TradeStation | TradingApp Store | Developer Center | Institutional Services | Call Tradestation 800.822.0512 | Client Center | LogIn

You are logged in to the forums as GerryTS
NO NEW PRIVATE MESSAGES
FORUM LOGOUT

WHY TRADESTATION PRODUCTS TRADING TECHNOLOGY EDUCATION COMMUNITY

TradeStation Wiki | Most Recent Forum Posts | My Forum Subscriptions | Forum Help | Quick Forum Search Go | Advanced Forum Search

Home > All Forums > EasyLanguage Library >
TradeStation Compatible DLLs > Collections for EasyLanguage

New Poll Add Reply Subscribe to Topic

8 Pages 1 2 3 4 5 6 7 8 > »

Next Page

Author Topic Send topic to friend Printer Friendly

Bamboo
1258 Posts

Posted - 12/02/2004 17:58:41

***Latest Revision:** v1.05, posted 2006-03-01 10:10 PST*

Have you ever wished that EasyLanguage provided some of the powerful and flexible data structures offered by other programming languages? The built-in arrays are a wonderful tool, but they do not offer the flexibility of a full-fledged collections library.

I created the ELCollections library to fill that gap. This library offers two kinds of collections for EasyLanguage: lists and maps.

A **list** is like a supercharged dynamic array. You can get and put data at any index, just as you can with an array, but a list offers many additional capabilities:

- You can grow the list dynamically as you go. You don't have to worry about setting the size of the list first.
- You can push data onto the back or front of a list, and you can pop it off the back or front.
- You can insert data at any position in the list, and you can remove it from any position.
- You can sort the list in very flexible ways, including keeping it sorted as you insert items.
- You can perform fast lookups on sorted lists.
- A list can be shared between multiple symbols and studies.
- A list can hold other collections (lists and/or maps) as well as numbers or strings.
- A list can be saved to a text file and read from a text file.

A **map** is a collection of key-value pairs. Each *key* in the map is associated with a particular *value*, and you can use the key to quickly look up the value.

There's a common example of a map that we all use in our daily lives: it's called a telephone book. The keys are the names, and the associated values are the telephone numbers. Just as you can quickly look up a number by name in a telephone book, you can quickly look up a value by its key in a map. Furthermore, it's easy to add key-value pairs to a map and to remove them from the map. Maps are designed to perform all of these operations very quickly.

You can use maps in EasyLanguage to save data by symbol, by date, by bar, by price, and in many other ways.

Maps offer many of the same capabilities as lists:

- A map can be shared between multiple symbols and studies.
- A map can hold other collections (lists and/or maps) as well as numbers or strings.
- A map can be saved to a text file and read from a text file.

The ability to use maps and lists together (with collections of collections) makes it possible to create data structures of arbitrary complexity.

For a powerful example of what you can do with ELCollections, check out my All-Data-Everywhere library. ADE provides a high-level framework for storing data for any symbol and timeframe, and then accessing that data from any other symbol and timeframe.

If you'd like to learn more about ELCollections before you download the library, here is the documentation:

Attachment:DATA/20041222135612ELCollections.doc 115712 bytes

Here is the full library, which includes the documentation:

Attachment:DATA/20060301130930ELCollections.zip 270455 bytes

To install the ELCollections library, just perform the following steps:

1. Unzip the zip file into a directory.

2. Copy the ELCollections.dll file to your TradeStation program directory (e.g. C:\Program Files\TradeStation 8.0 (Build xxxx)\Program).

3. Import the ELCollections.ELD file into TradeStation.

4. Install the library in your brain. In other words, read the documentation! This is a big library, so studying the documentation will save you a lot of time and effort in the long run.

Version History
1.00: Initial beta release.
1.01: Read very long lines from text file. Improve performance with string keys (works around TS issue).
1.02: Many new functions. See post below for details.
1.03: Updated to support ADE with TypeZero bars.
1.04: Fixed issue with ADE TypeZero data files.
1.05: Fixed ListN.InsertSorted so that it always returns the correct value.

