

VisualAPL with Strong Data Typing Yields Significant Performance Advantages

Timing of operations can often be used to compare alternate implementations of a solution to an application requirement. For the purposes of this illustration, thousands of timing trials of the addition of two vectors of one million floating point numbers was observed. The performance of an APL+Win (Win32) function using dynamic data types is compared to the performance of VisualAPL (.Net) functions using strong data types and using dynamic data types. The timing values illustrated were obtained using the same machine and using functions designed in essentially the same way in each environment. A more skillful programmer may be able to improve the performance of the illustrated functions. Timing other operations might yield different performance results. The VisualAPL solution provided with this analysis may be easily modified to time other operations. The timing results in milliseconds are summarized below:

	Dynamically Typed - Debug	Dynamically Typed - Release	Strongly Typed - Debug	Strongly Typed - Release
APL+Win	15004	15004	Not available in APL+Win	Not available in APL+Win
VisualAPL	17774	16202	10431	6614

Relative performance is summarized below (smaller values are more desirable):

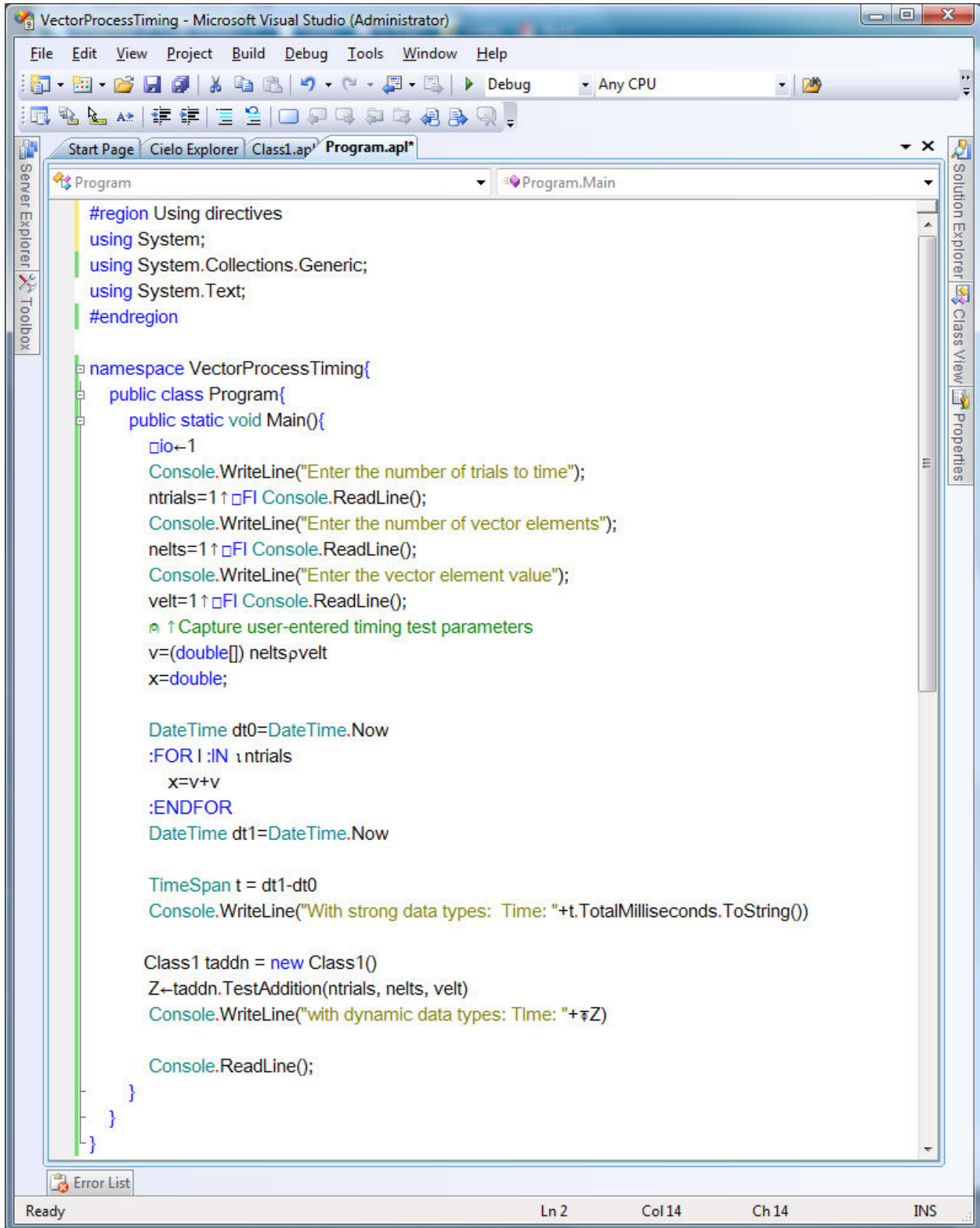
	Dynamically Typed - Debug	Dynamically Typed - Release	Strongly Typed - Debug	Strongly Typed - Release
VisualAPL time compared to APL+Win time	118.5%	107.9%	69.5%	44.1%

