Converting APL+Win Code to VisualAPL Code – Bulk Method

Ad hoc conversion of APL+Win code to VisualAPL code is easy using APL+Win "Edit > Copy" and VisualAPL "Edit > Paste APL+Win", however it may not be efficient for a larger group of APL+Win functions. For "bulk" conversions of a group of APL+Win functions, the "AplNext.Utils.apl2unicode.dll" is installed and registered as a .Net assembly exposed as COM on the programmer's machine when VisualAPL is installed.

The" AplNext.Utils.apl2unicode.dll" is designed to convert one or more APL+Win functions to VIsualAPL format. This conversion involves several elements:

- The APL+Win ∇ ("Alt+g") function signature prefix is preserved.
- The standard Visual Studio function code block delimiters "{…}" are added.
- The APL+Win "comparision-tolerance-sensitive equals operator" ("Alt+5") is replaced with the VisualAPL "comparison-tolerance-sensitive equals operator" ≈("Alt+5").
- The proprietary APL+Win font is converted to the Unicode font.

The method is best illustrated by the following example:

1. Start an APL+Win session and load the functions to be converted to VisualAPL in to the current workspace.
2. Copy the "uniout" and "unifnout" functions from the VisualAPL-provided "uniout.w3" workspace in to the current workspace.
3. Check that the VisualAPL-provided "AplNext.Utils.apl2unicode.dll" is properly registered as a COM object on the APL+Win programmer's machine by using the APL+Win statement:

   '#' []wi 'XInfo' 'AplNext.APL2UNICODE'

   This statement should return:

   AplNext.APL2UNICODE ActiveObject AplNext.Utils.apl2unicode.apl2uni

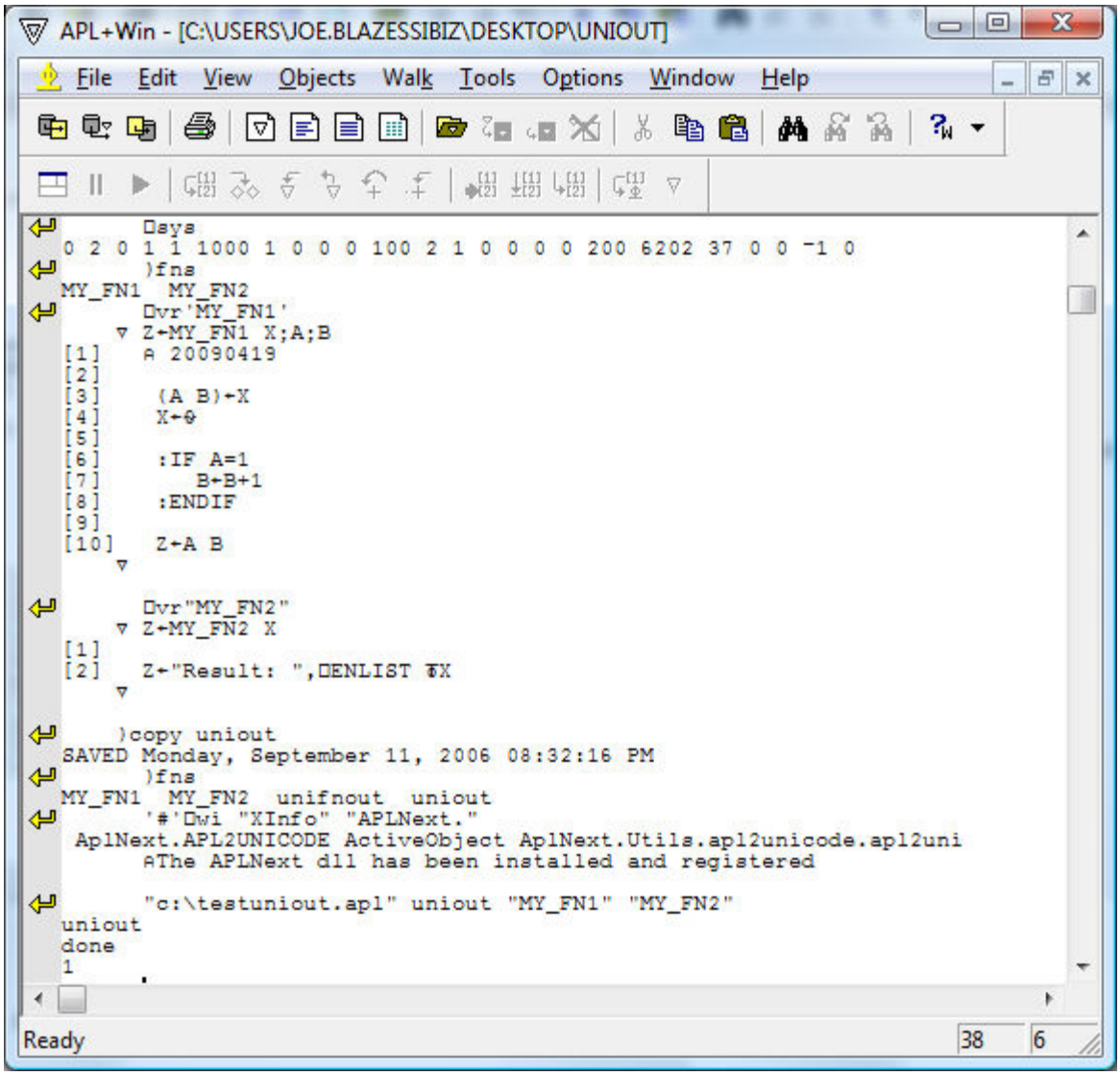
   If this result is not obtained install VisualAPL on the APL+Win programmer's machine.