You can check what version you have by opening the ELC_Version.txt file from the ELCollections.zip file. If you don't have that file, you don't have the latest version.

ELCollections is in the public domain and may be freely used and distributed.
The ELCollections software is offered "as-is", with no warranty, either express or implied, of its suitability for any purpose.

Poll Description:

Choices:

○ Extremely Useful

○ Useful

○ Somewhat Useful

○ Not Useful

VOTE NOW View Results

Bamboo
1258 Posts

Posted - 12/02/2004 18:00:23

FAQs, White Papers, and Source Code

Frequently Asked Questions about ELCollections and ADE

(A big *thank you* to mmillar for creating and maintaining this topic!)

White Papers

How Collection Ownership Works: DATA/20041227160523Collection Ownership.doc 25600 bytes

Source Code

For those who wish to compile the source code themselves, here it is. You will need Microsoft Visual C++ 6.0 or later to compile this.

Attachment:DATA/20060301131312ELCollectionsCode.zip 31728 bytes

Please note that the availability of source code means that there may be versions of ELCollections.dll out there that are different from what I've written. If you want to be sure that you have the version I wrote, then you must either (a) download the DLL from this topic, or (b) compile it yourself from the source code provided here.

Also, I cannot provide any support for the source code. If you want to modify and extend it, you're on your own. The source code is intended for those with a high level of working knowledge about C++, or for those who need to compile the code themselves because of policy or other issues.

Bamboo
1258 Posts

Posted - 12/02/2004 18:04:32

A word about support. I will do my best to provide support for ELCollections here on TSW -- within reason! Since I am offering this library for free, I can't provide the same level of service that you would get with a commercial product. After all, I am a trader, and that must remain my primary focus.

If the community finds this library useful, I hope that other members will help me support it. I strongly encourage other members to post examples and demos, and to respond to other members' questions if you know the answer. Any help I receive in supporting this product will be greatly appreciated!

BlackSwan 13
3346 Posts

Posted - 12/02/2004 18:22:09

Thank you Bamboo !!

s-a

686 Posts

Posted - 12/02/2004 18:41:39

Awesome stuff! Bamboo, do you think it might be possible for generic iterators? This could work for collections on all items, or on maps allowing for iteration of each key and value, or perhaps even iteration of just keys or just values.

HG

Bamboo
1258 Posts

Posted - 12/02/2004 18:46:38

HG,

Lists and Maps both provide iteration functions called Rewind and Next. For Lists, the Next function iterates through the values in the list in index order. For Maps, the Next function iterates through the key-value pairs (returning both key and value) in key order.

eKam
9367 Posts

Posted - 12/02/2004 20:05:57

Another great contribution. Thank you, Bamboo.

minidot
184 Posts

Posted - 12/02/2004 22:30:22

Thank You, Bamboo.

s-a
686 Posts

Posted - 12/03/2004 01:11:21

Bamboo,

Outstanding, seems like you have put together a first class generic collections framework - thank you.

Signed,
Holden Glova

LongBow
VEGAS CONFERENCE 2005
185 Posts

Posted - 12/03/2004 16:34:47

Bamboo,

A master piece!

It is not just the construct, but the architecture and the educational material as well. I dare to say this because I coded my own Lists and Maps in C++ prior to Tools++ and STL. I have always dreamed to have Maps and Sorted Lists in EL.

ADE will change what we think is possible in EL. Many creative programs will emerge, just like Maps and Lists did to C++ and Java. ADE will soon be, if not already, a major milestone in EL history.

Thank you for this important contribution. My hat's off to you.

dougcs
592 Posts

Posted - 12/03/2004 19:33:27

WOW!!!!!!!!!!!!!!!!

Great stuff; thank you for sharing this.