	Debug	Release
VisualAPL Strong Typing compared to VisualAPL Dynamic Typing	58.7%	40.8%

These results suggest that for this operation VisualAPL using dynamic data typing has a slight performance disadvantage compared to APL+Win which supports only dynamic data typing.

These results also suggest that for this operation VisualAPL using strong data typing has a very significant performance advantage (more than twice as fast) over APL+Win or over VisualAPL using dynamic data typing.

The VisualAPL Console project implements the timing experiment with strong data types:



The screenshot shows the Microsoft Visual Studio (Administrator) interface. The title bar reads "VectorProcessTiming - Microsoft Visual Studio (Administrator)". The menu bar includes File, Edit, View, Project, Build, Debug, Tools, Window, and Help. The toolbar shows various icons for file operations, editing, and debugging. The "Start Page" tab is active, showing a "Program" view with "Program.Main" selected. The main editor displays the source code of "Program.apl". The code is as follows:

```
#region Using directives
using System;
using System.Collections.Generic;
using System.Text;
#endregion

namespace VectorProcessTiming{
    public class Program{
        public static void Main(){
            cl←1
            Console.WriteLine("Enter the number of trials to time");
            ntrials←1↑⌊FI Console.ReadLine();
            Console.WriteLine("Enter the number of vector elements");
            nelts←1↑⌊FI Console.ReadLine();
            Console.WriteLine("Enter the vector element value");
            velt←1↑⌊FI Console.ReadLine();
            ⌘ Capture user-entered timing test parameters
            v=(double[]) nelts p velt
            x=double;

            DateTime dt0=DateTime.Now
            :FOR I :IN 1 ntrials
                x=v+v
            :ENDFOR
            DateTime dt1=DateTime.Now

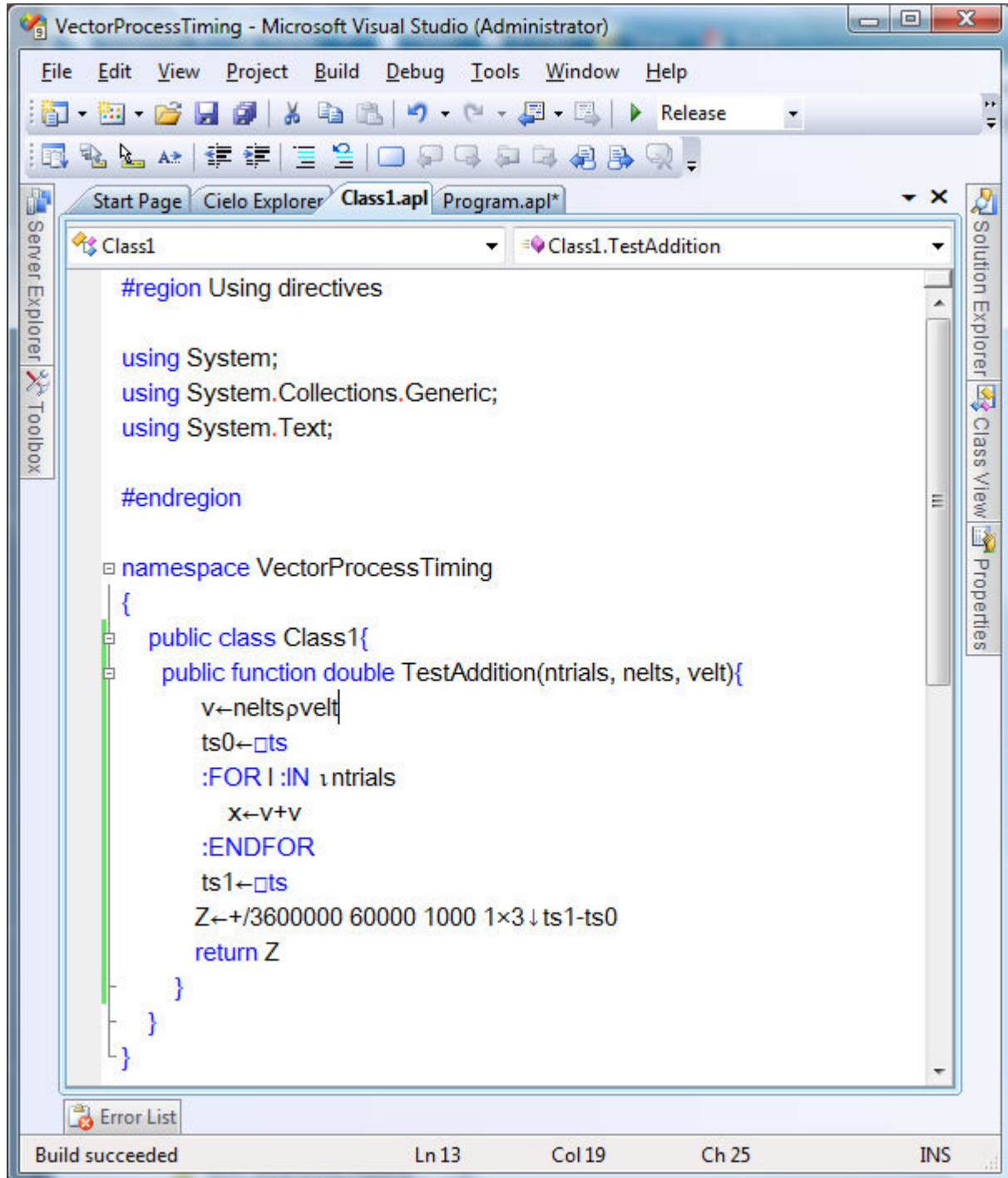
            TimeSpan t = dt1-dt0
            Console.WriteLine("With strong data types: Time: "+t.TotalMilliseconds.ToString())

            Class1 taddn = new Class1()
            Z←taddn.TestAddition(ntrials, nelts, velt)
            Console.WriteLine("with dynamic data types: Time: "+Z)

            Console.ReadLine();
        }
    }
}
```

The status bar at the bottom shows "Ready", "Ln 2", "Col 14", "Ch 14", and "INS".

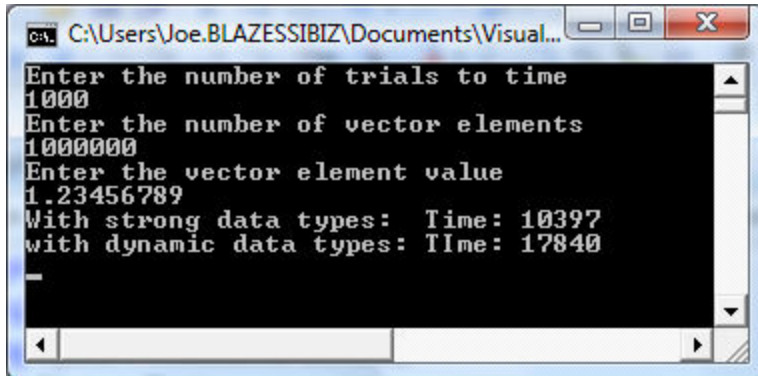
The TestAddition() function, which is also called by the VisualAPL console project, implements the timing experiment using dynamic data typing:



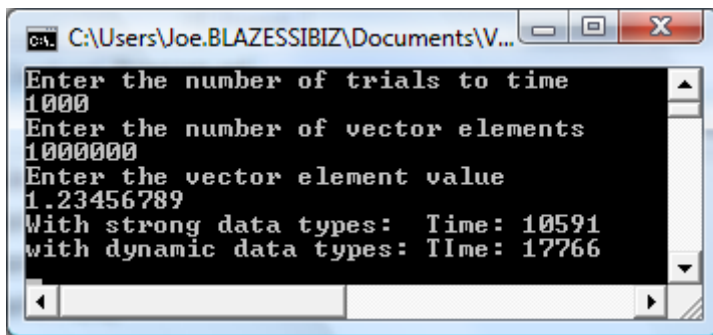
```
VectorProcessTiming - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Tools Window Help
... Release
Start Page Cielo Explorer Class1.apl Program.apl*
Class1 Class1.TestAddition
#region Using directives
using System;
using System.Collections.Generic;
using System.Text;
#endregion
namespace VectorProcessTiming
{
    public class Class1{
        public function double TestAddition(ntrials, nelts, velt){
            v←nelts*pvelt
            ts0←□ts
            :FOR I :IN 1 ntrials
                x←v+v
            :ENDFOR
            ts1←□ts
            Z←+/36000000 60000 1000 1×3↓ts1-ts0
            return Z
        }
    }
}
```

Build succeeded Ln 13 Col 19 Ch 25 INS

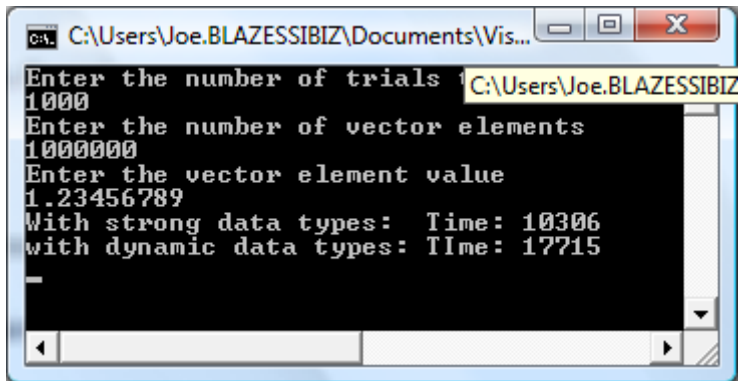
Typical output from the VisualAPL console project, run in “debug” mode, by using the Visual Studio 2008 “Debug > Start Debugging” menu item:



```
C:\Users\Joe.BLAZESSIBIZ\Documents\Visual...
Enter the number of trials to time
1000
Enter the number of vector elements
1000000
Enter the vector element value
1.23456789
With strong data types: Time: 10397
with dynamic data types: Time: 17840
-
```



```
C:\Users\Joe.BLAZESSIBIZ\Documents\V...
Enter the number of trials to time
1000
Enter the number of vector elements
1000000
Enter the vector element value
1.23456789
With strong data types: Time: 10591
with dynamic data types: Time: 17766
-
```

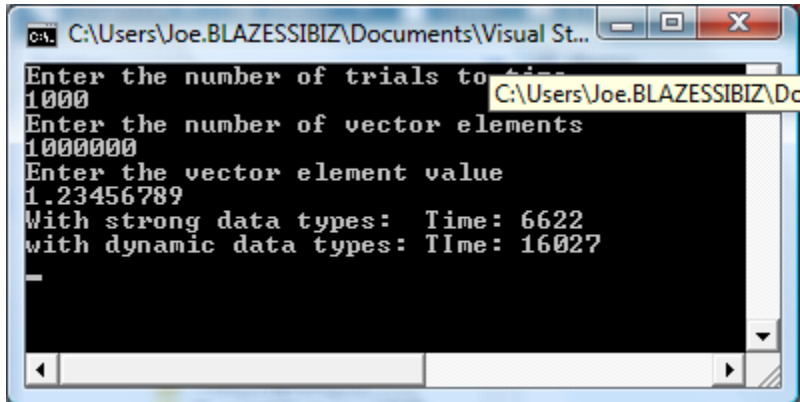


```
C:\Users\Joe.BLAZESSIBIZ\Documents\Vis...
Enter the number of trials to time 1 C:\Users\Joe.BLAZESSIBIZ
1000
Enter the number of vector elements
1000000
Enter the vector element value
1.23456789
With strong data types: Time: 10306
with dynamic data types: Time: 17715
-
```

