Converting APL+Win Code to VisualAPL Code: General

When converting APL+Win code to VisualAPL code, please keep the following issues in mind:

- The VisualAPL product development team is ready to help you perform conversion from APL+Win to VisualAPL of individual functions or an entire application. This consulting service is economically priced and will get your application system up and running quickly.

- VisualAPL is APL for the Microsoft .Net framework. It is a completely new implementation of APL, so it is not a replication of any particular APL or APL environment.

- VisualAPL is integrated with Microsoft Visual Studio, so you will need to learn a bit about Visual Studio. Visual Studio provides a complete IDE (interactive development environment) which is used by "mainstream" application system programmers. The same IDE applies to every .Net language, such as C#, VB.Net and VisualAPL. So tools, such as search & replace, are available directly in Visual Studio. VisualAPL uses these Visual Studio tools rather than implementing them in the proprietary manner of legacy APLs.

- Because VisualAPL was designed to be a peer of any other .Net programming language with seamless interoperation of projects among these languages, VisualAPL adopts some of the conventions required of all .Net languages including:

    - VisualAPL is inherently OOP (object oriented). Everything in VisualAPL is an object.

    - VisualAPL is inherently Unicode-based.

    - VisualAPL provides both the traditional APL "assignment by value" (using the "Alt+[" operator), but also the .Net standard "assignment by reference" (using the "=" operator).

    - Since .Net languages have defined the "assignment by reference" operator to be "=", VisualAPL uses the "approximately equal" operator ("Alt+5") for the APL comparison with comparison tolerance operator.

    - VisualAPL provides both the traditional APL function (operator) signature using the "Alt+g" operator and the .Net standard function "method" signature using the "function" keyword or the "Alt+f" operator.

    - VisualAPL adopts the .Net standard for default localization and explicit global attributes where needed rather than the ‘dynamic localization’ of legacy APLs. In VisualAPL functions, the “;Local1;Local2;…” header to define local variables and functions is no longer necessary. Variables and functions can be marked as global, public, etc., using the standard .Net keywords.

      The VisualAPL “Application Shared Datastore” makes the specification and shared-utilization of global variables across .Net classes easy and effective.

    - VisualAPL uses the Microsoft JIT (just-in-time) compiler which produces CIL (common intermediate language) code which runs in the Microsoft CLR (common language runtime) like any other .Net language. Therefore VisualAPL inherently includes a “manifest” of required resources and produces fully “managed” code like any other .Net language. Thus, there is no proprietary APL interpreter needed for VisualAPL.

    - As a .Net language VisualAPL provides access to all the .Net data types, including, of course, the traditional APL double, integer, Boolean and string data types.

- VisualAPL provides for APL+Win statement and function conversion using either the "Edit > Paste APL+Win" method or the ‘uniout’ assembly and workspace for ‘bulk’ conversions. After using these methods to convert APL+Win code to VisualAPL code, so manual intervention by the VisualAPL programmer is necessary.

- VisualAPL class libraries can be easily exposed as COM which can be accessed by the ActiveX interface of APL+Win.

- VisualAPL provides an interactive, immediate mode session called the Cielo Explorer which also supports script creating, editing, running and saving which is analogous to the legacy APL ‘session’.

- VisualAPL continues to provide dynamic data typing, but also supports strong (explicit) data typing for the ultimate in performance.

- VisualAPL fully supports the traditional APL “execute” operator to execute string expressions.

- Traditional APL, VisualAPL enhanced APL and C# syntax can be used in VisualAPL code. The wealth of C# examples on the Web can be utilized immediately in VisualAPL code.

- Because of the enhanced security requirements of the Microsoft .Net framework, certain unsafe programming features sometimes found in legacy APL implementations are not available in VisualAPL. Generally there are superior .Net substitutes for these deprecated options. The []wcall and []na operators are not possible in a "managed" code environment, but all of the vast .Net Framework is inherently available to the VisualAPL programmer. The .Net Framework is a significant improvement over the legacy Win32 API (application programming interface) in terms of performance, features and documentation.

- VisualAPL provides extensive support for the legacy APL []wi tools, however, because of significant Microsoft improvements in GUI design and development tools, it is recommended that []wi-based GUI be replaced with the Microsoft WPF (Windows Presentation Foundation)-based GUI.

- VisualAPL provides support for native file access, e.g. []nread, etc. VisualAPL also provides for a new and enhanced APL "component" file system. The VisualAPL product development team recommends that, where possible, proprietary format file-based application implementations be avoided in favor of using mainstream data storage approaches, such as Microsoft SQL Server. The .Net Framework provides superior tools for accessing such mainstream databases both locally and over the web.