DS

Steve in California
571 Posts

Posted - 12/04/2004 11:29:36

I browsed the Collections documentation last evening.

Now that we have this Collections utility--and AFTER it's thoroughly tested and any inevitable bugs fixed--is there any reason for using any other Global Variables utility?

Seems to me that the answer is no. What do you think?

Thanks Bamboo!

eKam
9367 Posts

Posted - 12/04/2004 12:14:16

The doc of ELCollections says this (emphasis mine):

> **Note that a collection can only be shared within a single application**. Charting, RadarScreen, and OptionStation are different applications, so a collection cannot be shared across them.

That would be the reason to keep GV.

Bamboo, any plans to improve ELCollections to work across applications?

Bamboo

Posted - 12/04/2004 12:22:31

1258 Posts

eKam,

You're right. If you need to pass information across application boundaries (e.g. from charting to RadarScreen or vice versa), then you need to use GlobalVariables.

For most cases where you need to pass information around inside an application (e.g. from chart to chart, indicator to indicator, or symbol to symbol within RS), I believe that ELCollections will be a superior tool.

Because of the high flexibility of ELCollections' data structures, writing a version that works cross-process is very challenging. It's probably not impossible, but it would be a big chunk of work. (For the technically minded, it would mean writing custom allocators for STL.) I'm not ruling it out, but I'm not promising it any time soon either!

Camb

VEGAS CONFERENCE 2005

38 Posts

Posted - 12/07/2004 20:34:39

Bamboo,

Let me first say this is a fantastic piece of work that many people should find very useful.

While running some initial tests, I came across the following bug/limitation that I thought was worth mentioning. Essentially, the ReadFile functionality for Lists of Lists or Maps of Lists only functions if the number of sublists is moderately small (a few hundred at most). If the number of sublists is too large the comma delimited data input line apparently becomes longer than the read function can handle and locks up TS. Here is a simple sampe strategy to reproduce the behaviour. Just increase the sublistSize input value to say 2000 and you should get a reproduction of the problem.

```
                Value1 = ListN.PushBack(ListID,Open);
                Value1 = ListN.PushBack(ListID,Open);
                Value1 =ListC.PushBack(BigListID, ListID);
        end;

        print("write file");
        Value1=ListC.WriteFile(BigListID,"c:\temp\testwrite4.txt");
        print("clear list");
        Value1=ListC.Clear(BigListID);
        print("read file back");
        Value1=ListC.ReadFile(BigListID,"c:\temp\testwrite4.txt");
        print("test done");
end;
```

select all

Bamboo

1258 Posts

Posted - 12/07/2004 20:44:01

Camb,

Thanks for pointing that out. I think the problem is that the DLL reads each line into a buffer, and the lines are exceeding the buffer size.

Unfortunately there's no easy way for the DLL to predict how big a buffer it will need, but I will add a function that allows the user to set the line buffer size. Also, I will try to make the ReadFile function more robust so that it just generates a runtime error instead of locking up TS.

In the meantime, the best advice I can give is to "rotate" the data if possible so that the smaller number of items is represented by the sublists. I realize that this may not be an option in all cases. But (for example) if you want to store metrics by symbol, the optimal arrangement is to make the metrics the sublists (columns), and make the symbols the rows. This allows you to store an arbitrarily large number of symbols. You can get fast lookup of the symbols by using BuildRowMap to create an index of the symbols.

I will soon be posting a TableN object (written completely in EL using ELCollections) that provides a higher-level abstraction of exactly this functionality.

Camb

VEGAS CONFERENCE 2005

38 Posts

Posted - 12/07/2004 21:07:25

Bamboo,

> quote:
>
> the best advice I can give is to "rotate" the data

That is exactly what I did when I discovered the issue and it worked perfectly. I really worked up the example to reproduce the problem as a small repayment for the huge contribution you are making to everyone. I have always found that thorough testing and reproduction of issues is the most time consuming and least rewarding aspect of development (although some folks apparently love that portion of the development process, :)). I just hope my input will save you a small amount of effort in the future.

