloading " + text + " file by Binary"
                 CONTROLCMD
d1,40,@SWSETPANETEXT,0,Temp
                    error = FtpGetFileA(hconnect, source, dest, 1, 0,
                           @FTP_TRANSFER_TYPE_BINARY, 0)
            ENDIF
        ELSE
            IF (TheState = 2)
                    Temp = "Downloading " + text + " file by ASCII"
                 CONTROLCMD
d1,40,@SWSETPANETEXT,0,Temp
                    error = FtpGetFileA(hconnect, source, dest, 1, 0,
[Next Line]
                             @FTP_TRANSFER_TYPE_ASCII, 0)
            ENDIF
            IF (TheState = 1)
                    Temp = "Downloading " + text + " file by Binary"
                 CONTROLCMD
d1,40,@SWSETPANETEXT,0,Temp
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
                error = FtpGetFileA(hconnect, source, dest, 1, 0,
[Next Line]
                         @FTP_TRANSFER_TYPE_BINARY, 0)
            ENDIF
          ENDIF
        ENDIF
        item = item + 1
      ENDWHILE
      SETCURSOR (d2, @CSARROW)
      Temp = "Done"
      CONTROLCMD d1,40,@SWSETPANETEXT,0,Temp
      GetLocalFolder
RETURN
```

DClkChDirLoc Sub is used when you Double Click on a Folder in the Local List View. This bypasses the need to click the Ch Dir **(ChDir) Button.**



1.	Get **itemno** from the **lv.iItem**. This is the item you **Double Clicked** on in the **List View.**
2.	**FIRST OPTION** If **Attrib** = 6 you wish to go **up** one level in the **Folder Tree.**
3.	Get **SearchPath** from the **Edit Box** (**ID 9**).

```
------------DOUBLE CLICK - CHANGE FOLDER - LOCAL
SUB DClkChDirLoc

   DEF itemno,Fnd,loc:INT
   DEF Attrib,SearchPath,Paths:STRING

' itemno is the number of the item you Double Clicked on in
the List View
   itemno = lv.iItem

' DirName contains Name of the Folder you Double Clicked
in the List View
   DirName =
CONTROLCMD(d1,10,@LVGETTEXT,itemno,0)

   ' Attrib contains a number to represent DirName
Attribute
   Attrib =
CONTROLCMD(d1,10,@LVGETTEXT,itemno,1)

   ' OPTION ONE
   IF (Attrib = "6")
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	This is the Current Path of the **LOCAL List View.**
4.	With **INSTR** delete the **last Folder**. e.g.: "C:\Base\data" would become "C:\Base"
5.	Set **Edit Box** (**ID 9**) with **new Folder** Address. e.g.: "C:\Base"
6.	**GetLocalFolder** Sub to update the contents of the **List View.**
7.	**SECOND OPTION**
8.	📁If **Attrib = 16** you have **Double Clicked** on a **Folder**.
9.	Get **Current Folder Path** from **Edit Box** (**ID 9**) and add **DirName.** e.g.: "C:\Base" would become "C:\Base\data"
10.	Set **Edit Box** (**ID 9**) with **new Folder** Address. e.g.: "C:\Base"
11.	**GetLocalFolder** Sub to update the contents of the

```
 ' Get the Folder Address of the Current Folder in List View
        SearchPath = GETCONTROLTEXT (d1, 9)
        SearchPath = SearchPath
        Fnd = 1
        pos = 1

 ' Look for "\" in the SearchPath String to locate the LAST
"\"
        DO
          IF (INSTR(pos, SearchPath, "\") > 0) THEN loc = pos
ELSE Fnd = 0
              pos = pos + 1
        UNTIL Fnd = 0

  ' Copy into Paths the new path with the LEFT$ Function
        Paths = LEFT$(SearchPath, loc-1)

 ' Put the new Path into the Edit Box (ID 9) above the List
View
        SETCONTROLTEXT (d1, 9, Paths)

 ' Use the Sub GetLocal Folder to update the contents of the
New Folder
        GetLocalFolder

    ENDIF

    ' OPTION TWO
    IF (Attrib = "16")
' Get the Folder Address of the Current Folder in List View
        SearchPath = GETCONTROLTEXT (d1, 9)

' Add the Double Clicked Folder in the List View to the
SearchPath
        SearchPath = SearchPath + "\" + DirName

' Put new List View Folder Path into the Edit Box (ID 9)
[above List View]
        SETCONTROLTEXT (d1, 9, SearchPath)
```

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

List View.

```
' Use the Sub GetLocal Folder to update the contents of the New Folder
        GetLocalFolder

    ENDIF

RETURN
```

The MakeDirLocal Sub allows you to Create a New Folder.

Mk Dir

1.	Run the **InpDia Dialog**
2.	Check for **IDOK** and an entry in the **DirName** from the **InpDia Dialog**.
3.	Add the **DirName** to the **Path** in **Edit Box (ID 9)**
4.	📁 **CreateDir**
5.	**GetLocalFolder Sub** to **update Folder** contents.

```
-----------------------MAKE NEW DIRECTORY - LOCAL-
SUB MakeDirLocal

    DEF effor,Ans: INT
    DEF DirPath:STRING

    Ans = DOMODAL(InpDia)

    IF (Ans = @IDOK)
        IF (DirName <> "")
            DirPath = GETCONTROLTEXT(d1, 9)
            DirPath = DirPath + "\" + DirName
            error  = CREATEDIR(DirPath)
        ENDIF

' Use the Sub GetLocal Folder to update the contents of the Folder
        GetLocalFolder

    ENDIF

RETURN
```

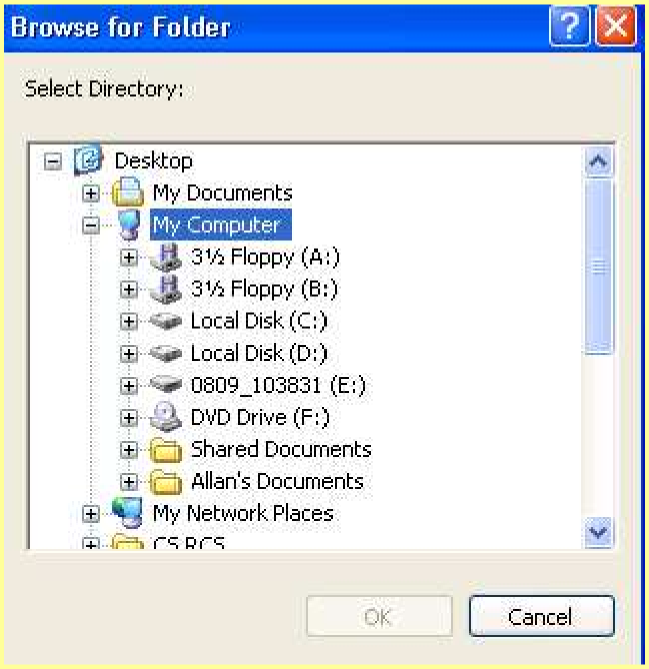
The SelectFolder Sub allows you to change to a different Folder via the Folder Browser.

Ch Dir

```
---------------------SELECT FOLDER --
----FOR CHANGE FOLDER LOCAL
SUB SelectFolder
    DEF Buffer:STRING
    DEF bi:BROWSEINFO
    DEF lpIDList:INT
```

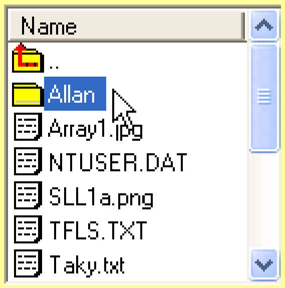
IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2



```
    bi.hOwner = d1
    bi.pszDisplayName = Buffer
    bi.lpszTitle = "Select Directory:"
    bi.ulFlags = 0x00000001
    lpIDList = SHBrowseForFolder(bi)
    IF (lpIDList <> 0)
        pszPath =
SHGetPathFromIDList(lpIDList,Buffer)
        CoTaskMemFree(lpIDList)
    ENDIF
    IF (Buffer <> "")
        ' Set Edit Box with Folder Address
        SETCONTROLTEXT (d1, 9, Buffer)
    ENDIF
RETURN
```

The GetLocalFolder Sub reads the Selected Folder into the List View File List. When the Folder is searched Icons are added to each Item in the List View.
There are three different Icons: one for an Up (Parent Folder), one for Folders, one for Files.
The Icons were loaded into the Image List and inserted into the List View Controls Icon List when FTP NOW! was started.



1.	Set Cursor **WAIT**
2.	Get **Path** from **Edit**

```
--------------------------------READ DIRECTORY - LOCAL-----
SUB GetLocalFolder
    DEF error,hSearch,itemno:INT
    DEF SearchPath,DrNm:STRING

    SETCURSOR (d2, @CSWAIT)
    SearchPath = GETCONTROLTEXT (d1, 9)
    itemno = 0
    ' Read files, folders into LV [10]
    hSearch = FINDOPEN(SearchPath + "\*.*")
    IF (hSearch)
        CONTROLCMD d1, 10, @LVDELETEALL  'Clear LV
[10]
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	Box (ID 9)
3.	**FINDOPEN** Path
4.	Empty **List View**
5.	Set up **DO UNTIL** Function with **FINDNEXT**
6.	**Check for Folder**
7.	Insert Item into **List View**
8.	**SendMessage** to get Item from **lvi Variable**
9.	**SendMessage** to set **lvi** with the **Folder Icon**
10.	Increment **itemno**
11.	**Check for Files**
12.	Insert Item into **List View**
13.	**SendMessage** to get Item from **lvi Variable**
14.	**SendMessage** to set **lvi** with the **Folder Icon**
15.	Increment **itemno**
16.	**FINDCLOSE** Closes Search.
17.	Set Cursor **ARROW**

```
        DO
           DrNm = FINDNEXT(hSearch)
          IF (DrNm = "..")
             CONTROLCMD
d1,10,@LVINSERTITEM,itemno,DrNm
             CONTROLCMD d1,10,@LVSETTEXT,itemno,1,"6"
               lvi.mask = LVIF_IMAGE
               lvi.iItem = itemno
               lvi.iSubItem = 0

SendMessageA(hwndListViewLoc,@LVM_GETITEMA,0,lvi)
               lvi.iImage = 0

SendMessageA(hwndListViewLoc,@LVM_SETITEMA,0,lvi)
               itemno = itemno + 1
          ENDIF
          IF (DrNm <> ".") & (DrNm <> "..") & (DrNm <> "")
             IF (GetFileAttributesA(SearchPath + "\" + DrNm)
&       [Next Line]
                    @FILE_ATTRIBUTE_DIRECTORY)
                 ' Save as DIR to LV [10]
                CONTROLCMD
d1,10,@LVINSERTITEM,itemno,DrNm
                CONTROLCMD
d1,10,@LVSETTEXT,itemno,1,"16"
                  lvi.mask = LVIF_IMAGE
                  lvi.iItem = itemno
                  lvi.iSubItem = 0

SendMessageA(hwndListViewLoc,@LVM_GETITEMA,0,lvi)
                  lvi.iImage = 1

SendMessageA(hwndListViewLoc,@LVM_SETITEMA,0,lvi)
                  itemno = itemno + 1
            ELSE
                ' Save as File to LV [10]
                CONTROLCMD
d1,10,@LVINSERTITEM,itemno,DrNm
                CONTROLCMD
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
d1,10,@LVSETTEXT,itemno,1,"0"
                lvi.mask = LVIF_IMAGE
                lvi.iItem = itemno
                lvi.iSubItem = 0

SendMessageA(hwndListViewLoc,@LVM_GETITEMA,0,lvi)
                lvi.iImage = 2

SendMessageA(hwndListViewLoc,@LVM_SETITEMA,0,lvi)
                itemno = itemno + 1
          ENDIF
        ENDIF
      UNTIL DrNm = ""
      ' Close the Search for Files and Folders Function
      FINDCLOSE hSearch
   ELSE
      MessageBeep(0)
   ENDIF
   SETCURSOR (d2, @CSARROW)
RETURN
```

The DeleteDirLocal Sub allows you to Select a Folder in the LOCAL List View and Delete it.

Del Dir

1.	Get number of **items** in **List View**
2.	Create **WHILE** loop to iterate through the **List View**
3.	Check if **item** is **selected**
4.	🗀 **If Attrib = 16**