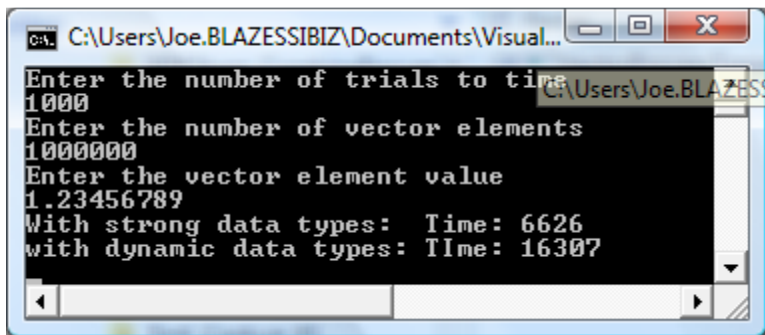
Average Time for 3000 VisualAPL trials in Debug mode:

Strong Types:	10431
Dynamic Types:	17774

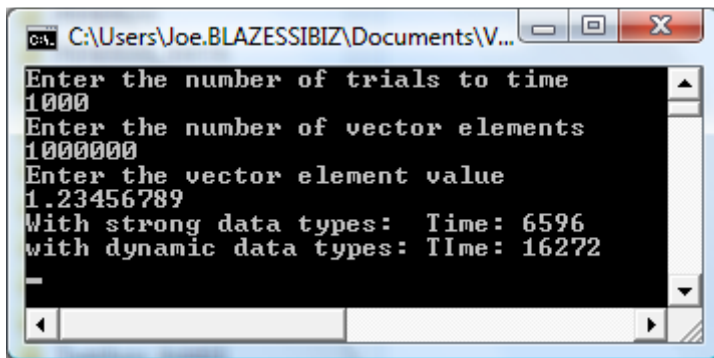
Typical output from the VisualAPL console project, run in “release” mode, by double-clicking the “...\VectorProcessTiming\bin\Release\VectorProcessTiming.exe” file:



```
C:\Users\Joe.BLAZESSIBIZ\Documents\Visual St...
Enter the number of trials to time
1000
Enter the number of vector elements
1000000
Enter the vector element value
1.23456789
With strong data types: Time: 6622
with dynamic data types: Time: 16027
```



```
C:\Users\Joe.BLAZESSIBIZ\Documents\Visual...
Enter the number of trials to time
1000
Enter the number of vector elements
1000000
Enter the vector element value
1.23456789
With strong data types: Time: 6626
with dynamic data types: Time: 16307
```