Thanks,

Camb

Camb

38 Posts

Posted - 12/08/2004 21:56:54

Bamboo,

I just finished a few more tests associated with my use of your utility and wanted to provide feedback on another item. It appears as though the speed performance of the Get function for Maps that use strings as the key, degrades with each call and builds up a fair amount of memory that is not deallocated until final garbage collection at the end of a strategy evaluation. Here is a code snippet to replicate the behavior and the manually timed test results. Note the nonlinear increase in time required and the constantly increasing memory usage by orchart.exe. You may want to explicitly deallocate the memory that is being utilized inside the Get calls at the end of each call to alleviate this issue.

// Simple string map lookup speed test
// Manual time test results
// 3 seconds for 5,000 - max utilized by orchart.exe ~91,000 K
// 7 seconds for 15,000 - max utilized by orchart.exe ~116,000 K
// 10 seconds for 20,000 - max utilized by orchart.exe ~145,000 K
// 24 seconds for 30,000 - max utilized by orchart.exe ~190,000 K
// infinite loop error for 50,000 - max utilized by orchart.exe ~200,000 K

```
print("test start");
Value6=MapSC.New;
Value1=MapSC.Put(Value6,"List1",ListN.New);
Value1=MapSC.Put(Value6,"List2",ListN.New);

Value7=MapSC.New;
Value1=MapSC.Put(Value7,"List1",ListN.New);
Value1=MapSC.Put(Value7,"List2",ListN.New);
Value1=MapSC.Put(Value7,"List3",ListN.New);
Value1=MapSC.Put(Value7,"List4",ListN.New);
Value1=MapSC.Put(Value7,"List5",ListN.New);

for value2=1 to 50000 begin
        Value1=MapSC.Get(Value6,"List2");
```

select all

I ran a similar test for MapNC maps and saw no performance degradation or memory allocation creep at all. Here is the code snippet for that test and the results.

// Simple number map lookup speed test
// Manual time test results
// 2 seconds for 50,000
// 4 seconds for 250,000

```
Value6=MapNC.New;
Value1=MapNC.Put(Value6,1,ListN.New);
Value1=MapNC.Put(Value6,2,ListN.New);
Value1=MapNC.Put(Value6,3,ListN.New);

Value7=MapNC.New;
Value1=MapNC.Put(Value7,1,ListN.New);
Value1=MapNC.Put(Value7,2,ListN.New);
Value1=MapNC.Put(Value7,3,ListN.New);
Value1=MapNC.Put(Value7,4,ListN.New);
Value1=MapNC.Put(Value7,5,ListN.New);
Value1=MapNC.Put(Value7,6,ListN.New);
Value1=MapNC.Put(Value7,7,ListN.New);
Value1=MapNC.Put(Value7,8,ListN.New);
```

select all

Please note that my test times are very rough as I am simply toggling the on/off status for the strategy that I have the code snippets embedded in and measuring the time until the strategy finishes refreshing.

Hope you find the feedback helpful.

Best Regards,

Camb

Bamboo

1258 Posts

Posted - 12/08/2004 23:01:36

Camb,

Thanks for the report regarding string keys and performance. I wish the problem was in my DLL (because then I could fix it), but unfortunately all the evidence points to this being a TS issue.

To illustrate this, I changed the definition of MapSC_Get (the underlying C++ function) as follows:

```
double __stdcall MapSC_Get(IEasyLanguageObject* pEL, COLLECTION_ID nID, LPSTR key)
{
  return 0;
}
```

select all

If I run your test with this change, I get essentially the same problem. Furthermore, if I remove the indicator from the chart, the memory shown for orchart.exe in Task Manager is not released. I double-checked that the DLL was indeed unloaded (by making a change and recompiling; this will fail if the DLL is still loaded) -- so I'm as sure as I can be that this memory is not being held by the DLL. I believe the performance issues are on the TS side as well.