```
--------------------------------DELETE FOLDER - LOCAL---------
SUB DeleteDirLocal
   DEF count,item,selected,Ans,error:INT
   DEF text,DirPth,Attrib:STRING

   item = 0
   count = CONTROLCMD( d1, 10, @LVGETCOUNT)

   WHILE (item <= count)
      selected = CONTROLCMD( d1, 10, @LVGETSELECTED,
item)
      text = CONTROLCMD( d1, 10, @LVGETTEXT, item, 0 )
      Attrib = CONTROLCMD(d1,10,@LVGETTEXT,item,1)
      IF (selected > 0)
       IF (Attrib = "16")
            IDLabel = "Delete This Folder"
            IDName = text
            Ans = DOMODAL(InpDia)
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	Run **InpDia** Dialog
5.	If **InpDia** returns an **@IDOK** get the **DirPth** and add the **selected Folder** to the **DirPth** string
6.	**REMOVEDIR DirPth**
7.	If any errors then display **Messagebox**
8.	Increment **item** to iterate through the **List View.**
9.	After **WHILE** Loop finished then **GetLocalFolder**
10.	**List View File List** is now updated with deleted Folders removed

```
            IF (Ans = @IDOK)
                  DirPth = GETCONTROLTEXT(d1,9)
                  DirPth = DirPth + "\" + text
                  error = REMOVEDIR(DirPth)
               IF (error = 0)
                   MESSAGEBOX d1,
[Next Line]
                       "Selected Folder MUST be EMPTY of any FILES!",
"Delete Foler"
               ENDIF
            ENDIF
              IDLabel = "Name"
              IDName = ""
         ELSE
   MESSAGEBOX d1, "Selected Item NOT a Folder!", "Delete Folder"
         ENDIF
        ENDIF
        item = item + 1
    ENDWHILE

' Use the Sub GetLocal Folder to update the contents of the Folder
    GetLocalFolder

RETURN
```

RenameFileLocal Sub is used to Rename a File.

1.	Get number of **items** in the **List View**
2.	Create a **WHILE** Loop to iterate through the **List View.**
3.	If **item** is **selected**

```
-----------------------------------RENAME FILE- LOCAL -------
SUB RenameFileLocal
    DEF count,item,selected,Ans,error:INT
    DEF text,DirPth,Attrib,source,dest:STRING

    item = 0
    count = CONTROLCMD( d1, 10, @LVGETCOUNT)

' Iterate through the List View to check each item for selected
state
    WHILE (item <= count)
        selected = CONTROLCMD( d1, 10,
@LVGETSELECTED, item)
    text = CONTROLCMD( d1, 10, @LVGETTEXT, item, 0 )
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	and **Attrib = 0** then a **File** is selected
4.	Set **Static Text** Control and **Edit Box** (in **InpDia**) to show the File to be Renamed
5.	Run **InpDia** to confirm the **Rename** of the File
6.	Add **text** to **DirPth** to get **source**. Add **DirName** (from the **InpDia**) to **DirPth** to get **dest**.
7.	**COPYFILE** **source** to **dest**
8.	If no **error** then **DELETEFILE** **source** File
9.	Set **Static Text** and **Edit Box** (**InpDia**) to original text.
10.	**GetLocalFolder** to Update Folder with changed File Names

```
Attrib = CONTROLCMD( d1, 10, @LVGETTEXT, item, 1 )
      ' If item is selected and is a File (Attrib = 0)
      IF ((selected) & (Attrib = "0"))
          IDLabel = "Rename File " + text + " to . . ."
          IDName = ""
          Ans = DOMODAL(InpDia)
        IF (Ans = @IDOK)
            IF (DirName <> text)
                DirPth = GETCONTROLTEXT(d1,9)
                source = DirPth + "\" + text
                dest = DirPth + "\" + DirName
                fail = 1        :'Not to overwrite if file exists
                error = COPYFILE(source, dest, fail)
              IF (error) THEN error = DELETEFILE(source)
            ELSE
MESSAGEBOX d1, "Filename Already Exists!", "Rename File"
            ENDIF
        ENDIF
          ' Reset IDLabel and IDName to original settings
          IDLabel = "Name"
          IDName = ""
      ENDIF
      ' Increment item so as to go to next item in List View
      item = item + 1
    ENDWHILE

' Use the Sub GetLocal Folder to update the contents of the
Folder
    GetLocalFolder

RETURN
```

DeleteFileLocal
Sub allows you to select File/s to be Deleted from the Folder.

```
-----------------------------------DELETE FILE - LOCAL---------
SUB DeleteFileLocal
    DEF count,item,selected,Ans,error:INT
    DEF text,DirPth,Attrib:STRING

    item = 0
    count = CONTROLCMD( d1, 10, @LVGETCOUNT)
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

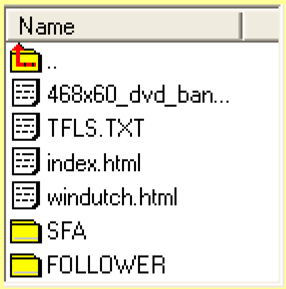
1.	Get number of **items** in the **List View**
2.	Create a **WHILE** Loop to iterate through the **List View**
3.	If **item** is **selected** and **Attrib = 0** then a **File** is selected
4.	Set **Static Text** Control and **Edit Box** (in **InpDia**) to show the File to be Deleted
5.	Run **InpDia** to confirm the **Delete** of the File.
6.	Add **text** to **DirPth** to get Path of File to Delete.
7.	**DELETEFILE** DirPth
8.	Set **Static Text** and **Edit Box** (**InpDia**) to original text.
9.	**GetLocalFolder** to Update Folder without deleted File Names

```
' Iterate through the List View to check each item for selected state
   WHILE (item <= count)
 selected = CONTROLCMD( d1, 10, @LVGETSELECTED, item)
      text = CONTROLCMD( d1, 10, @LVGETTEXT, item, 0 )
    Attrib = CONTROLCMD( d1, 10, @LVGETTEXT, item, 1 )

      ' If item is selected and is a File (Attrib = 0)
     IF ((selected) & (Attrib = "0"))
         IDLabel = "Delete This File"
         IDName = text
         Ans = DOMODAL(InpDia)
       IF (Ans = @IDOK)
           DirPth = GETCONTROLTEXT(d1,9)
           DirPth = DirPth + "\" + text
           error = DELETEFILE(DirPth)
       ENDIF
         ' Reset IDLabel and IDName to original settings
         IDLabel = "Name"
         IDName = ""
     ENDIF

      ' Increment item so as to go to next item in List View
     item = item + 1
   ENDWHILE

' Use the Sub GetLocal Folder to update the contents of the Folder
   GetLocalFolder

RETURN
```

DClkChDirRem Sub is used when you Double Click on a Folder in the Remote List View. This bypasses the need to click the Ch Dir **(ChDir) Button.**

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2


Name
..
468x60_dvd_ban...
TFLS.TXT
index.html
windutch.html
SFA
FOLLOWER

1.	Get **itemno** in **Remote List View** (**ID 20**) from **lv** UDT
2.	Get **Folder Name** into **DirNm** string
3.	Get **Attrib** of the **itemno**
4.	**OPTION ONE** If **Attrib** is equal to **'6'**
5.	Get Folder Path into **SearchPath** string
6.	Find last **"/"** in Path and use **LEFT$** to remove **last Folder** from **SearchPath**
7.	If **Paths** is an empty string then set **Paths** to **"/"**
8.	**FtpSetCurrentDirectoryA** to the new **Paths** string
9.	**GetRemoteFolder** to update **List View**
10.	**OPTION TWO** If **Attrib = 16** Get **SearchPath** and add **DirNm**
11.	Set **Edit Box (ID 19)** in **Remote** to the **SearchPath**

DOUBLE CLICK - CHANGE DIRECTORY REMOTE

```
SUB DClkChDirRem
    DEF itemno,Fnd,loc:INT
    DEF Attrib,SearchPath,Paths,DirNm:STRING

' Get itemno from the lv UDT - This is the Item you Double Clicked
   itemno = lv.iItem
   DirNm =
CONTROLCMD(d1,20,@LVGETTEXT,itemno,0)
   Attrib =
CONTROLCMD(d1,20,@LVGETTEXT,itemno,1)
   ' If Attrib is up one level in the Folder Tree
    IF (Attrib = "6")
        SearchPath = GETCONTROLTEXT (d1, 19)
        Fnd = 1
        pos = 1
' Get new Path by removing Last Folder from Current Path in ID 19
        DO
         IF (INSTR(pos, SearchPath, "/") > 0) THEN loc
= pos ELSE Fnd = 0
             pos = pos + 1
        UNTIL Fnd = 0
        Paths = LEFT$(SearchPath, loc-1)
        ' If Paths empty set to Root Directory "/"
        IF (Paths = "") THEN Paths = "/"
         SETCONTROLTEXT (d1, 19, Paths)
          ' Use API in Wininet.dll to Set Current Folder
        error = FtpSetCurrentDirectoryA(hconnect, Paths)
' Use Sub to get contents of the new Folder in List View (ID 20)
           GetRemoteFolder
       ENDIF
       ' Check if Selected item is a Folder
       IF (Attrib = "16")
           SearchPath = GETCONTROLTEXT (d1, 19)
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

12.	**FtpSetCurrentDirectoryA** with the **SearchPath**
13.	**GetRemoteFolder** to update **List View File List**

```
        SETCONTROLTEXT (d1, 19, SearchPath)
          ' Use API in Wininet.dll to Set Current Folder
  error = FtpSetCurrentDirectoryA(hconnect, SearchPath)

' Use Sub to get contents of the new Folder in List
View (ID 20)
          GetRemoteFolder
      ENDIF

RETURN
```

MakeDirRem Sub is called when you want to create a New Folder on the Web Site.

Mk Dir

1.	Call **InpDia** Dialog
2.	Obtain **New Folder Name** from **InpDia**
3.	If **@IDOK** then **FtpCreateDirectoryA** with **DirName**
4.	If new folder created **OK** then **Update** the **Remote List View Folder** by calling **GetRemoteFolder**

```
--------------------MAKE NEW DIRECTORY -
REMOTE

SUB MakeDirRem
   DEF ans:INT

   ' Run InpDia to get the Name of the New Folder
   Ans = DOMODAL(InpDia)

   IF (Ans = @IDOK)
' Call FtpCreateDirectory from Wininet.dll to Create
New Folder
         Ans = FtpCreateDirectoryA(hconnect, DirName)

' If Successful then Update the Remote Folder in the
List View [ID 20]
        IF (Ans) THEN GetRemoteFolder

   ENDIF

RETURN
```

ChangeDirRemote Sub is called when you Select a Folder in the REMOTE List View and then Click the Ch Dir Button.