4. In the APL+Win session execute the 'uniout' function to convert the selected APL+Win functions to VisualAPL functions:

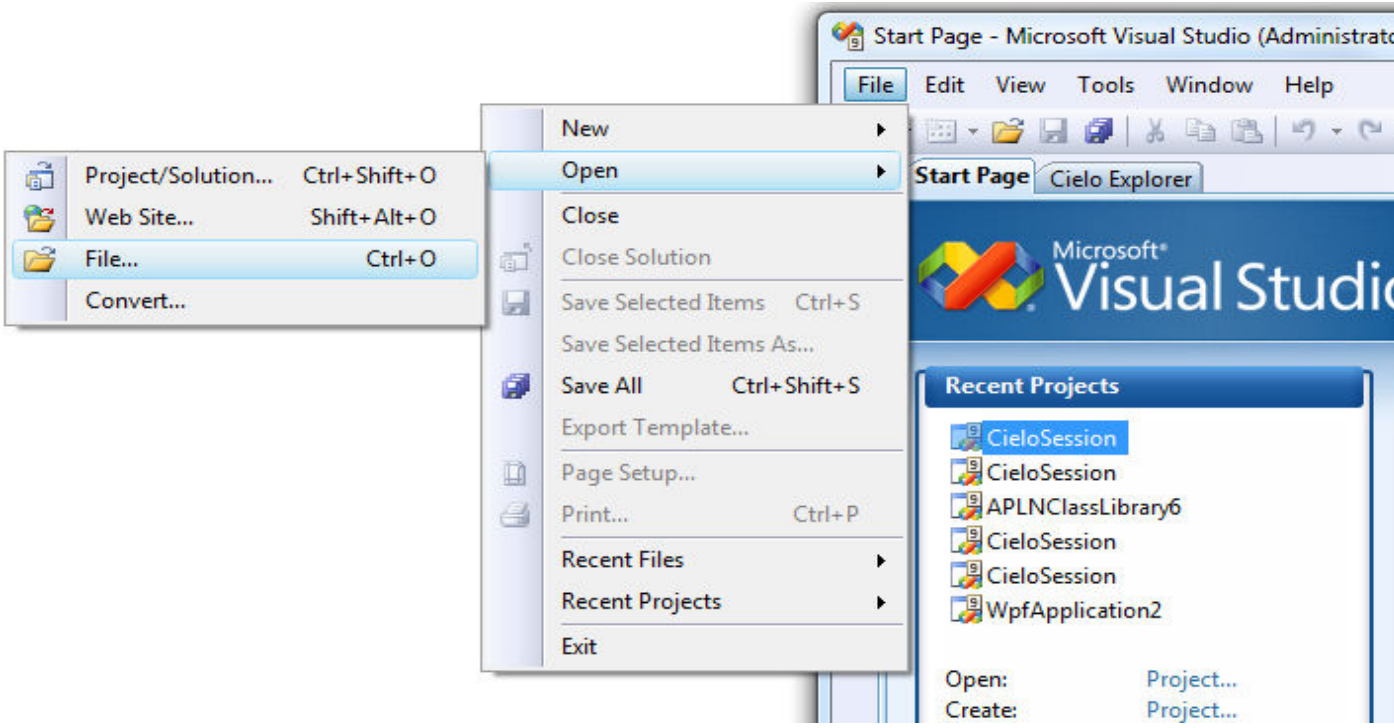
   Out_Path uniout "APLWinFn1" … "APLWinFnN"