I don't think a DLL is supposed to deallocate string arguments passed to a function. The extension SDK documentation is silent on this point (if memory serves), but the ELX code that has been made available doesn't deallocate these pointers. But just to be very sure that this wasn't the problem, I added a deallocation to the stub function above, crossed my fingers, and ran the test again. Same results. (It didn't even crash, which is pretty weird.)

I don't know what is going on here, but it sure looks like some kind of weirdness in the EL->DLL interface. I think we should take this up with TS Support.

Camb
VEGAS CONFERENCE 2005
38 Posts

Posted - 12/08/2004 23:51:35

Bamboo,

It definitely sounds like a TS issue at this point. I'm switching to working with MapNC maps instead of the MapSC maps for now.

Thanks for the fast response, I'll let you know if I come across anything else of interest.

> quote:
>
> I don't think a DLL is supposed to deallocate string arguments passed to a function.

I believe you are correct. Given the size of the memory creep, I was assuming there were some allocations within the DLL above and beyond the memory required to simply pass the string. Your test clearly showed that was not the case. TS must be allocating a little over 3k of buffer space for each string as they pass it over and it isn't getting cleaned up until strategy finish for some reason.

trado
5 Posts

Posted - 12/09/2004 02:35:48

Bamboo,

Thank you so much for ELCollections.

TradeStation Securities should owe you big time for bringing the TS to higher level.🙂

Bamboo
1258 Posts

Posted - 12/09/2004 12:00:41

Good news! I found a workaround for the string key problem that Camb uncovered. On a hunch I changed the EL definition of MapSC.Get from this:

```
external method: "ELCollections.dll", double, "MapSC_Get", double, LPSTR;

Inputs:
        ID(NumericSimple),
        Key(StringSimple);

MapSC.Get = MapSC_Get(ID, Key);
```

select all

... to this:

```
external method: "ELCollections.dll", double, "MapSC_Get", double, LPSTR;

Inputs:
        ID(NumericSimple),
        Key(StringSimple);

Vars:
        vKey("");

vKey = Key;
MapSC.Get = MapSC_Get(ID, vKey);
```

select all

Voila! The test code now runs quickly without excessive memory consumption.

As you can see, all I'm doing is first assigning the string argument to a string variable, and then passing the *variable* to the DLL function instead of the *argument*. Apparently passing a string variable works fine, but TS is buggy when passing a string argument to a DLL function.

I will modify all of the relevant functions in the ELCollections library to use this technique. I'll try to post the update later today.

(One warning: If you make this change yourself, it takes a very, *very* long time to verify. You may even think TS has gone into an infinite loop -- I certainly did! Fortunately, I waited it out and it did complete. But don't try this at home unless you're prepared for a long wait.)

eKam

9367 Posts

Posted - 12/09/2004 13:18:11

Wow, what a workaround.

I have saved this tidbit in my gems page.

Bamboo

1258 Posts

Posted - 12/09/2004 23:20:14

I have posted a new version (1.01) of ELCollections. This addresses the following issues:

1. The ReadFile functions can now read very long lines of text -- up to one megabyte. I implemented an adaptive read buffer that starts at 8K and resizes itself dynamically if necessary to adjust to longer lines of text. This addresses the issue with reading some files which Camb reported above. For example, his test indicator in that post now works with 100,000 columns. If the maximum line length of 1MB is exceeded, ELCollections will raise a runtime error.

2. I implemented the workaround described above for the TS problem with passing string arguments to DLL functions. This should solve the performance problems with collections that contain large numbers of string keys or values.

3. There is now an ELC.RaiseError function. Please use this if you need to raise a runtime error in your code that uses ELCollections or ADE. I have found the TS RaiseRuntimeError to be problematic when working with ELCollections, and the ELC.RaiseError function avoids those problems. The first argument is the AppName (a string describing your application). The second argument is the error message.