```
-------------------CHANGE DIRECTORY - REMOTE
SUB ChangeDirRemote
   DEF count,item,selected,Ans,error:INT
   DEF text,Attrib,MyPath:STRING

   ' Get the number of items (count) in the List View
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

1.	Get number of **items** in **List View** (**ID 20**)
2.	Create **WHILE** Loop to iterate through the **List View**
3.	Check if **item** is **selected**
4.	Check if **Attrib = "16"**
5.	Set **InpDia IDLabel**
6.	Set **IDName** with content of **text**
7.	Run **InpDia** to get the **New Folder Name**
8.	If **@IDOK** from **InpDia** then get **MyPath** from **Edit Box** (**ID 19**) and add **text** to **MyPath**
9.	Call the **Wininet.dll FtpSetCurrentDirectory Function** to set the **New Folder** to the **MyPath** string
10.	If **error** > 0 (means no error found) creating new Folder set **item** to **count** + 10 to exit the **WHILE** loop

```
    item = 0;
  count = CONTROLCMD( d1, 20, @LVGETCOUNT)

' Set up a WHILE Loop to iterate through the List View
    WHILE (item <= count)
selected = CONTROLCMD( d1, 20, @LVGETSELECTED, item)
text = CONTROLCMD( d1, 20, @LVGETTEXT, item, 0 )
Attrib = CONTROLCMD(d1, 20, @LVGETTEXT, item, 1)

       ' Check if item is selected and text is valid
       IF ((selected > 0) & (Attrib = "16"))
          ' Set InpDia Label and Edit Box
          IDLabel = "Change To This Folder"
          IDName = text
' Run InpDia Dialog to get the Folder Name to change to
          Ans = DOMODAL(InpDia)

         IF (Ans = @IDOK)
             MyPath = GETCONTROLTEXT(d1,19)
             MyPath = MyPath + text
' Use Wininet.dll Function to Set Current Directory to MyPath
             error = FtpSetCurrentDirectoryA(hconnect, MyPath)
            IF (error > 0)
               item = count + 10
            ENDIF
         ENDIF

         ' Set InpDia Controls back to original content
          IDLabel = "Name"
          IDName = ""
       ENDIF
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

11.	Set **InpDia** Controls to original settings
12.	**GetRemoteFolder** to Update the **List View** Folder contents

```
        ' Increment item to iterate through List View
        item = item + 1
    ENDWHILE

' If finished Update Remote List View with Folder Contents
' If item > count + 4 then Change Folder has been Accepted
    IF (item > count + 4)  THEN GetRemoteFolder
RETURN
```

DelDirRemote Sub is used when you select a Folder in the Remote List View and then Click the Del Dir Button.

Del Dir

1.	Get number of **items** in **List View** (**ID 20**)
2.	Create **WHILE** Loop to iterate through the **List View**
3.	Check if **item** is **selected**
4.	Check if **Attrib = "16"**
5.	Set **InpDia IDLabel**
6.	Set **IDName** with content of **text**
7.	Run **InpDia** to check the **Remove Folder Name**
8.	If **@IDOK** from **InpDia** then **Remove**

```
----------------REMOVE DIRECTORY - REMOTE
SUB DelDirRemote
    DEF count,item,selected,Ans,error:INT
    DEF text,Attrib,MyPath:STRING

    ' Get the number of items (count) in the List View
    item = 0
  count = CONTROLCMD( d1, 20, @LVGETCOUNT)

' Set up a WHILE Loop to iterate through the List View
    WHILE (item <= count)
selected = CONTROLCMD( d1, 20,
@LVGETSELECTED, item)
text = CONTROLCMD( d1, 20, @LVGETTEXT,
item, 0 )
Attrib = CONTROLCMD(d1, 20, @LVGETTEXT,
item, 1)
        ' Check if item is selected and text is valid
        IF ((selected > 0) & (Attrib = "16"))
            ' Set InpDia Label and Edit Box
            IDLabel = "Delete Directory"
            IDName = text
' Run InpDia Dialog to check the Folder Name to Remove
            Ans = DOMODAL(InpDia)
          IF (Ans = @IDOK)
' Use Wininet.dll Function to Set Remove Directory in
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	Folder
9.	Call the **Wininet.dll FtpRemoveDirectory Function** to **delete** the **Folder** in **text**
10.	If **error** > 0 (means no error) removing the Folder set **item** to **count** + 10 to exit the **WHILE** loop
11.	Set **InpDia** Controls to original settings
12.	**GetRemoteFolder** to Update the **List View** Folder contents

```
text
            error = FtpRemoveDirectoryA(hconnect, text)
            IF (error > 0) THEN item = count + 10
            IF (error = 0)
            MESSAGEBOX d1, "Selected Folder MUST be EMPTY of any FILES!", "Delete Folder"
            ENDIF
        ENDIF
        ' Set InpDia Controls back to original content
          IDLabel = "Name"
          IDName = ""
      ENDIF
      ' Increment item to iterate through List View
      item = item + 1
  ENDWHILE

' If finished Update Remote List View with Folder Contents
' If item > count + 4 then Remove Folder has been Accepted
  IF (item > count + 4) THEN GetRemoteFolder

RETURN
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

RenameFileRem Sub is active when you select a File in the List View and then Click the Ren File Button.

Ren File

1.	Get number of **items** in **List View** (**ID 20**)
2.	Create **WHILE** Loop to iterate through the **List View**
3.	Check if **item** is **selected**
4.	Check if Attrib is equal to "0"
5.	Set **InpDia IDLabel**
6.	Run **InpDia** to input the **Rename File Name**
7.	If **@IDOK** from **InpDia** then **Rename File**
8.	Call the **Wininet.dll FtpRenameFile Function** to Rename the File in **text** to the **File** in **DirName**
9.	Set **InpDia** Controls to original settings
10.	**GetRemoteFolder** to Update the contents of the **List View**

```
------------------------RENAME FILE - REMOTE
SUB RenameFileRem
   DEF count,item,selected,Ans,error:INT
   DEF text,Attrib:STRING

    ' Get the number of items (count) in the List View
    item = 0
    count = CONTROLCMD( d1, 20,
@LVGETCOUNT)

' Set up a WHILE Loop to iterate through the List
View
    WHILE (item <= count)
        selected = CONTROLCMD( d1, 20,
@LVGETSELECTED, item)
        text = CONTROLCMD( d1, 20,
@LVGETTEXT, item, 0 )
        Attrib = CONTROLCMD( d1, 20,
@LVGETTEXT, item, 1 )
' Check if item is selected and Attrib is equal to "0"
(File)
        IF ((selected > 0) & (Attrib = "0"))
            ' Set InpDia Label and Edit Box
            IDLabel = "Rename File " + text + " to . . ."
            IDName = ""
' Run InpDia Dialog to get the name to Rename File to
            Ans = DOMODAL(InpDia)
          IF (Ans = @IDOK)
             IF (DirName <> text)
' Use Wininet.dll Function to Rename File in text to
DirName
        error = FtpRenameFileA(hconnect, text, DirName)
             ELSE
MESSAGEBOX d1, "Filename Already Exists!",
"Rename File"
             ENDIF
         ENDIF
          ' Set InpDia Controls back to original content
            IDLabel = "Name"
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
            IDName = ""
        ENDIF
        ' Increment item to iterate through List View
        item = item + 1
    ENDWHILE

' If finished Update Remote List View with changed
Folder Contents
    GetRemoteFolder

RETURN
```

DeleteFileRem Sub is activated when you select a File in the Remote List View and Click on the Del File Button.

1.	Get number of **items** in **List View** (**ID 20**)
2.	Create **WHILE** Loop to iterate through the **List View**
3.	Check if **item** is **selected**
4.	Check if **text** is not equal to **".."**
5.	Set **InpDia IDLabel**
6.	Run **InpDia** to check

```
----------------------DELETE FILE - REMOTE
SUB DeleteFileRem
    DEF count,item,selected,Ans,error:INT
    DEF text,MyPath:STRING

    ' Get the number of items (count) in the List View
    item = 0
    count = CONTROLCMD( d1, 20,
@LVGETCOUNT)

' Set up a WHILE Loop to iterate through the List
View
    WHILE (item <= count)
        selected = CONTROLCMD( d1, 20,
@LVGETSELECTED, item)
        text = CONTROLCMD( d1, 20,
@LVGETTEXT, item, 0 )

' Check if item is selected and Attrib is not equal to
".." (File)
        IF ((selected) & (text <> ".."))
            ' Set InpDia Label and Edit Box
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

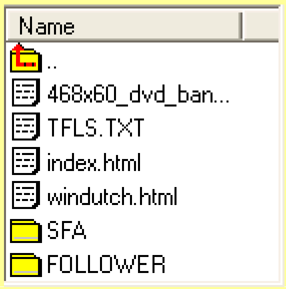
	the **Delete File Name**
7.	If **@IDOK** from **InpDia** then **Delete File**
8.	Call the **Wininet.dll FtpDeleteFile Function** to Delete the File in **text**
9.	Set **InpDia** Controls to original settings
10.	**GetRemoteFolder** to Update the contents of the **List View**

```
            IDLabel = "Delete File "
            IDName = text
' Run InpDia Dialog to check the name of Delete File
            Ans = DOMODAL(InpDia)
          IF (Ans = @IDOK)
          ' Use Wininet.dll Function to Delete File in text
               error = FtpDeleteFileA(hconnect, text)
          ENDIF
            ' Set InpDia Controls back to original content
            IDLabel = "Name"
            IDName = ""
       ENDIF

       ' Increment item to iterate through List View
       item = item + 1
    ENDWHILE

' If finished Update Remote List View with changed Folder Contents
    GetRemoteFolder

RETURN
```

GetRemoteFolder Sub is used whenever you wish to update the Files/Folders in the REMOTE List View.
We use three main APIs in the Wininet.dll to achieve the task. They are: FtpGetCurrentDirectoryA, FtpFindFirstFileA and InternetFindNextFileA. Also Icons are added to the List View according to the File/Folder found. Additionally we add a Folder pointing to the Parent Folder (📁).

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2



Name
..
468x60_dvd_ban...
TFLS.TXT
index.html
windutch.html
SFA
FOLLOWER

1.	**ALLOCMEM** for **mData**
2.	Set Cursor to **WAIT**
3.	Use the **Wininet.dll** with **FtpGetCurrent-Directory** to get **Current Folder** on **Web Site**
4.	Set **Edit Box** (**ID 19**) in **Remote Group** to the Current Folder Path.
5.	Call **FtpFindFirstFileA** to find **First File**
6.	Read **mData** in Memory into **pData** UDT
7.	Clear the **List View** (**ID 20**)
8.	**FIRST ENTRY in the LIST VIEW** Insert **".."** into **List View** to make **Parent Folder** available. **itemno** = 0 as 0 is first entry in **List View**.
9.	Set **lvi** UDT to **itemno** used for **".."**. This gets the **List View** data into **lvi** for this entry.
10.	Get **lvi** data for **itemno**

```
------------------------GET FILES FOR REMOTE FOLDER----
SUB GetRemoteFolder
   DEF pData:WIN32_FIND_DATA
   DEF mData:MEMORY
   DEF hFind,itemno:INT
   DEF lRet:INT
   DEF lpdw:UINT
   DEF Tmp,path:STRING

   lpdw = 254
   ALLOCMEM mData,1,LEN(pData)
   SETCURSOR (d2, @CSWAIT)
   path = ""
   FtpGetCurrentDirectoryA(hconnect, path, lpdw)  :' Get Current
Folder
   SETCONTROLTEXT d1,19,path
   itemno = 0
   pData.cFileName = STRING$(259, 0)
   hFind = FtpFindFirstFileA(hconnect, "*.*", mData, 0, 0)      :'
Find the first file
   READMEM mData,1,pData
   CONTROLCMD d1,20,@LVDELETEALL    :' Clear LV [20]
   CONTROLCMD d1,20,@LVINSERTITEM,itemno,".." :' .. in
List View
   CONTROLCMD d1,20,@LVSETTEXT,itemno,1,"6"
   lvi.mask = LVIF_IMAGE
   lvi.iItem = itemno
   lvi.iSubItem = 0
   SendMessageA(hwndListViewRem,@LVM_GETITEMA,0,lvi)
   lvi.iImage = 0
   SendMessageA(hwndListViewRem,@LVM_SETITEMA,0,lvi)
   itemno = itemno + 1
   IF (hFind <> 0)                :' if there's no file, then exit sub
      'put the filename in List View
      IF (pData.dwFileAttributes =
@FILE_ATTRIBUTE_DIRECTORY)
         CONTROLCMD
d1,20,@LVINSERTITEM,itemno,pData.cFileName
         CONTROLCMD d1,20,@LVSETTEXT,itemno,1,"16"
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	by **LVM_GETITEMA**
11.	📁Set **lvi.iImage** to 0 - which is the first **Icon** in the **Image List**.
12.	Save **lvi** with **Icon** with **LVM_SETITEMA** Now the **List View** we have added has the **Icon**.
13.	Increment **itemno** for the next Item to go into the **List View.**
14.	**SECOND ENTRY** If **Attribute** is **Directory** insert into **List View** as a **Directory** else insert as a **File**
15.	Insert **pData.cFileName** into **List View** to make **Folder** or **File** available.
16.	Set **lvi** UDT to the **itemno**
17.	Get **lvi** data for **itemno**
18.	📁Set **lvi.iImage** to **1** if **Attribute** is **Directory** or 📄 Set **lvi.iImage** to **2** if a **File**
19.	Save **lvi** with **Icon**
20.	**REPEAT ENTRIES** Set **DO** Loop to iterate through rest of Folder
21.	Use the **Wininet API** **InternetFindNextFileA** to locate any more **Sub Folders** or **Files** in the Current Folder Path
22.	Insert them into the

