**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):

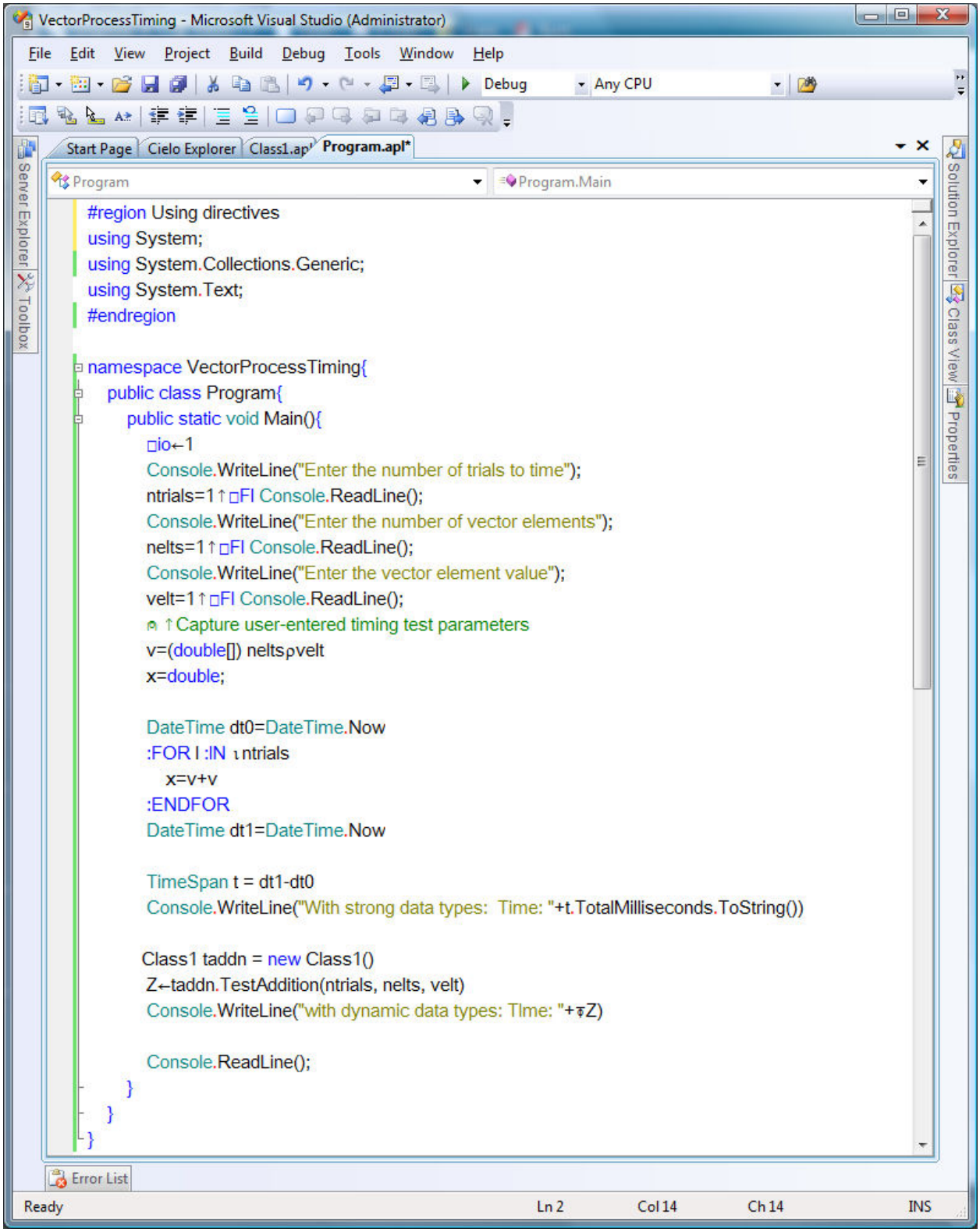
| | Dynamically Typed - Debug | Dynamically Typed - Release | Strongly Typed - Debug | Strongly Typed - Release |
|---|---|---|---|---|
| VisualAPL time compared to APL+Win time | 118.5% | 107.9% | 69.5% | 44.1% |