4. The ListC.Clear, ListC.Resize, MapNC.Clear, and MapSC.Clear functions will now release the memory for subcollections immediately, rather than when the study memory is released.

jtbaccarat

221 Posts

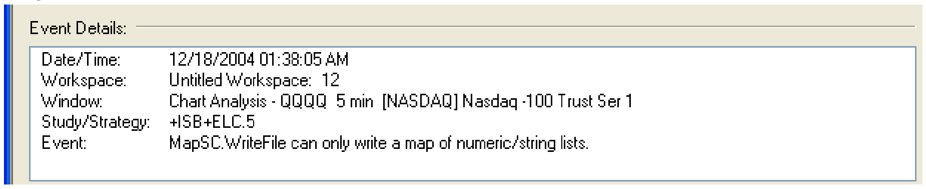
Posted - 12/18/2004 01:50:14

Bamboo,

I am going through the documentation. On p.10 there is a code example to populate a map of maps. I copied the code exactly, and used my JTMBarDateTime, which has a slight addition to the eKam version DateTimeStr, found here:
https://community.tradestation.com/discussions/Topic.aspx?TOPIC_ID=31411

I can get this to verify, but it triggers an ELCollections error.

Error:


```
Event Details:
Date/Time:        12/18/2004 01:38:05 AM
Workspace:        Untitled Workspace: 12
Window:           Chart Analysis - QQQQ  5 min  [NASDAQ] Nasdaq -100 Trust Ser 1
Study/Strategy:   +ISB+ELC.5
Event:            MapSC.WriteFile can only write a map of numeric/string lists.
```

Code for Map of Maps from ELCollections documentation:

```
Vars:
        MapID(MapSC.Share(GetSymbolName + "_BarMap")),
        PriceBar(0);

// This code runs on every bar, so it will add a new map
// of OHLC values to the map on every bar.
PriceBar = MapSN.New;
Value1 = MapSN.Put(PriceBar, "Open", Open);
Value1 = MapSN.Put(PriceBar, "High", High);
Value1 = MapSN.Put(PriceBar, "Low", Low);
Value1 = MapSN.Put(PriceBar, "Close", Close);

Value1 = MapSC.Put(MapID, JTMBarDateTime(0, 1, 1, 0, "", ""), PriceBar); // add PriceBar to map
```

select all

Code for JTMBarDateTime:

```
//////////////////////////////////////////////// ***CREDIT SECTION***
/////////////////////////////////////////////////
//
//
//
// Credit:                        eKam
// URL:                           <a href="https://community.tradestation.com/discussions/Topic.aspx?
TOPIC_ID=31411 " target="_blank">https://community.tradestation.com/discussions/Topic.aspx?
TOPIC_ID=31411 </a>
// TSWCodeAuthor:       eKam
// OtherCredit:
// OriginalDate:        2004.12.16
//
//////////////////////////////////////////////// *****JTM SECTION*****
```

select all

Bamboo

1258 Posts

Posted - 12/18/2004 11:01:56

JTM,

ELCollections only supports reading and writing of certain combinations of collections, and a map of maps is not one of them. MapSC.WriteFile can only write a map of lists. If you remove the WriteFile and ReadFile calls, the code will run without errors.

Also, in cases where WriteFile does work, you need to clear the map before reading it back, since ReadFile expects an empty map. I will revise the DLL code to clear the map automatically before reading a file if the map isn't empty.

A map (MapSC) of lists (ListN and/or ListS) is usually the best structure for working with tabular text files. The map keys represent the column headings in the text file, and each associated list represents the values in that column. This is the structure that ADE uses (see the "Concepts" section in the ADE document).

Next Page

8 Pages 1 2 3 4 5 6 7 8 > »

Quick Reply

Message:

Submit Reply Preview Reply Reset Form