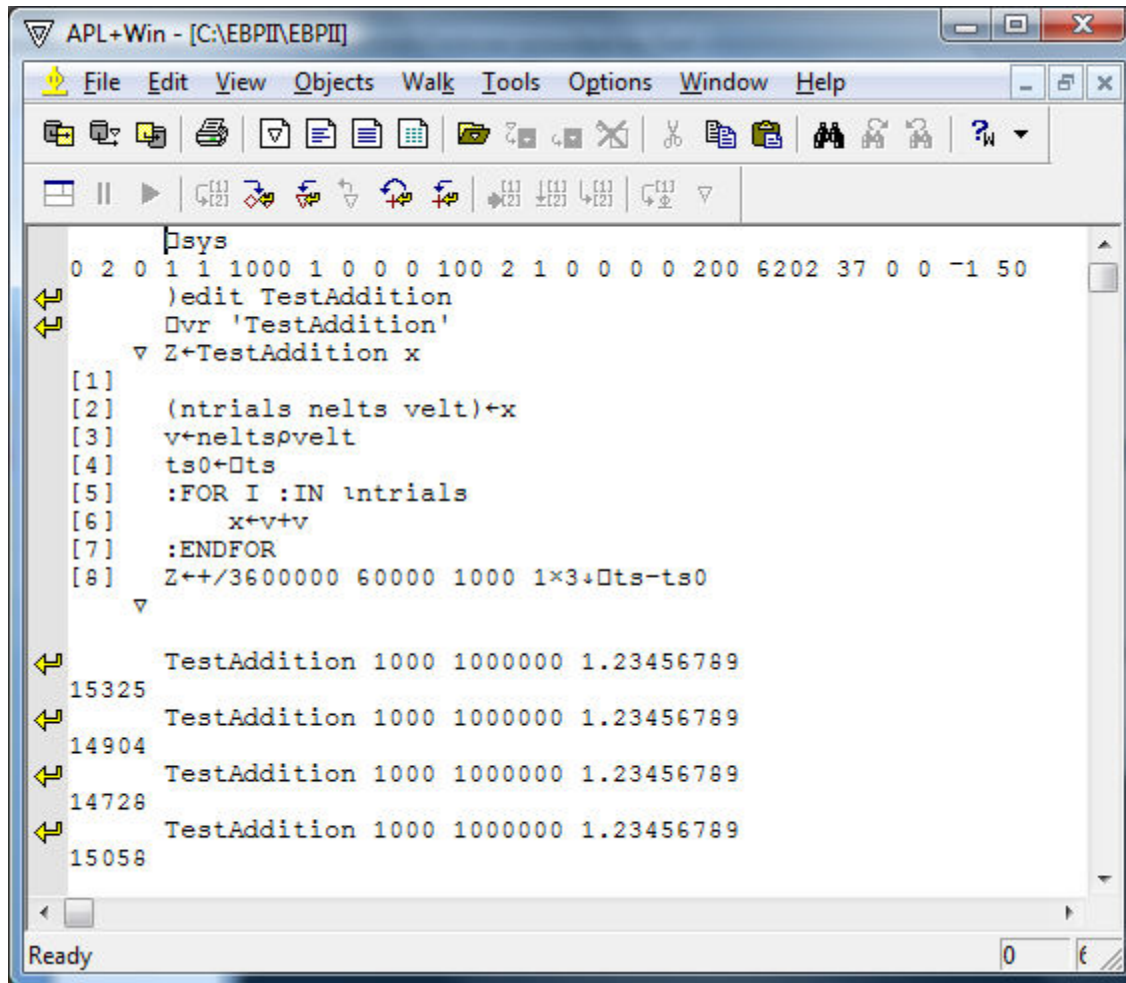
```
C:\Users\Joe.BLAZESSIBIZ\Documents\V...
Enter the number of trials to time
1000
Enter the number of vector elements
1000000
Enter the vector element value
1.23456789
With strong data types: Time: 6596
with dynamic data types: Time: 16272
```

Average Time for 3000 VisualAPL trials in Release mode:

Strong Types: 6614

Dynamic Types: 16202

APL+Win timing results on the same machine::



The screenshot shows the APL+Win IDE window titled "APL+Win - [C:\EBPII\EBPII]". The menu bar includes File, Edit, View, Objects, Walk, Tools, Options, Window, and Help. The toolbar contains various icons for file operations, editing, and execution. The main text area displays the following APL code and its output:

```
sys
0 2 0 1 1 1000 1 0 0 0 100 2 1 0 0 0 0 200 6202 37 0 0 -1 50
)edit TestAddition
)vr 'TestAddition'
▽ Z←TestAddition x
[1]
[2] (ntrials nelts velt)←x
[3] v←neltsρvelt
[4] ts0←⌈ts
[5] :FOR I :IN ntrials
[6]   x←v+v
[7] :ENDFOR
[8] Z←+/3600000 60000 1000 1×3+⌈ts-ts0
▽
TestAddition 1000 1000000 1.23456789
15325
TestAddition 1000 1000000 1.23456789
14904
TestAddition 1000 1000000 1.23456789
14728
TestAddition 1000 1000000 1.23456789
15058
```

The status bar at the bottom shows "Ready" and a numeric display with "0".

Average Time for 4000 APL+Win trials in Release (same as Debug) mode:

Strong Types: not available in APL+Win

Dynamic Types: 15004