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

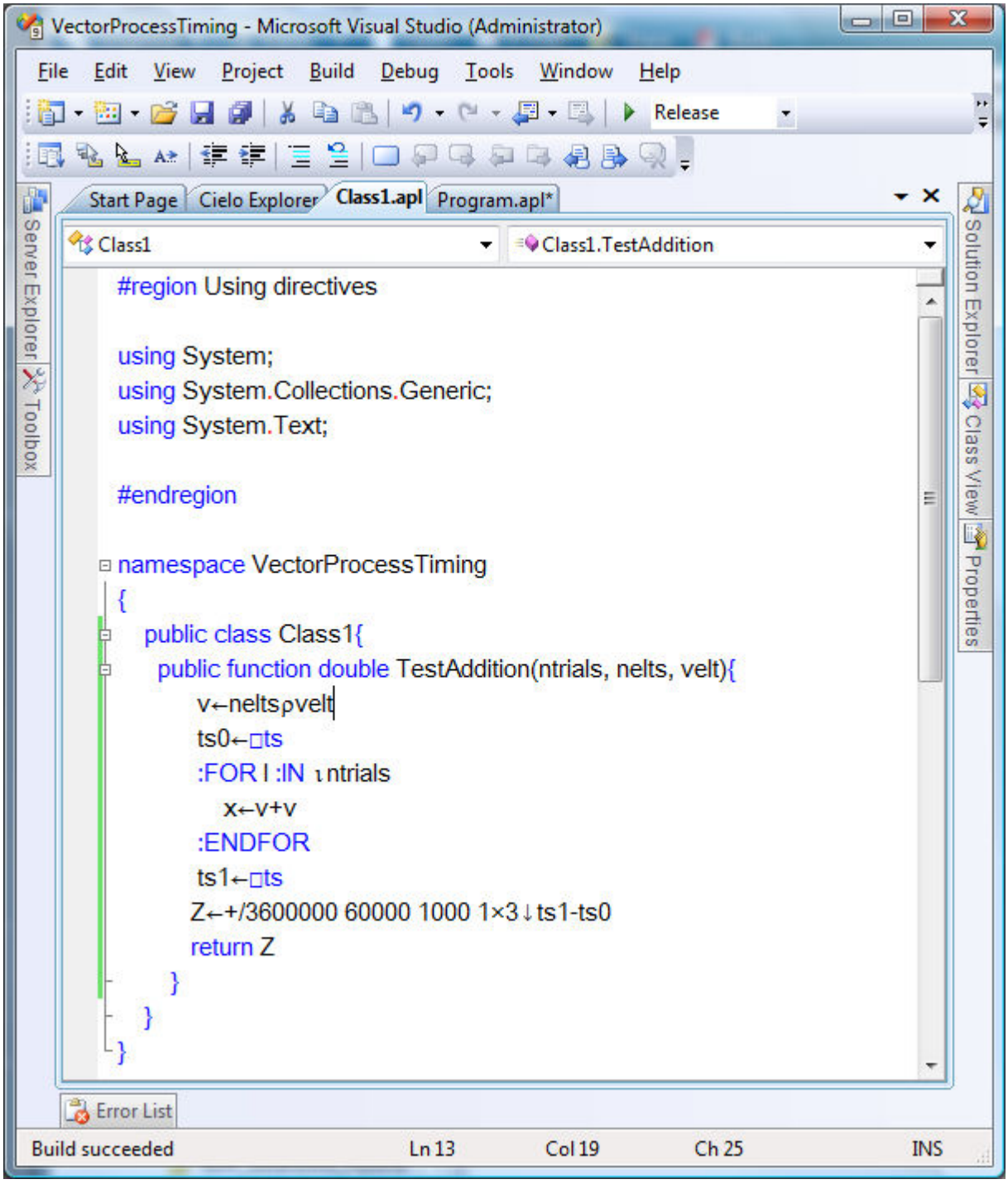
These results suggest that for this operation VisualAPL using dynamic data typing has a slight performance disadvantage compared to APL+Win which supports only dynamic data typing.

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.

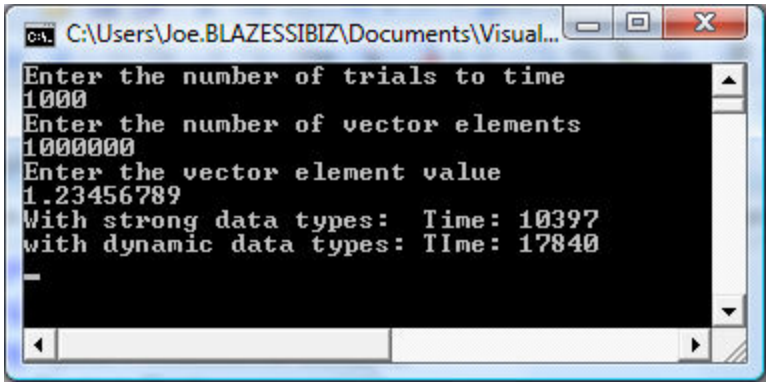
The VisualAPL Console project implements the timing experiment with strong data types:

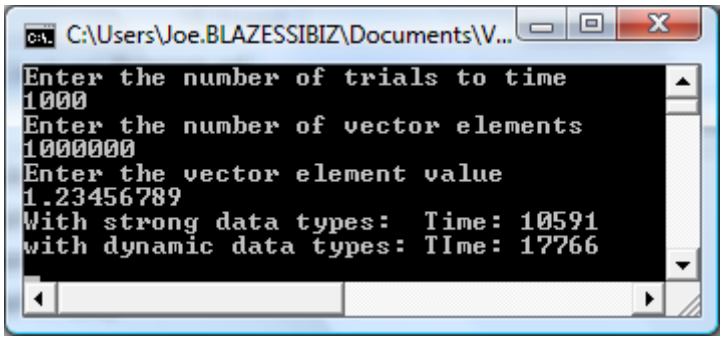


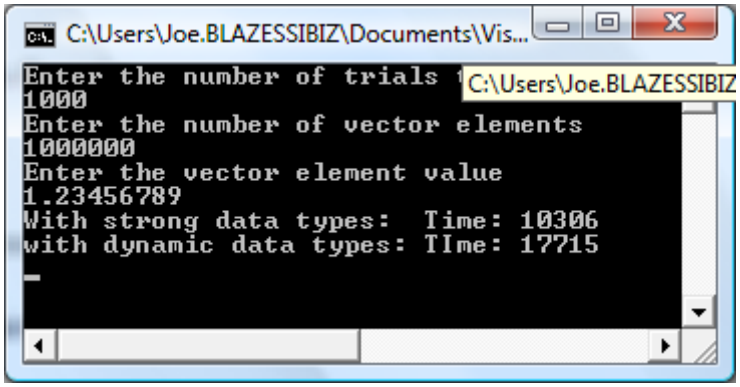
The TestAddition() function, which is also called by the VisualAPL console project, implements the timing experiment using dynamic data typing:



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



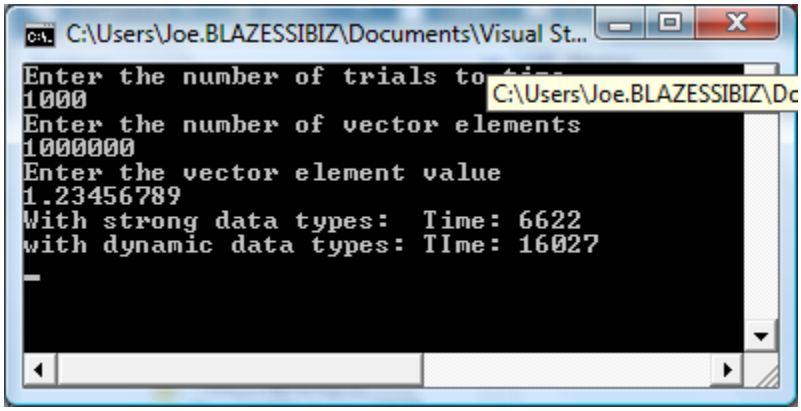



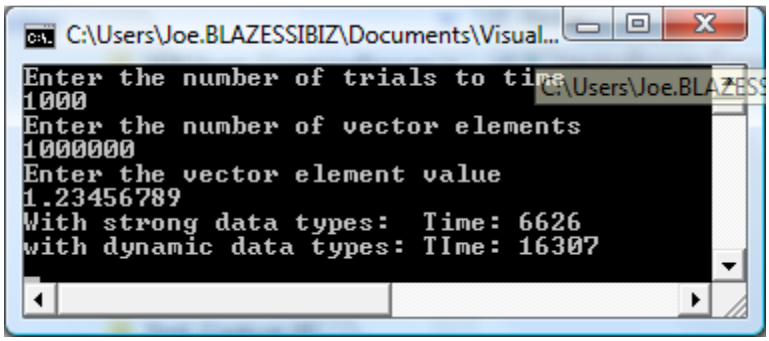



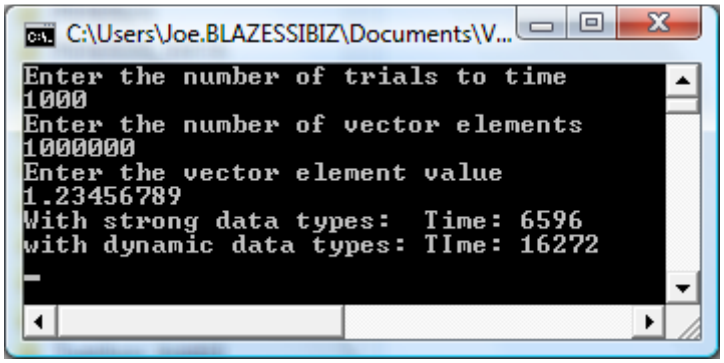

Average Time for 3000 VisualAPL trials in Debug mode:

| | |
|---|---|
| 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:



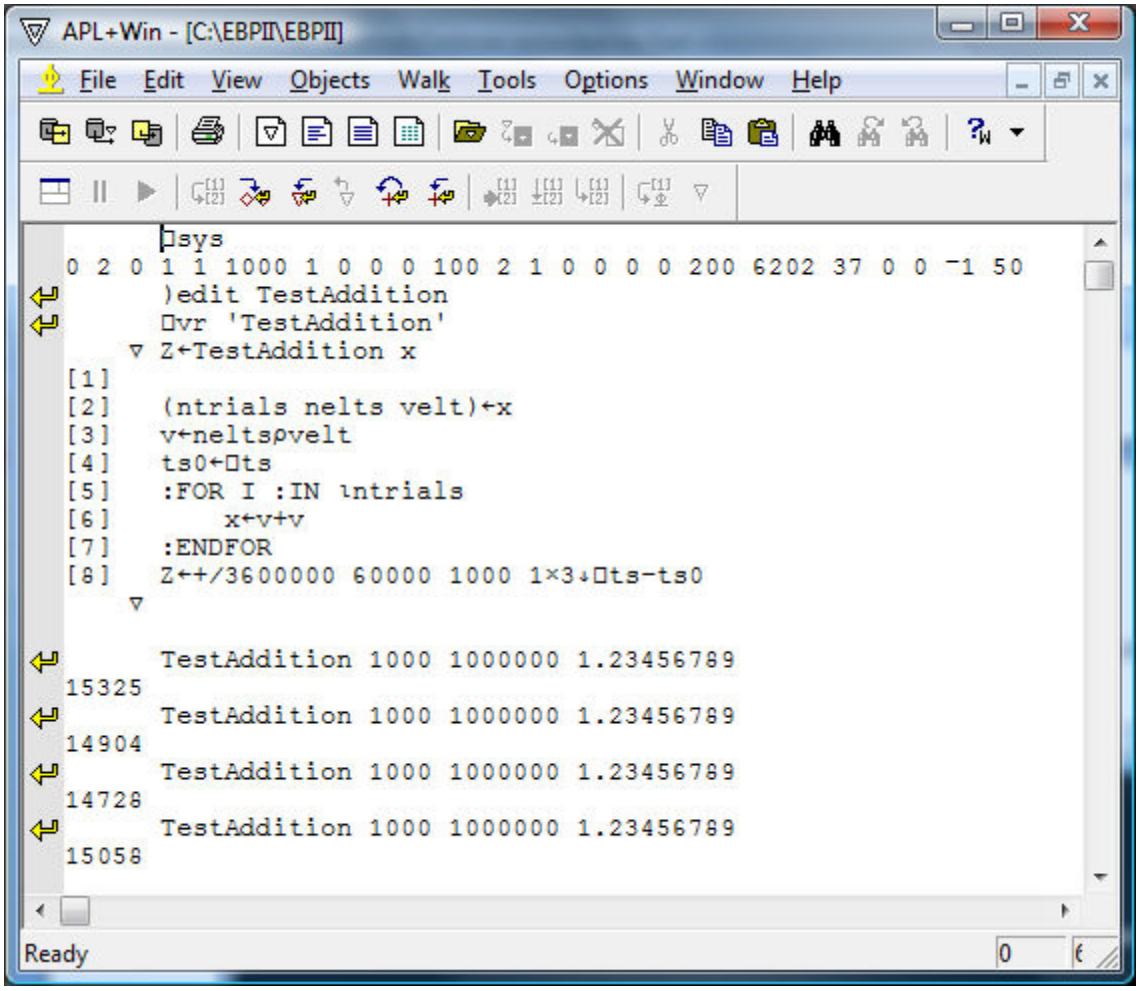




Average Time for 3000 VisualAPL trials in Release mode:

| | |
|---|---|
| Strong Types: | 6614 |
| Dynamic Types: | 16202 |

APL+Win timing results on the same machine::




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

Strong Types: not available in APL+Win

Dynamic Types: 15004