   The left argument of the "uniout" function, Out_Path, should be selected with a filename format of "disk:\subdir…\filename.apl", where the ".apl" file extension will permit VisualAPL to directly load the resulting file containing the APL+Win functions converted to VisualAPL functions.

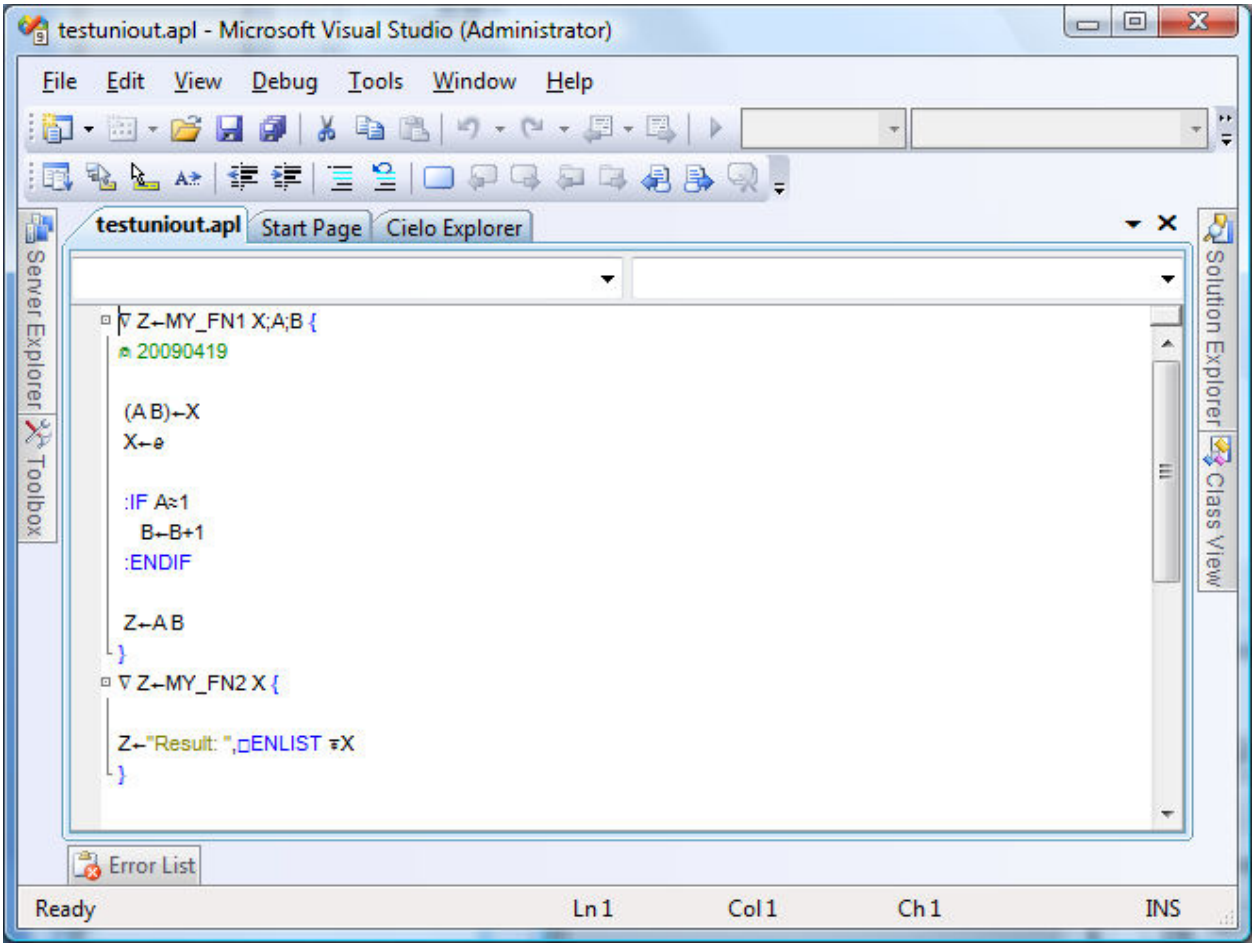
   The right argument of the "uniout" function, is an array of the APL+Win function names which are to be converted to VisualAPL functions.