```
        lvi.mask = LVIF_IMAGE
        lvi.iItem = itemno
        lvi.iSubItem = 0

SendMessageA(hwndListViewRem,@LVM_GETITEMA,0,lvi)
        lvi.iImage = 1

SendMessageA(hwndListViewRem,@LVM_SETITEMA,0,lvi)
        itemno = itemno + 1
      ELSE
        CONTROLCMD
d1,20,@LVINSERTITEM,itemno,pData.cFileName
        CONTROLCMD d1,20,@LVSETTEXT,itemno,1,"0"
        lvi.mask = LVIF_IMAGE
        lvi.iItem = itemno
        lvi.iSubItem = 0

SendMessageA(hwndListViewRem,@LVM_GETITEMA,0,lvi)
        lvi.iImage = 2

SendMessageA(hwndListViewRem,@LVM_SETITEMA,0,lvi)
        itemno = itemno + 1
      ENDIF
      lRet = 1   :' Set for DO UNTIL
      DO
        pData.cFileName = STRING$(259, 0)
        lRet = InternetFindNextFileA(hFind, mData)      :' Find the
next file
        READMEM mData,1,pData
        IF (lRet <> 0)                          : ' If  no next file, exit do
           ' Put the filename in List View [20]
         IF (pData.dwFileAttributes =
@FILE_ATTRIBUTE_DIRECTORY)
            CONTROLCMD d1,20,@LVINSERTITEM,
                        itemno,pData.cFileName
            CONTROLCMD d1,20,@LVSETTEXT,itemno,1,"16"
            lvi.mask = LVIF_IMAGE
            lvi.iItem = itemno
            lvi.iSubItem = 0
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

	Remote List View, then . . .
23.	**Update** the **List View** with the correct **Icon**.
24.	After all the **Folders** and **Files** are processed in the DO Loop then use **InternetCloseHandle** Function to close the **FindFile** Function.
25.	Set Cursor **ARROW**
26.	**FREEMEM mData** to free the memory you originally allocated.

```
SendMessageA(hwndListViewRem,@LVM_GETITEMA,0,lvi)
            lvi.iImage = 1

SendMessageA(hwndListViewRem,@LVM_SETITEMA,0,lvi)
         ELSE
           CONTROLCMD d1,20,@LVINSERTITEM,
                        itemno,pData.cFileName
            CONTROLCMD d1,20,@LVSETTEXT,itemno,1,"0"
            lvi.mask = LVIF_IMAGE
            lvi.iItem = itemno
            lvi.iSubItem = 0

SendMessageA(hwndListViewRem,@LVM_GETITEMA,0,lvi)
            lvi.iImage = 2

SendMessageA(hwndListViewRem,@LVM_SETITEMA,0,lvi)
            itemno = itemno + 1
         ENDIF
       ENDIF
     UNTIL (lRet = 0)
       InternetCloseHandle hFind        : 'Close the search handle
   ENDIF
   SETCURSOR (d2, @CSARROW)
   FREEMEM mData
RETURN
```

FTP NOW! Program is a good example of how simple IBasic makes it to code powerful Internet Applications for Windows. A few of the SUBroutines in the program could be made into one SUB with parameters, however, the Code has been written in this style to make it simple to understand and follow. To improve the program more you could add more Code, for example: Error Checking with MessagBox notifying errors, if desired. The Code in this Article, **Part II**, added to the Code in **Part I** makes the complete program.

For your convenience the full working program and the FTPNow.iba file and other associated files are available in zip format. - ***Bizzy***

IBasic Monthly
Dedicated to everything IBasic!

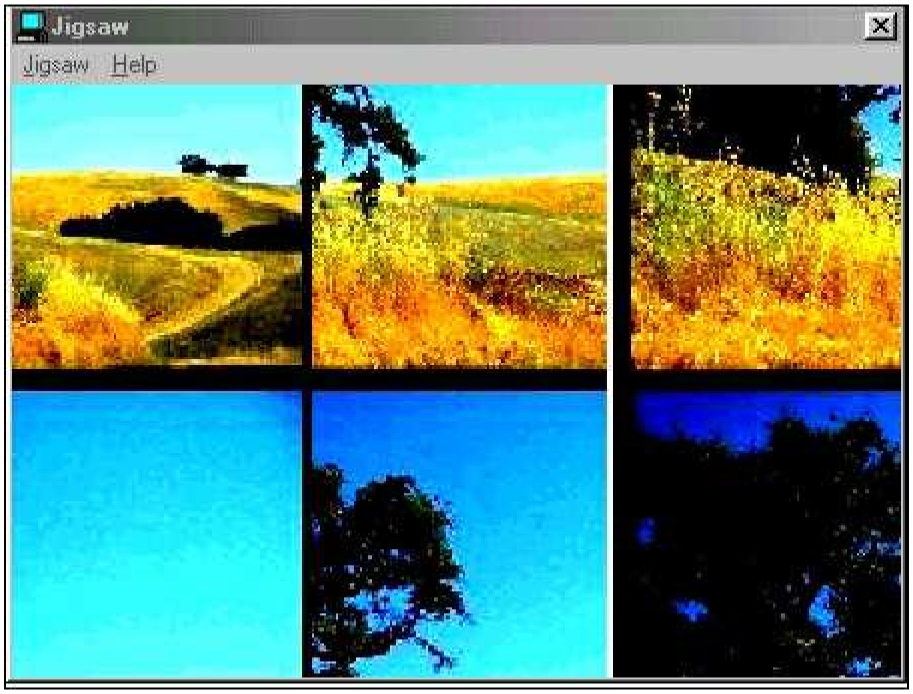
December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Of Jigsaw Puzzles, Mice And Me

By

Paul Love (pel)

A few weeks ago, I put together a quick program that works like a simple sliding block puzzle:



You just click on the piece you want to move, then click on the square you want to move it to; the program switches the two pieces around and eventually you wind up with the original picture. This sample program only uses six pieces, but of course the more you use, the more challenging the puzzle. There are a number of programs that can take an image and split it into multiple pieces for you -- Adobe Photoshop for instance, although I used something called "Dicer" (which is freeware) to make these pieces. Each of the images is 150 x 150, but if you change the values for w and h in the "Initialization" subroutine, you can scale them up or down (since they're all loaded with the @IMGSCALABLE property).

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

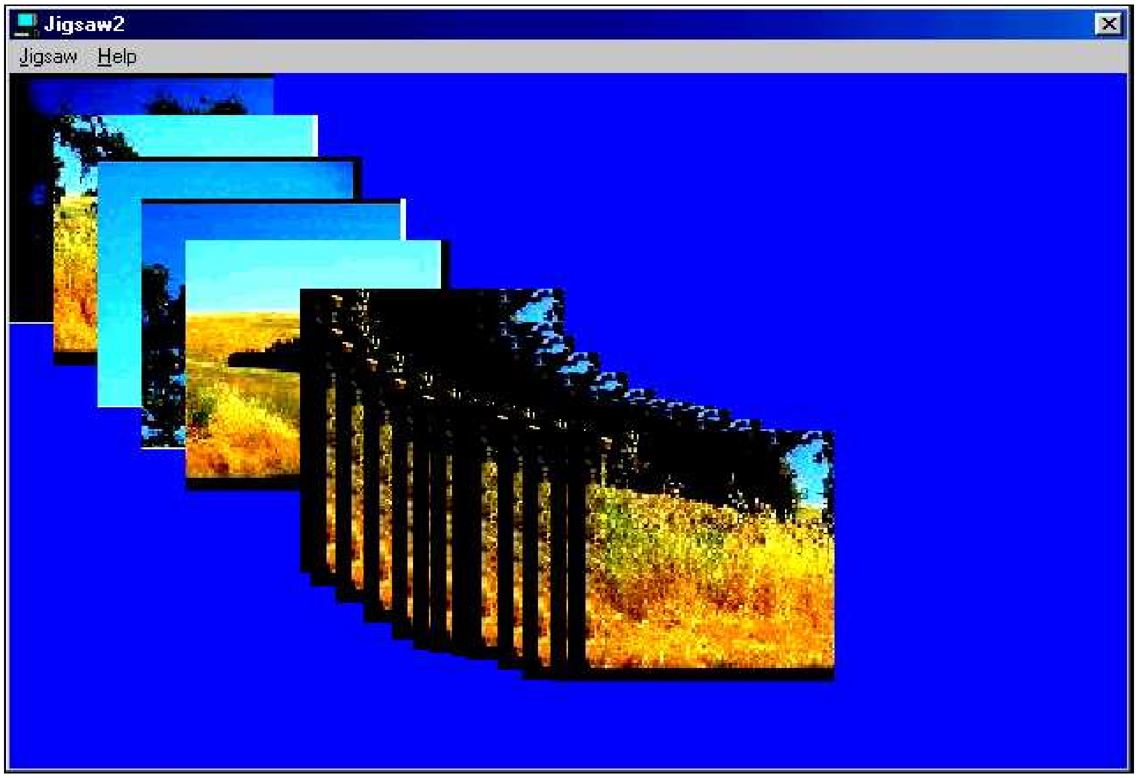
```
rem Jigsaw1

DEF win:WINDOW

DEF bitmapz[6]:INT
def bitmap1,bitmap2:INT
def cellbmp[2,3]:INT

DEF i,j,k,i1,j1,i2,j2,k1,k2,mi,mj,count:INT
DEF x,y,w,h:INT
DEF xx,hi,hj,answer:INT
DEF path$,null$:STRING

path$ = GETSTARTPATH
bitmapz[0] = LoadImage (path$+"scen000.bmp",@IMGSCALABLE)
bitmapz[1] = LoadImage (path$+"scen001.bmp",@IMGSCALABLE)
bitmapz[2] = LoadImage (path$+"scen002.bmp",@IMGSCALABLE)
bitmapz[3] = LoadImage (path$+"scen003.bmp",@IMGSCALABLE)
bitmapz[4] = LoadImage (path$+"scen004.bmp",@IMGSCALABLE)
bitmapz[5] = LoadImage (path$+"scen005.bmp",@IMGSCALABLE)

WINDOW win,0,0,450,340,@CAPTION|@SYSMENU,0,"Jigsaw",mainwindow
menu win,"T,&Jigsaw,0,0","I,Quit,0,3"
insertmenu win,1,"T,&Help,0,0","I,Contents,0,11","I,About,0,12"

icon1 = LoadImage (path$+"pc4.ico",@IMGICON)
seticon win,icon1

gosub initialize

' build jigsaw window
   k = 5
   for i = 0 to 1
      for j = 0 to 2
         x = j * w
         y = i * h
         showimage win,bitmapz[k],@IMGSCALABLE,x,y,w,h
         cellbmp[i,j] = k
         k = k - 1
       next j
    next i

run=1

waituntil run=0
closewindow win

end
```

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

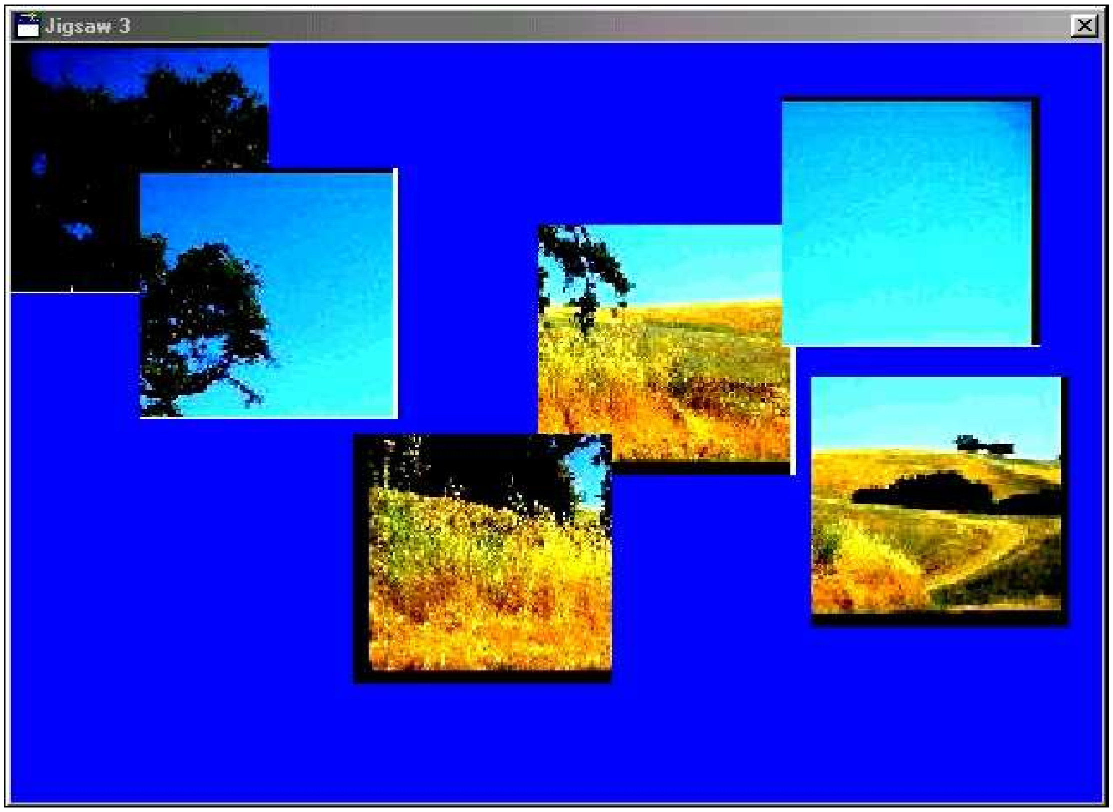
```
mainwindow:
select @CLASS
        CASE @IDCLOSEWINDOW
           for i = 0 to 5
              deleteimage bitmapz[i],@IMGSCALABLE
           next i
           run=0

      CASE @IDCREATE
           CENTERWINDOW win

        case @IDMENUPICK
         select @MENUNUM
            case 3
               for i = 0 to 5
                 deleteimage bitmapz[i],@IMGSCALABLE
               next i
               run = 0
            case 11
               showwindow win,@SWHIDE
               answer = domodal d1
               showwindow win,@SWRESTORE
            case 12
               messagebox win,"This is an example of a simple jigsaw puzzle
game","Jigsaw",64
         endselect

        CASE @IDLBUTTONDN
           mi = @mousey
           mj = @mousex
           i = ceil(mi / h)-1
           j = ceil(mj / w)-1

           ' make sure that the mouse position is within the puzzle border
           if (i>=0) & (i<=1) & (j>=0) & (j<=2)
             count = count + 1
            ' if count = 1 this is the piece to be moved -- if count = 2 (2nd
click) then this is the square to be switched
            '        with the first piece clicked on.
             if count = 1
                i1 = i:  j1 = j
             endif
             if count = 2
                i2 = i:  j2 = j
                k1 = cellbmp[i1,j1]
                k2 = cellbmp[i,j]
                cellbmp[i1,j1] = k2
                cellbmp[i,j] = k1
                x = j1 * w
                y = i1 * h
                showimage win,bitmapz[k2],@IMGSCALABLE,x,y,w,h
                x = j * w
                y = i * h
                showimage win,bitmapz[k1],@IMGSCALABLE,x,y,w,h
                count = 0
             endif
           endif

endselect
return

SUB initialize
   w = 150
   h = 150
   rmax = 2
   cmax = 3
   i = -1
   j = -1
RETURN
```

After messing with this for a little while, I decided that I wanted to create something a little more like a real jigsaw puzzle program, where you can slide the pieces around freely within a window and assemble them that way. The major problem with this approach is that you need to be able to create a visible trail while you're moving a puzzle piece so that you can tell where it is as you're moving it into the position you want. The simplest way to do this (once you've determined which piece has been clicked on) is to keep doing a "showimage" instruction at each point as you drag the mouse along. Once you "drop" the piece (let up on the mouse key) you can redraw the updated window. However, while you're dragging the puzzle piece, the screen gets rather messy with the trail of images of the piece being moved (see below).

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2



After a little experimenting with different ideas (all of which failed miserably) I finally decided to try converting "jigsaw.iba" to a DirectX program (screen shot below):



This version works pretty well as far as simulating true drag and drop moving of images. And (most important for me, since I'm easily confused) the coding for this turned out to be pretty simple and straightforward:

```
REM Jigsaw 3
REM REQUIRES DIRECTX 7.0 OR GREATER

IF GETDXVERSION < 7
      MESSAGEBOX 0,"This program requires" + chr$(13) + "DirectX 7.0 or
greater","Error"
      END
ENDIF

DEF win:WINDOW
DEF state,imageno,drag,w,h,x,y,x1,y1,x2,y2,mi,mj,i,j,k:INT
DEF path$:STRING
DEF spr[7]:INT
DEF pic[7]:STRING
def bmploc[7,2]:INT
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
WINDOW win,0,0,640,480,@NOAUTODRAW,0,"Jigsaw 3",mainwindow
IF CREATESCREEN(win,640,480) < 0
      MESSAGEBOX win, "Could not create DirectX screen","Error"
      CLOSEWINDOW win
      END
ENDIF

' load the sprite images into the pic array
path$ = GETSTARTPATH
pic[1] = path$+"scen000.bmp"
pic[2] = path$+"scen004.bmp"
pic[3] = path$+"scen002.bmp"
pic[4] = path$+"scen001.bmp"
pic[5] = path$+"scen005.bmp"
pic[6] = path$+"scen003.bmp"

'load the sprite(s)
spr[1] = DXSPRITE(win,pic[1],150,150,1)
spr[2] = DXSPRITE(win,pic[2],150,150,1)
spr[3] = DXSPRITE(win,pic[3],150,150,1)
spr[4] = DXSPRITE(win,pic[4],150,150,1)
spr[5] = DXSPRITE(win,pic[5],150,150,1)
spr[6] = DXSPRITE(win,pic[6],150,150,1)

gosub initialize

' build jigsaw window
x = 0:  y = 0
for k = 1 to 6
   DXSETSPRITEDATA spr[k],@SDXPOS,x
   DXSETSPRITEDATA spr[k],@SDYPOS,y
   bmploc[k,0] = x:  bmploc[k,1] = y
   x = x + 25:  y = y + 25
next k
'DXFLIP win

state = 0

run = 1
WAITUNTIL run = 0
CLOSEWINDOW win
END

mainwindow:
SELECT @CLASS
    CASE @IDLBUTTONDN
       ' find the puzzle piece that's just been clicked on (routine
"pickbmp")
```

```
       drag = 1
       mi = @mousey
       mj = @mousex
       gosub pickbmp
       x2 = mj:  y2 = mi

    CASE @IDMOUSEMOVE
       ' if "drag" = 1 (that is, a piece was clicked on and is being dragged)
then calculate the new x and y
       '    coordinates of that piece and move it to the new location
       if drag = 1
          if (imageno > 0) & (imageno < 7)
             mi = @mousey:  mj = @mousex
             mi = mi - (w/2)
             mj = mj - (h/2)
             x1 = mj:  y1 = mi
             if (abs(x1-x2) >= 0) | (abs(y1-y2) >= 0)
                bmploc[imageno,0] = mj:  bmploc[imageno,1] = mi
                DXSETSPRITEDATA(spr[imageno],@SDXPOS,mj)
                DXSETSPRITEDATA(spr[imageno],@SDYPOS,mi)
                DXMOVESPRITE spr[imageno],mj,mi
                x2 = x1:  y2 = y1
            endif
          endif
        endif

        CASE @IDLBUTTONUP
           ' if "drag" = 1 (that is, a piece was clicked on and has been
dragged) then update the location of that piece
           if drag = 1
              if (imageno > 0) & (imageno < 7)
                 mi = @mousey:  mi = mi - (w/2)
                 mj = @mousex:  mj = mj - (h/2)
                 bmploc[imageno,0] = mj:  bmploc[imageno,1] = mi
                 drag = 0
              endif
           endif

    CASE @IDDXUPDATE
       if state = 0 then gosub update1
       if state = 1 then gosub update2

    CASE @IDCLOSEWINDOW
        run = 0
ENDSELECT
RETURN

SUB initialize
   ' set puzzle piece width and height to 150
   w = 150
   h = 150
   imageno = -1
   drag = 0
RETURN

SUB pickbmp
   ' find out which puzzle piece was clicked on (compare the click
coordinates with the individual piece coordinates),
   '   and set imageno to the index number for that piece
   imageno = -1
   k = 6
   for i = 1 to 6
      x1 = bmploc[k,0]
      y1 = bmploc[k,1]
      x2 = bmploc[k,0] + w
      y2 = bmploc[k,1] + h
      if (mj >= x1) & (mj <= x2) & (mi >= y1) & (mi <= y2)
         imageno = k
         i = 6
      endif
      k = k - 1
   next i
RETURN

SUB update1
   ' execute one time only, to draw the sprites (puzzle pieces) initially,
then set "state" to 1
   DXFILL win,RGB(0,0,255)
   DXFLIP win
   state = 1
   for k = 1 to 6
      DXDRAWSPRITE win,spr[k]
   next k
   DXFLIP win
RETURN

SUB update2
   ' if a puzzle piece is selected (imageno <> -1) redraw all background and
all the pieces
   if imageno <> -1
      DXFILL win,RGB(0,0,255)
      for k = 1 to 6
         DXDRAWSPRITE win,spr[k]
      next k
   endif
   DXFLIP win,0,0
RETURN
```

Incidentally, if anyone is wondering why I didn't just use Tony's "drag bitmap" routine (which allows you to drag and drop a bitmap using Win API calls), I would have but (as Tony pointed out) if you have overlapping images you wind up with "artifacts" or chunks of images left on the screen.

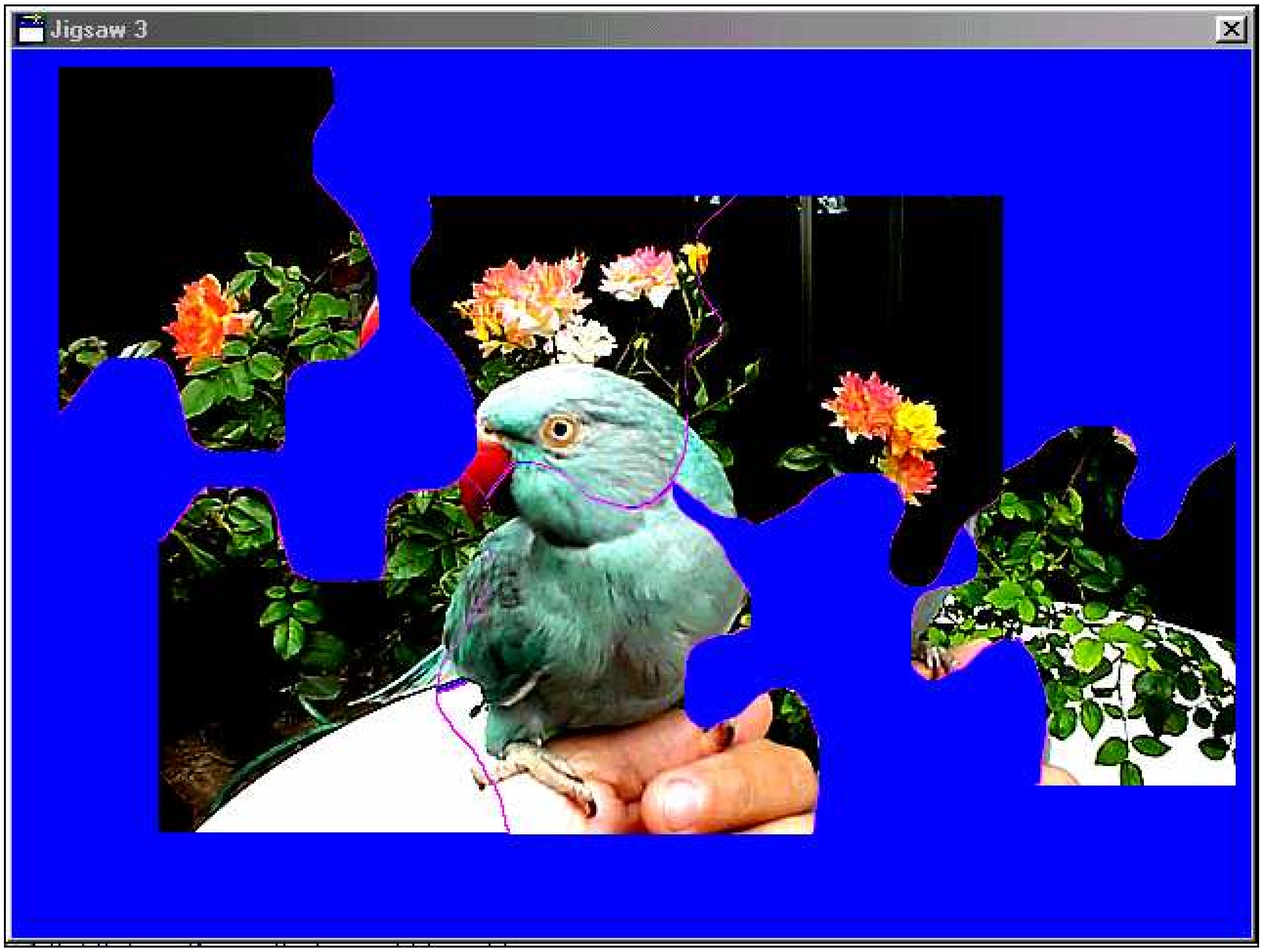
At any rate, the next challenge in creating a real jigsaw program would be to figure out how to handle irregularly shaped pieces -- if I manage that, I'll post an update on the forum. In the meantime, if anyone would like to see a really nice example of a jigsaw program, you might take a look at http://www.adcsoft.com/bjigsaw.html and try the shareware trial version.