APL+Win - [C:\USERS\JOE.BLAZESSIBIZ\DESKTOP\UNIOUT]
File Edit View Objects Walk Tools Options Window Help
      ⎕sys
0 2 0 1 1 1000 1 0 0 0 100 2 1 0 0 0 0 200 6202 37 0 0 ¯1 0
)fns
MY_FN1 MY_FN2
      ⎕vr'MY_FN1'
    ▽ Z←MY_FN1 X;A;B
[1]   ⍝ 20090419
[2]
[3]   (A B)←X
[4]   X←⍬
[5]
[6]   :IF A=1
[7]   B←B+1
[8]   :ENDIF
[9]
[10]   Z←A B
    ▽
      ⎕vr"MY_FN2"
    ▽ Z←MY_FN2 X
[1]
[2]   Z←"Result: ",⎕ENLIST ⍕X
    ▽
)copy uniout
SAVED Monday, September 11, 2006 08:32:16 PM
)fns
MY_FN1 MY_FN2 unifnout uniout
      '#'⎕wi "XInfo" "APLNext."
 AplNext.APL2UNICODE ActiveObject AplNext.Utils.apl2unicode.apl2uni
      ⍝The APLNext dll has been installed and registered
      "c:\testuniout.apl" uniout "MY_FN1" "MY_FN2"
uniout
done
1
Ready
38
6

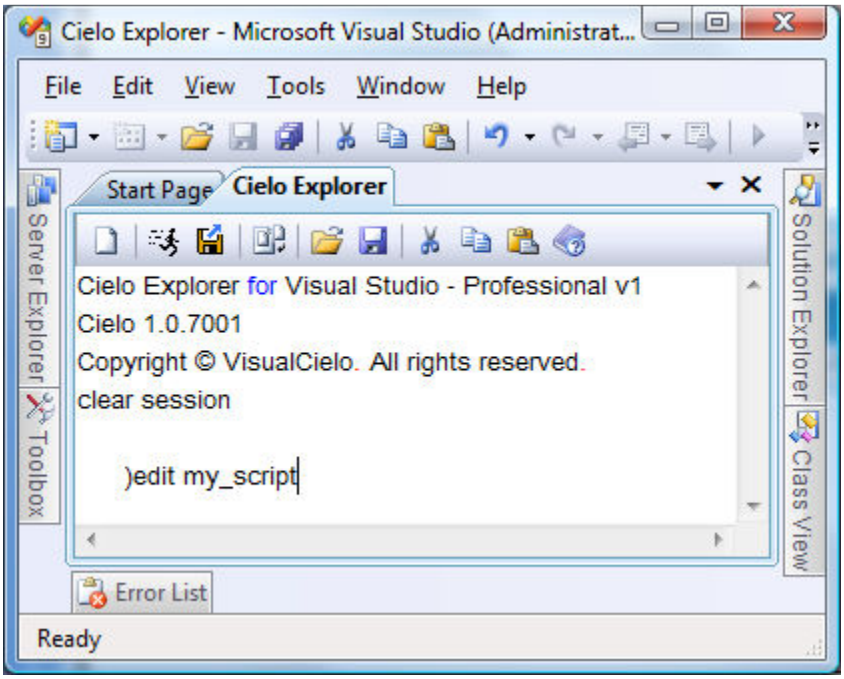
5. After the processing of the “uniout” function is complete, start a Visual Studio session and use the “File > Open” dialog to open the “.apl” format file created by the “uniout” function.