Late Jigsaw Update - Irregularity Achieved!

With a little help (actually a lot of help) from my younger daughter, I managed to put together a modified version of the Jigsaw program that handles irregular sized pieces:

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2



The puzzle pieces were created in Adobe Photoshop by dividing the original image and putting each piece on a rectangular background with an RBG color of 241,3,244. Then I modified the Jigsaw program by loading the width and height of each puzzle piece and specifying a transparency color for each sprite of 241,3,24 so the rectangular backgrounds don't show.

```
REM Jigsaw 4
REM Author: Paul Love
REM REQUIRES DIRECTX 7.0 OR GREATER

IF GETDXVERSION < 7
      MESSAGEBOX 0,"This program requires" + chr$(13) + "DirectX 7.0 or
greater","Error"
      END
ENDIF

DEF win:WINDOW
DEF state,imageno,drag,w,h,x,y,x1,y1,x2,y2,mi,mj,i,j,k:INT
DEF path$:STRING
DEF spr[7]:INT
DEF pic[7]:STRING
def bitmapz[7]:INT
def bmploc[7,2]:INT
def bmpwh[7,2]:INT

WINDOW win,0,0,640,480,@NOAUTODRAW,0,"Jigsaw 4",mainwindow
IF CREATESCREEN(win,640,480) < 0
      MESSAGEBOX win, "Could not create DirectX screen","Error"
      CLOSEWINDOW win
      END
ENDIF

' load the sprites
path$ = GETSTARTPATH
pic[1] = path$+"parrot1.bmp"
pic[2] = path$+"parrot2.bmp"
pic[3] = path$+"parrot3.bmp"
pic[4] = path$+"parrot4.bmp"
pic[5] = path$+"parrot5.bmp"
pic[6] = path$+"parrot6.bmp"

bitmapz[1] = LoadImage (pic[1],@IMGBITMAP)
bitmapz[2] = LoadImage (pic[2],@IMGBITMAP)
bitmapz[3] = LoadImage (pic[3],@IMGBITMAP)
bitmapz[4] = LoadImage (pic[4],@IMGBITMAP)
bitmapz[5] = LoadImage (pic[5],@IMGBITMAP)
bitmapz[6] = LoadImage (pic[6],@IMGBITMAP)

GetBitmapSize (bitmapz[1],w,h)
bmpwh[1,0] = w:  bmpwh[1,1] = h
GetBitmapSize (bitmapz[2],w,h)
bmpwh[2,0] = w:  bmpwh[2,1] = h
GetBitmapSize (bitmapz[3],w,h)
bmpwh[3,0] = w:  bmpwh[3,1] = h
GetBitmapSize (bitmapz[4],w,h)
bmpwh[4,0] = w:  bmpwh[4,1] = h
GetBitmapSize (bitmapz[5],w,h)
bmpwh[5,0] = w:  bmpwh[5,1] = h
GetBitmapSize (bitmapz[6],w,h)
bmpwh[6,0] = w:  bmpwh[6,1] = h

'load the sprite(s)
spr[1] = DXSPRITE(win,pic[1],bmpwh[1,0],bmpwh[1,1],1,RGB(242,3,244))
spr[2] = DXSPRITE(win,pic[2],bmpwh[2,0],bmpwh[2,1],1,RGB(242,3,244))
spr[3] = DXSPRITE(win,pic[3],bmpwh[3,0],bmpwh[3,1],1,RGB(242,3,244))
spr[4] = DXSPRITE(win,pic[4],bmpwh[4,0],bmpwh[4,1],1,RGB(242,3,244))
spr[5] = DXSPRITE(win,pic[5],bmpwh[5,0],bmpwh[5,1],1,RGB(242,3,244))
spr[6] = DXSPRITE(win,pic[6],bmpwh[6,0],bmpwh[6,1],1,RGB(242,3,244))

gosub initialize

' build jigsaw window
x = 0:   y = 0
for k = 1 to 6
   DXSETSPRITEDATA spr[k],@SDXPOS,x
   DXSETSPRITEDATA spr[k],@SDYPOS,y
   bmploc[k,0] = x:  bmploc[k,1] = y
   x = x + 25:  y = y + 25
next k
'DXFLIP win

state = 0

run = 1
WAITUNTIL run = 0
CLOSEWINDOW win
END

mainwindow:
SELECT @CLASS
    CASE @IDLBUTTONDN
       drag = 1
       mi = @mousey
       mj = @mousex
       'i = ceil(mi / h)-1
       'j = ceil(mj / w)-1
       gosub pickbmp
       x2 = mj:  y2 = mi

    CASE @IDMOUSEMOVE
       if drag = 1
          if (imageno > 0) & (imageno < 7)
             mi = @mousey:  mj = @mousex
             mi = mi - (w/2)
             mj = mj - (h/2)
             x1 = mj:  y1 = mi
             if (abs(x1-x2) >= 0) | (abs(y1-y2) >= 0)
                bmploc[imageno,0] = mj:  bmploc[imageno,1] = mi
                DXSETSPRITEDATA(spr[imageno],@SDXPOS,mj)
                DXSETSPRITEDATA(spr[imageno],@SDYPOS,mi)
                DXMOVESPRITE spr[imageno],mj,mi
                x2 = x1:  y2 = y1
            endif
          endif
        endif

        CASE @IDLBUTTONUP
           if drag = 1
              if (imageno > 0) & (imageno < 7)
                 mi = @mousey
                 mi = mi - (w/2)
                 mj = @mousex
                 mj = mj - (h/2)
                 bmploc[imageno,0] = mj:  bmploc[imageno,1] = mi
                 drag = 0
              endif
           endif

    CASE @IDDXUPDATE
       if state = 0 then gosub update1
       if state = 1 then gosub update2
    CASE @IDCLOSEWINDOW
        run = 0
ENDSELECT
RETURN

SUB initialize
   'w = 150
   'h = 150
   imageno = -1
   drag = 0
RETURN

SUB pickbmp
   imageno = -1
   k = 6
   for i = 1 to 6
      x1 = bmploc[k,0]
      y1 = bmploc[k,1]
      w = bmpwh[k,0]:  h = bmpwh[k,1]
      x2 = bmploc[k,0] + w
      y2 = bmploc[k,1] + h
      if (mj >= x1) & (mj <= x2) & (mi >= y1) & (mi <= y2)
         imageno = k
         i = 6
      endif
      k = k - 1
   next i
RETURN

SUB update1
   DXFILL win,RGB(0,0,255)
   DXFLIP win
   state = 1
   for k = 1 to 6
      DXDRAWSPRITE win,spr[k]
   next k
   DXFLIP win
RETURN

SUB update2
   if imageno <> -1
      DXFILL win,RGB(0,0,255)
      for k = 1 to 6
         DXDRAWSPRITE win,spr[k]
      next k
   endif
   DXFLIP win,0,0
RETURN
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Freeware Reviews

Review: Inno Setup (Jordan Russell, http://www.innosetup.com)
(Used and reviewed by Jerry Muelver)

Inno Setup packages your software project into a standalone installation program.

It's incredible that this remarkable program is freeware. Inno Setup does everything the Big Expensive Installation Software does, quickly, easily, and with almost no fuss. The program steps you through the process of setting up your application with all its support files into a simple command script. Then Inno runs the script and builds a single-file EXE that provides professional-quality installation for your dream project.

Inno's documentation is detailed, helpful, and well written. With Inno around, there's no longer any excuse for packing your project as loose files in a Zip archive with nothing but a README to guide your users through installation.

Review: CS-RCS Basic (ComponentSoftware Inc.)
http://www.componentsoftware.com/Products/RCS/
Reviewed by Tony Jones

CS-RCS Basic is a Revision Control System by ComponentSoftware, Inc. They offer two versions, a Basic version, which is free and a Pro version. But don't let the Basic title fool you. This is one nice piece of software. CS-RCS allows you to backup your source code and to return to any revision of it, with just the click of a mouse button. Backing up your IBasic source is as simple as right clicking the file in Windows Explorer and choosing Add to RCS from the pop-up menu. Once a file is added to CS-RCS, you can then view the Revision History, compare the differences between two revisions, check-in a new version or check out an older version.

This is really an amazing piece of software that will save you many headaches and late nights. If you value your source code and want to protect it you can't beat CS-RCS.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Re-Sizing Windows To Fit The Screen

By

Graham Sutton

When you write a program, you put a lot of effort into making the screen layout attractive to the user. Controls, images and text are positioned carefully, fonts, and colors chosen for best effect.

You distribute your program, and users load it on to their machine and run it …. Oh dear! What has happened to your neat layout? The window probably appears much smaller in the top left corner of the screen. If the user tries to enlarge the window by dragging the border, the window size increases, but all the controls remain in place and their size remains the same.

The problem occurs whenever the user's screen resolution differs from yours. If their screen is running at 1600x1200, and you designed your program for 800x600 resolution, the window will only fill a quarter of the screen. If the user is running at 640x480 pixels, the window will be far too large to fit on the screen.

So the first problem is how to build an automatic adjustment into your code, so that windows will appear the same whatever the screen resolution.

The second, related problem is how to respond if the user drags the window borders to alter the size.

A solution is suggested that requires a slightly different way of thinking about the location of controls on the screen. Instead of absolute locations, such as so many pixels from the left, so many from the top etc. - use **Proportional** locations and sizes.

For example, if you are designing a button to be 80 pixels wide and 60 pixels high at 800x600, specify this as 0.1 wide and 0.06 high**, as proportions of window size**. Then, whatever the number of pixels on screen, the control will appear proportionally the same size.

If the window is reduced in size, you can then arrange for a SETSIZE instruction to re-size all controls to remain in correct proportional size and location. A SETFONT instruction can similarly adjust the font sizes.

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

The example program shows how this is done, using a size control arrays[ ] which holds the locations and sizes of the controls in proportional form. The array has records of user type 'resize'.

If the window is re-sized by dragging, a message is sent to the handler subroutine for the window, which is intercepted by a CASE @idsize statement.

It is not possible to issue a SETSIZE intruction immediately, since this will generate another @idsize message, which will issue another SETSIZE, which will issue another @idsize message .. and so on – the program will lock up.

So it is necessary to do the SETSIZE somewhere else. The easiest way seems to be to start a timer, and deal with the SETSIZE inside a CASE @idtimer section. This also has the advantage that the re-size operation can be timed to take place several milliseconds after the 'drag' operation is hopefully finished. The timer is then stopped, and there is no efficiency hit on the program until another re-size occurs.

If your program uses several windows, it will be necessary to use a separate size control array for the controls in each window, the re-sizing being done in the handler routine for the appropriate window.

Some elements of a program, such as images and fixed screen text, are not controls, so they cannot be dealt with within the control re-sizing loop. Each of these will need to be re-sized individually.

One other small point – if Fonts are adjusted proportionally with the controls, text will rapidly become unreadable. A suggestion is to adjust the fonts by the **Square Root** of the size-adjusting factor. This means that text will remain readable down to much smaller sizes of the window, and with care, will not overflow the control borders.

The example program window should remain the same size on the screen at all resolutions, and the screen layout should stay the same whatever size (within reason) the window is set to. Text should remain readable down to about 25% of original size.

```
def w1:window
def sW,sH,dW,dH,run,i,im:int
def sm,w1W,w1H,sf,sfH,sfW:float
def wL,wT,wW,wH:int
def iL,iT,iW,iH:float
def tL,tT:float
def tW,tH:int
def intro:string
def ncontrols:int

autodefine "off"

' layout definitions used to re-size components ...
type resize
      def l:float
      def t:float
      def w:float
      def h:float
      def font:string
      def fontsize:int
      def fontbold:int
endtype

' specify how many controls need to be re-sized ...
ncontrols=3

' array holding resize information for each component ...
def s[ncontrols+1]:resize

dW = 800:dH = 600:' design screen area ...
getscreensize sW,sH:' determine current screen size ...

sm = sW/dW: ' calculate screensize multiplier ...

w1W=0.8:w1H=0.8:' set window proportion of screen width ...

' adjust window sizes to match the screen setting ...
w1W = sm * dW * w1W : w1H = sm * dH * w1H

WINDOW w1,0,0,w1W,w1H,@MINBOX|@MAXBOX|@SIZE,0,"ScreenSize Test Main
Window",mainwindow
setwindowcolor w1,rgb(0,0,80):' set background colour to dark blue ...

GETSIZE w1,wL,wT,wW,wH:' establish window position and size ...
sf = wH/dH :' calculate the scaling factor ....

' create a button ...
' the button will be centered so the left position is not needed ..
' define button size (as proportion of screen size)...
s[1].t=0.783:s[1].w=0.14:s[1].h=0.08
s[1].l=(1-s[1].w)/2: ' this button will be centered ...
s[1].font="Arial":s[1].fontsize=12:s[1].fontbold=700
control w1,"B,Exit,(1-s[1].w)/2*wW, s[1].t*wH, s[1].w*wW, s[1].h*wH,0,1"
SETFONT w1, s[1].font, sf*s[1].fontsize, s[1].fontbold,0,1

' load your own jpeg image, size around 400x300 pixels ...
im = LoadImage(getstartpath+"seahorse.jpg",@IMGSCALABLE)
```

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
iW=0.5:iH=0.5 :' set image design size as proportion of screen size ...
iT=0.083:' set top of image  - left is not needed because it will be centered
..
SHOWIMAGE w1,im,@IMGSCALABLE, (1 - iw)/2*wW, iT*wH, iW*wW, iH*wH

' set up some fixed text ...
intro="Screen Resize":' set intro text ..
SETFONT w1, "Arial", sf*40,700, @SFITALIC
GETTEXTSIZE w1, intro, tW, tH
tT=0.15
move w1, (wW-tW)/2, tT*wH
frontpen w1, RGB(100,100,255):' set text color to light blue ...
drawmode w1,@TRANSPARENT
PRINT w1,intro

' create an edit box ...
s[2].l=0.1:s[2].t=0.65:s[2].w=0.2:s[2].h=0.05
s[2].font="Arial":s[2].fontsize=12:s[2].fontbold=500
CONTROL w1,"E,,s[2].l*wW,s[2].t*wH,s[2].w*wW,s[2].h*wH,@cteditcenter,2"
SETFONT w1, s[2].font, sqrt(sf)*s[2].fontsize, s[2].fontbold,0,2
setcontroltext w1,2,"Edit Box"

' create a text box ...
s[3].l=0.65:s[3].t=0.65:s[3].w=0.25:s[3].h=0.05
s[3].font="Times New Roman":s[3].fontsize=14:s[3].fontbold=500
CONTROL w1,"T,,s[3].l*wW,s[3].t*wH,s[3].w*wW,s[3].h*wH,@cteditcenter,3"
SETFONT w1, s[3].font, sqrt(sf)*s[3].fontsize, s[3].fontbold,0,3
setcontroltext w1,3,"Text Box"


run = 1
waituntil run = 0

DeleteImage im,@IMGSCALABLE

closewindow w1
end


' the window message processing routine ...
mainwindow:
select @CLASS
      case @IDCLOSEWINDOW
            run = 0
      case @IDCONTROL
            select @controlid
                  case 1
                        run = 0
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
            endselect
      case @IDCHAR
      ' pressing 'ESC' will abort the program ...
          Key = @CODE
            if Key=27 then run=0
      case @idsize
            starttimer w1,100
            setwindowcolor w1,rgb(0,0,80):' re-set background ..
            GETCLIENTSIZE  w1,wL,wT,wW,wH:' establish new window position and
size ...
            sfH = wH/dH: ' calculate the new scaling factor ....
            sfW = wW/dW
            if sfH<sfW
                  sf=sfH
            else
                  sf=sfW
            endif
      case @idtimer
' re-size all controls ...
            for i = 1 to ncontrols
                  SETSIZE  w1,s[i].l*wW, s[i].t*wH, s[i].w*wW, s[i].h*wH,i:'
re-size the controls ...
                  SETFONT w1, s[i].font, sf*s[i].fontsize, s[i].fontbold,0,i:
' adjust the control font ...
            next i
' re-size the image ...
            SHOWIMAGE w1,im,@IMGSCALABLE, (1 - iw)/2*wW, iT*wH, iW*wW,
iH*wH:' redo the image ...
' re-size any fixed text ...
            SETFONT w1, "Arial", sf*40,700, @SFITALIC: ' resize screen font
...
            GETTEXTSIZE w1, intro, tW, tH
            move w1, (wW-tW)/2, tT*wH
            PRINT w1,intro
            stoptimer w1
endselect
return
```

A Christmas Puzzle For You

There is a hidden puzzle in that program, which folks might not stumble across. It could (being Christmas) form the basis of a fun puzzle - I don't know the answer - but I would buy a nice bottle of wine for whoever could solve it.

Below is the same program with a number of puzzle test steps.

Step 1 is to disable the control re-sizing command in the @idtimer FOR loop.
If you run the program then, of course the controls are not re-sized when you drag

the window borders.

Step 2 is to insert the SETSIZE w1,10,10,400,300 command in the @idsize section. Resize2.iba is set up like this. If you run this one, the window is re-sized - AND the controls appear to be AUTOMATICALLY re-sized.
But this is the puzzle - how on earth does it work - it shouldn't - the SETSIZE should issue another @idsize message, which should cause the program to go into a tight loop.

(The importance is, if a SETSIZE on the whole window automatically re-sizes the controls, it would save a whole lot of programming. Only the fonts would need to be dealt with).

Step 3 is to try the SETSIZE w1,wL,wT,wW,wH command, using variables, in the @idtimer section - where it should work - but it doesn't.

Step 5 is to try the constant re-size SETSIZE w1,10,10,400,300 - which worked in the @idsize section - down in the @idtimer section. Now it doesn't work!

I've been unable to solve this puzzle. Obviously under certain circumstances, a whole window re-size will resize all controls as well - but how to get it to work reliably?

I'll be happy to send a cheque for whatever a decent bottle of wine costs in the US, as a prize for whoever solves the puzzle.

All the best,

Graham

```
' Resize2.iba

def w1:window
def sW,sH,dW,dH,run,i,im:int
def sm,w1W,w1H,sf,sfH,sfW:float
def wL,wT,wW,wH:int
def iL,iT,iW,iH:float
def tL,tT:float
def tW,tH:int
def intro:string
def ncontrols:int

autodefine "off"

' layout definitions used to re-size components ...
type resize
```

```
      def l:float
      def t:float
      def w:float
      def h:float
      def font:string
      def fontsize:int
      def fontbold:int
endtype

' specify how many controls need to be re-sized ...
ncontrols=3

' array holding resize information for each component ...
def s[ncontrols+1]:resize

dW = 800:dH = 600:' design screen area ...
getscreensize sW,sH:' determine current screen size ...

sm = sW/dW: ' calculate screensize multiplier ...

w1W=0.8:w1H=0.8:' set window proportion of screen width ...

' adjust window sizes to match the screen setting ...
w1W = sm * dW * w1W : w1H = sm * dH * w1H

WINDOW w1,0,0,w1W,w1H,@MINBOX|@MAXBOX|@SIZE,0,"ScreenSize Test Main
Window",mainwindow
setwindowcolor w1,rgb(0,0,80):' set background colour to dark blue ...

GETSIZE w1,wL,wT,wW,wH:' establish window position and size ...
sf = wH/dH :' calculate the scaling factor ....

' create a button ...
' the button will be centered so the left position is not needed ..
' define button size (as proportion of screen size)...
s[1].t=0.783:s[1].w=0.14:s[1].h=0.08
s[1].l=(1-s[1].w)/2: ' this button will be centered ...
s[1].font="Arial":s[1].fontsize=12:s[1].fontbold=700
control w1,"B,Exit,(1-s[1].w)/2*wW, s[1].t*wH, s[1].w*wW, s[1].h*wH,0,1"
SETFONT w1, s[1].font, sf*s[1].fontsize, s[1].fontbold,0,1

' load your own jpeg image, size around 400x300 pixels ...
im = LoadImage(getstartpath+"seahorse.jpg",@IMGSCALABLE)

iW=0.5:iH=0.5 :' set image design size as proportion of screen size ...
iT=0.083:' set top of image  - left is not needed because it will be centered
..
SHOWIMAGE w1,im,@IMGSCALABLE, (1 - iw)/2*wW, iT*wH, iW*wW, iH*wH
```

```
' set up some fixed text ...
intro="Screen Resize":' set intro text ..
SETFONT w1, "Arial", sf*40,700, @SFITALIC
GETTEXTSIZE w1, intro, tW, tH
tT=0.15
move w1, (wW-tW)/2, tT*wH
frontpen w1, RGB(100,100,255):' set text color to light blue ...
drawmode w1,@TRANSPARENT
PRINT w1,intro

' create an edit box ...
s[2].l=0.1:s[2].t=0.65:s[2].w=0.2:s[2].h=0.05
s[2].font="Arial":s[2].fontsize=12:s[2].fontbold=500
CONTROL w1,"E,,s[2].l*wW,s[2].t*wH,s[2].w*wW,s[2].h*wH,@cteditcenter,2"
SETFONT w1, s[2].font, sqrt(sf)*s[2].fontsize, s[2].fontbold,0,2
setcontroltext w1,2,"Edit Box"

' create a text box ...
s[3].l=0.65:s[3].t=0.65:s[3].w=0.25:s[3].h=0.05
s[3].font="Times New Roman":s[3].fontsize=14:s[3].fontbold=500
CONTROL w1,"T,,s[3].l*wW,s[3].t*wH,s[3].w*wW,s[3].h*wH,@cteditcenter,3"
SETFONT w1, s[3].font, sqrt(sf)*s[3].fontsize, s[3].fontbold,0,3
setcontroltext w1,3,"Text Box"


run = 1
waituntil run = 0

DeleteImage im,@IMGSCALABLE

closewindow w1
end


' the window message processing routine ...
mainwindow:
select @CLASS
      case @IDCLOSEWINDOW
            run = 0
      case @IDCONTROL
            select @controlid
                  case 1
                        run = 0
            endselect
      case @IDCHAR
      ' pressing 'ESC' will abort the program ...
          Key = @CODE
            if Key=27 then run=0
      case @idsize
```

```
            starttimer w1,100
            setwindowcolor w1,rgb(0,0,80):' re-set background ..
            GETCLIENTSIZE  w1,wL,wT,wW,wH:' establish new window position and
size ...

'********************************************************************
' Puzzle Step 2:  How does this manage to automatically re-size controls?
' when it shouldn't really work at all in this position.
' Notice fixed values are used for re-sizing the whole window ...
'********************************************************************
            SETSIZE w1,10,10,400,300
'********************************************************************

            sfH = wH/dH: ' calculate the new scaling factor ....
            sfW = wW/dW
            if sfH<sfW
                  sf=sfH
            else
                  sf=sfW
            endif
      case @idtimer
' re-size all controls ...
            for i = 1 to ncontrols

'********************************************************************
' Puzzle Step 1:  Comment out the following Control Re-Size command .....
'********************************************************************
'                 SETSIZE  w1,s[i].l*wW, s[i].t*wH, s[i].w*wW, s[i].h*wH,i:'
re-size the controls ...
'*********************************************************

                  SETFONT w1, s[i].font, sf*s[i].fontsize, s[i].fontbold,0,i:
' adjust the control font ...

            next i

'********************************************************************
' Puzzle Step 3:  And so why doesn't this work with variable values?  .....
'********************************************************************
'           SETSIZE w1,wL,wT,wW,wH

' Puzzle Step 4:  and neither does this in this position .....
'********************************************************************
'           SETSIZE w1,10,10,400,300
'********************************************************************

' re-size the image ...
            SHOWIMAGE w1,im,@IMGSCALABLE, (1 - iw)/2*wW, iT*wH, iW*wW,
iH*wH:' redo the image ...
```

```
' re-size any fixed text ...
            SETFONT w1, "Arial", sf*40,700, @SFITALIC: ' resize screen font
...
            GETTEXTSIZE w1, intro, tW, tH
            move w1, (wW-tW)/2, tT*wH
            PRINT w1,intro
            stoptimer w1
endselect
return
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Christmas Greetings

'Tis the season to be joyous and celebrate with our families and friends and share laughter and good times. And, as Paul said in his column, the IBasic community has grown into one big extended family. A few members of this "extended family" decided to send their holiday greetings in the form of IBasic programs as listed below. To send your own greeting, please visit the following thread on the IBasic forums; www.pyxia.com/community/viewtopic.php?t=4990

Happy holidays all, enjoy! - **Editor**

```
' Merry Christmas all
' From RICK_LETT and family
' Have a joyous Christmas

DEF w1:WINDOW
DECLARE "kernel32",Sleep(dwMilliseconds:INT),INT
WINDOW w1,0,0,350,350,@MINBOX|@SIZE,0,"Merry Christmas",main
setfont w1,"papyrus",30,200,@sfitalic

frontpen w1,rgb(0,255,0)
setwindowcolor w1,rgb(0,0,0)
print w1 ,"       PEACE"
rect w1,130,150,50,170,rgb(0,0,0),rgb(255,255,255)
starttimer w1,1


WAITUNTIL w1 = 0
END

SUB main
    select @CLASS
         case @IDCLOSEWINDOW

           stoptimer w1
           CLOSEWINDOW w1
         case @idtimer
         flik = rnd(40)-25

      ellipse w1,147,130,12,flik,rgb(255,180,0),rgb(255,180,0)
       sleep(18)
      ellipse w1,147,125,12,flik,rgb(255,170,0),rgb(255,130,0)
endselect
RETURN
```

```
' Christmas tree
' By LarryA
DEF w:WINDOW
DEF rise, rad, frad, xshorten:float
DEF left, top, width, height, bpx, bpy, tpx, tpy:int
DEF x1, y1, x2, y2:int
WINDOW w, 20, 5, 440, 460, @caption, 0, "Merry Christmas", main
setwindowcolor w, rgb(255,235,190)
brown=rgb(130, 100, 0)
green=rgb(0, 80+rnd(40), 0)
'== GROW ===================
bpx=220 : bpy=410 : tpx=bpx
for aa=-4 to 4
  line w, bpx+aa, bpy, bpx, bpy-390, brown
next aa
rad=160: tpy=bpy-40
for ht=1 to 40
  for xs=-100 to 100 #40
    xshorten=xs/100
    rise=rnd(.3)
    line w, tpx, tpy, tpx+(xshorten*rad), tpy-rise*rad, rgb(0,80+rnd(40),0)
    for aa=1 to 30
       frad=rnd(.9)*rad
       x1=tpx+(xshorten*frad)
       y1=tpy-rise*frad
       x2=tpx+xshorten*(frad+rad/5)
       y2=tpy-rise*frad+(-rise+(rnd(.8)-.4))*(rad/5)
       line w, x1, y1, x2, y2, rgb(0, 80+rnd(40), 0)
       wait 1
    next aa
  next xs
  rad=rad-4 : tpy=tpy-9
next ht
'== DECORATE ======================
rad=160 : tpy=bpy-50
for ht=1 to 37
  circle w, tpx+rnd(2*rad)-rad, tpy+rnd(6)-3, 5, rgb(255,0,0),rgb(255,0,0)
  rad=rad-4 : tpy=tpy-9
next ht
'== DRAW BORDER ===========================
getclientsize w,left,top,width,height
setlinestyle w, @lssolid, 3
rect w, left+2, top+2, width-4, height-4, rgb(255,0,0)
rect w, left+5, top+5, width-10, height-10, rgb(0,120,0)
rect w, left+8, top+8, width-16, height-16, rgb(0,120,0)
rect w, left+11, top+11, width-22, height-22, rgb(255,0,0)
'== GREETING ==============================
setfont w, "times", 14, 700, @sfitalic
frontpen w, rgb(0,120,0)
move w, left+30, top+20 : print w, "Merry"
move w, left+20, top+50 : print w, "Christmas"

run = 1: WAITUNTIL run = 0: CLOSEWINDOW w: END

SUB main
SELECT @CLASS
  CASE @IDCLOSEWINDOW
    run = 0
ENDSELECT
RETURN
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
' Snow By Jolly_Roger
openconsole
locate 23,1
print "Press Q to exit"
color 15,9
DECLARE "kernel32",GetTickCount(),int
def sfx[31],sfy[31],snflky,snflkx,r:int
def ch1,ch2,ch3,ch4,ch5,bgimage[22],spcs22,blankline,bckbuffer[22]:string
'set up snowflake initial positions
for snowflake=1 to 30
  sfx[snowflake]=1+rnd(79)
  sfy[snowflake]=-rnd(20)
next snowflake
'set up initial background image
ch1=chr$(255):ch2=ch1+ch1:ch3=ch2+ch1:ch4=ch2+ch2:ch5=ch2+ch3
spcs22="                      "
blankline=spcs22+spcs22+spcs22+"              "
bgimage[8]=spcs22+ch4+"   "+ch5+"   "+ch3+"    "+ch4+"  "+ch5+"   "+spcs22
bgimage[9]=spcs22+ch1+"   "+ch1+"  "+ch1+"      "+ch1+"   "
bgimage[9]=bgimage[9]+ch1+"  "+ch1+"      "+ch1+"       "+spcs22
bgimage[10]=spcs22+ch1+"   "+ch1+"  "+ch1+"      "+ch1+"   "
bgimage[10]=bgimage[10]+ch1+"  "+ch1+"      "+ch1+"       "+spcs22
bgimage[11]=spcs22+ch4+"   "+ch5+"  "+ch5+"  "+ch1+"      "+ch5+"   "+spcs22
bgimage[12]=spcs22+ch1+"      "+ch1+"      "+ch1+"   "+ch1+"  "+ch1+"
"+ch1+"       "+spcs22
bgimage[13]=bgimage[12]
bgimage[14]=spcs22+ch1+"      "+ch5+"  "+ch1+"   "+ch1+"   "+ch4+"  "+ch5+"
"+spcs22
for row=1 to 7
  bgimage[row]=blankline
  bgimage[row+14]=blankline
next row
lastframetime=GetTickCount()
'main loop
do
  'move snowflakes down one line
  for snowflake=1 to 30
    sfy[snowflake]=sfy[snowflake]+1
    'if off bottom of display area,reset to top
    if sfy[snowflake]>21 then gosub resetsnowflake
  next snowflake
  for snowflake=1 to 30
    'check if in position can stick
    if (sfy[snowflake]>7) & (sfy[snowflake]<15) & (sfx[snowflake]>22) &
(sfx[snowflake]<56)
      snflky=sfy[snowflake]:snflkx=sfx[snowflake]
      if mid$(bgimage[snflky],snflkx,1)=ch1
        'make snowflake stick by copying snowflake character onto background
image
        bgimage[snflky]=left$(bgimage[snflky],snflkx-
1)+"*"+mid$(bgimage[snflky],snflkx+1)
        'reset snowflake to top of screen
        gosub resetsnowflake
      endif
    endif
  next snowflake
  'copy background image to backbuffer
  for row=1 to 21
    bckbuffer[row]=bgimage[row]
  next row
  'draw snowflakes to backbuffer
  for snowflake=1 to 30
    if sfy[snowflake]>0
      snflky=sfy[snowflake]:snflkx=sfx[snowflake]
      bckbuffer[snflky]=left$(bckbuffer[snflky],snflkx-
1)+"*"+mid$(bckbuffer[snflky],snflkx+1)
    endif
  next snowflake
  'wait loop so runs at 10 frames per second
  do
    timesincelastframe=GetTickCount()-lastframetime
  until timesincelastframe>=100
  lastframetime=GetTickCount()
  'show backbuffer
  locate 1,1
  for row=1 to 21
    print bckbuffer[row],
  next row
until ucase$(inkey$)="Q"
closeconsole
end

sub resetsnowflake
  sfy[snowflake]=1
'weight snowflake x position so stick more evenly
  r=rnd(40)
  select r
    case 1
      sfx[snowflake]=23
    case 2
      sfx[snowflake]=30
    case 3
      sfx[snowflake]=37
    case 4
      sfx[snowflake]=41
    case 5
      sfx[snowflake]=44
    case 6
      sfx[snowflake]=51
    default
      sfx[snowflake]=1+rnd(79)
  endselect
return
```

```
'xmas.iba,13-DEC-2002,By Boris
DECLARE "kernel32",SleepEx(dwMilliseconds:INT, bAlertable:INT),INT
def snw$[25],scene$[25],grnd$[26],grt$[6],s$:string
def gz,w,b:int:def wrd[4]:uint
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

```
for n=1 to 24:snw$[n]=space$(80):grnd$[n]=space$(80)
for nn=1 to 3:replace$ snw$[n],int(rnd(79)+1),1,".":next nn:next n
wrd=2783918575,4225693575,453492014,64876348:grt$[5]=gosub mkg
wrd=2904638497,541233280,612402532,4794692:grt$[4]=gosub mkg
wrd=3039788271,567210119,545752484,63564092:grt$[3]=gosub mkg
wrd=623395880,541218692,168049956,34122052:grt$[2]=gosub mkg
wrd=2783921647,601789447,67608878,199044924:grt$[1]=gosub mkg
grnd$[0]="1":openconsole:locate 25,1:color 15,0:print string$(80,"#"),
do:setsc:for n=1 to 24:locate n,1:print scene$[n],:next n
sleepex(45,0):until (inkey$<>"")|(val(grnd$[0])=25):closeconsole:end
setsc:
gz=val(grnd$[0]):snw$[0]=snw$[24]
for n=24 to 1 step -1:scene$[n]=snw$[n]
   if rnd(3)>1 then snw$[n]=mid$(snw$[n-1],2)+left$(snw$[n-1],1) else
snw$[n]=snw$[n-1]
   if n<6 then grt$[n]=mid$(grt$[n],2)+left$(grt$[n],1)
next n
if gz<13
   replace$ grnd$[gz],rnd(78)+1,2,"##":if rnd(3)>2 then replace$
grnd$[gz+1],rnd(79)+1,1,"#"
   if instr(grnd$[gz],"############################")
      grnd$[0]=str$(gz+1):grnd$[gz]=string$(80,"#")
   endif
else
   locate 25,36:color 15,rnd(3)-1:print " <ANY KEY> ",
endif
for n=1 to gz+1
   scene$[25-n]=left$(grnd$[n],80):if (n>5)&(n<11) then scene$[27-
n]=left$(grt$[n-5],80)
next n:color 15,0:return
mkg:
s$="":for w=1 to 4:for b=0 to 31:if ((2^b)&wrd[w-1]) then s$=s$+" " else
s$=s$+"#"
next b:next w:return s$+"###################################"
```

IBasic Monthly
Dedicated to everything IBasic!

December 15th, 2002 *Seasons Greetings!* Volume 1, Issue 2

Coming Next Month

John Sylvester shows us how to avoid huge IF and SELECT constructs when validating options by using INTEGER arrays. Jerry Muelver brings us up to speed on using Fletchies DynaString component and Matt Cox, show's us how to get the PC User Name, in Inside The Windows API. Rick Lett leads us on another exciting adventure as he introdues us to strings, plus articles on Drag and Drop, Windows Events and using resources with IBasic and much, much more! So look for your copy January 15th!

Happy holidays to all!

Tony Jones & Rick Lett
IBasic Monthly Staff