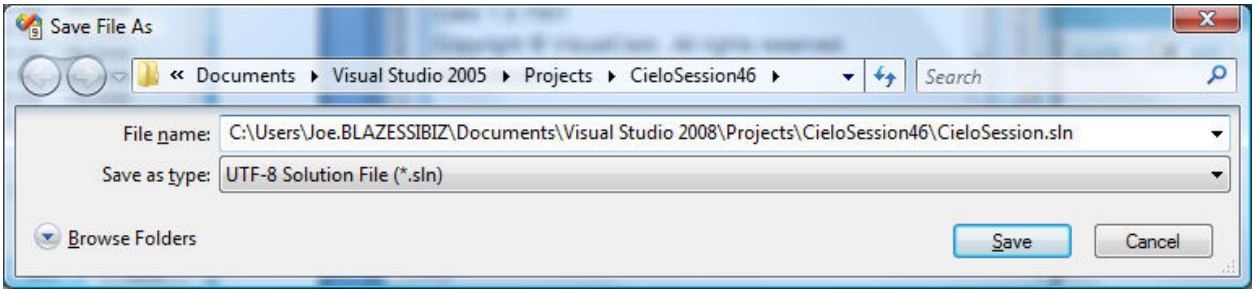
6. Browse to the .apl-format file created by the “uniout” function and open it in VisualAPL. After opening the “.apl” format file in VisualAPL, the converted functions can be incorporated into a VisualAPL class library (.Net assembly) or into a VisualAPL Cielo Explorer script, as desired by the VisualAPL programmer.



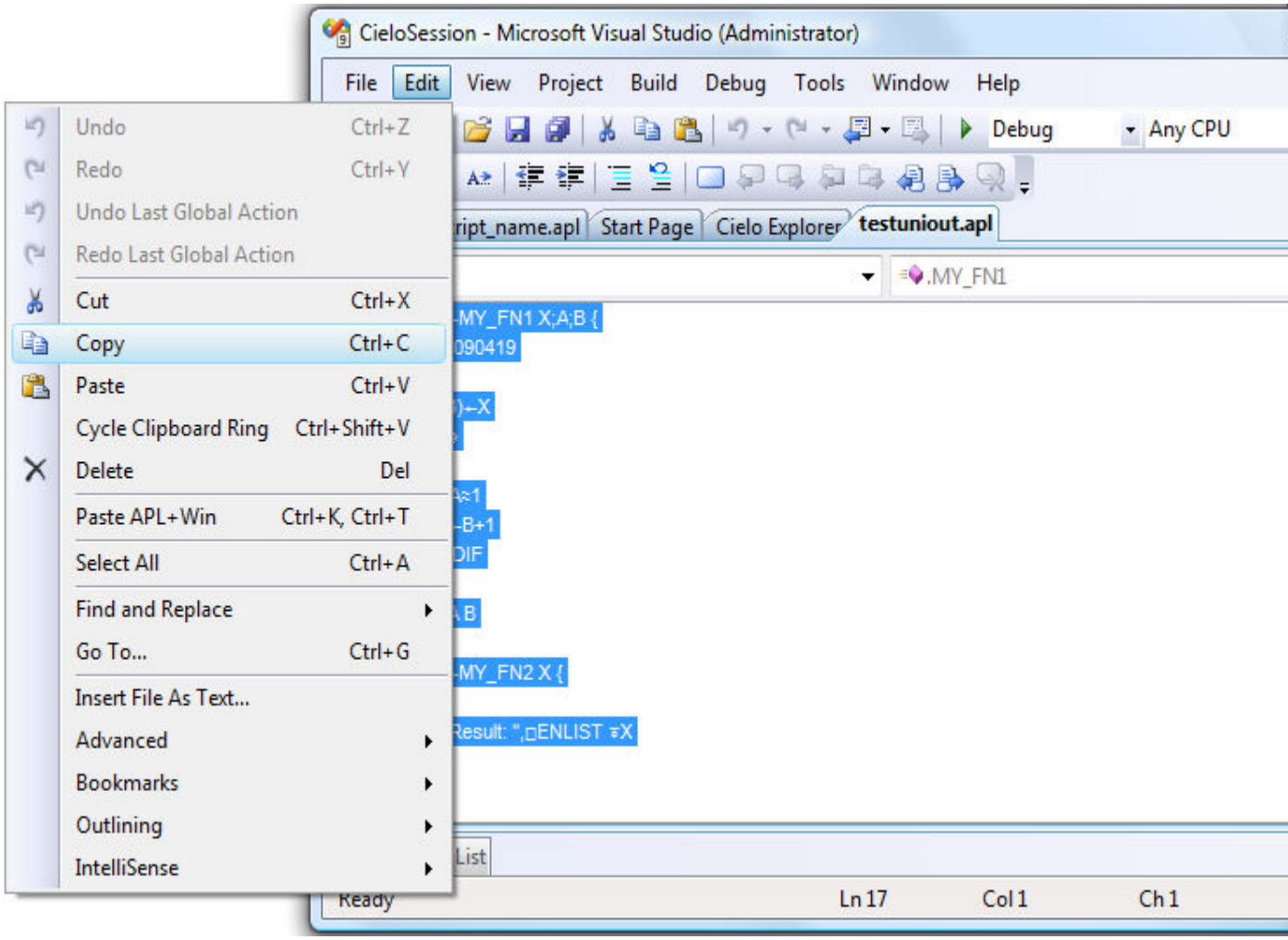
7. **To incorporate the converted APL+Win functions into a VisualAPL Cielo Explorer script**
    a. Open a Cielo Explorer session and use the “)edit my_script_name” command to create a new VisualAPL Cielo Explorer script:



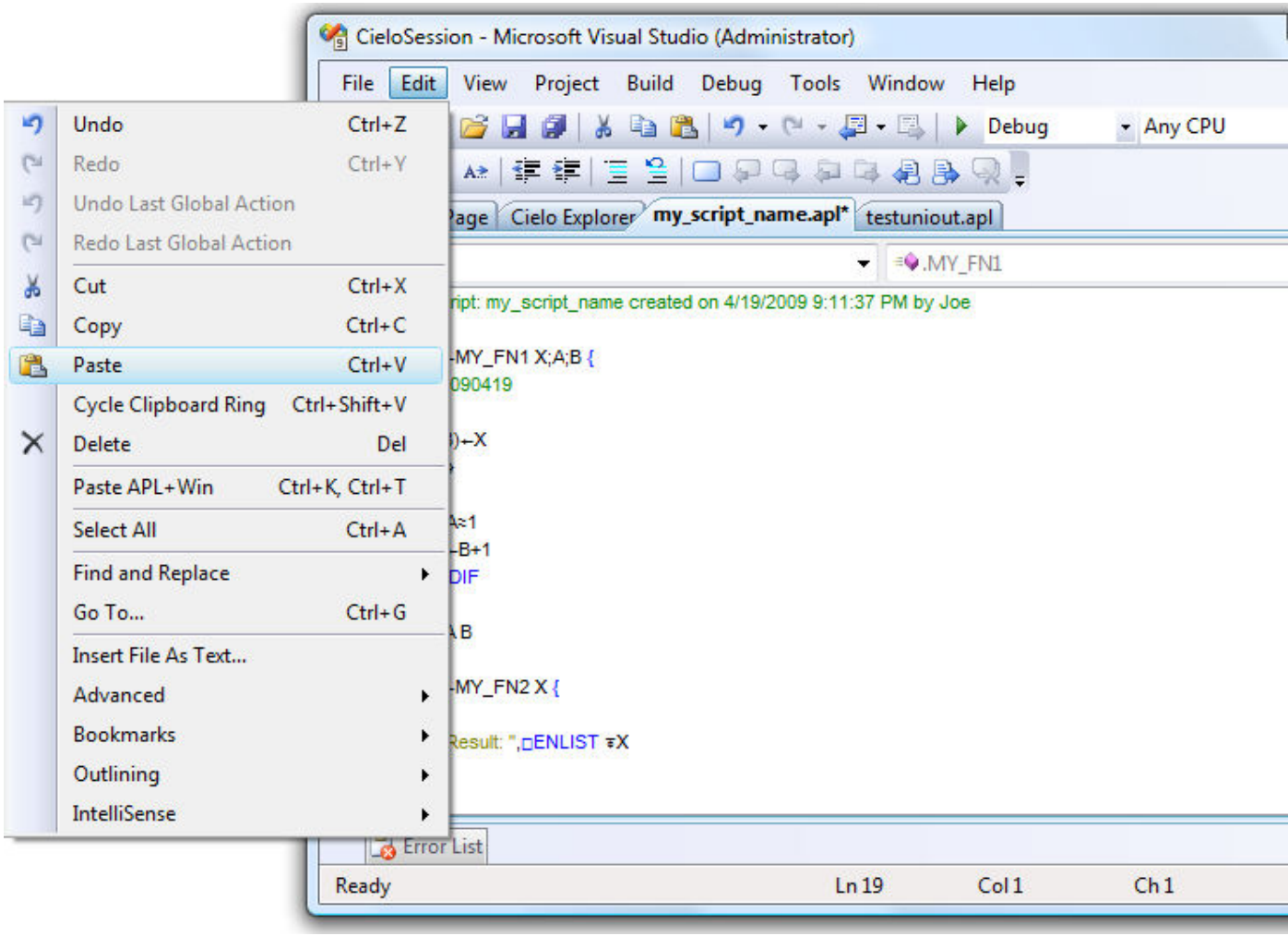
    b. Select the path where the VisualAPL Cielo Explorer script file will be saved:



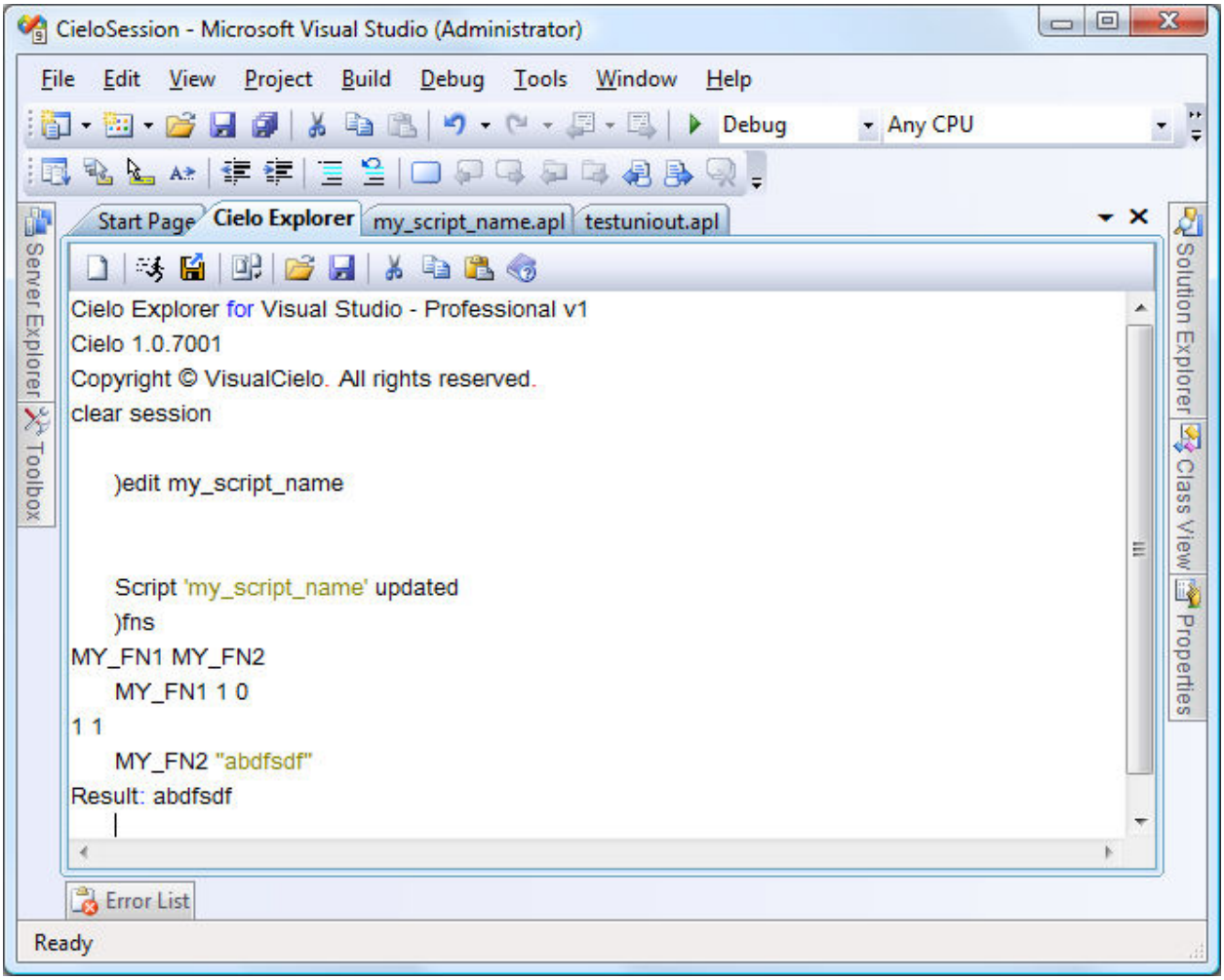
c. In the .apl-format file window containing the converted APL+Win functions, use the Visual Studio “Edit > Select All” and “Edit > Copy” menu items to place the selected Unicode text into the Windows Clipboard:



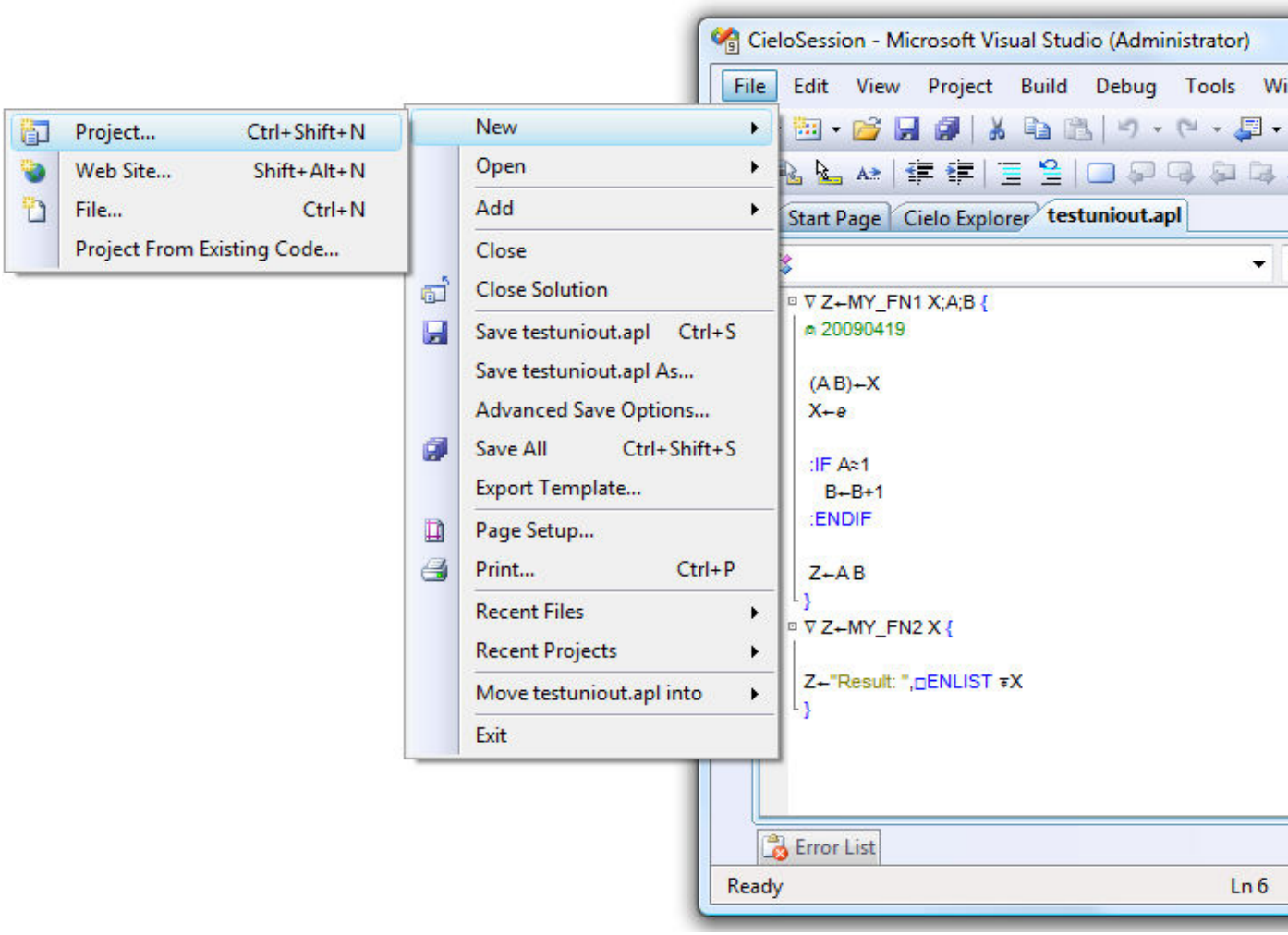
d. In the VisualAPL Cielo Script window, use the Visual Studio “Edit > Paste” menu option to put the converted APL+Win functions into the script:



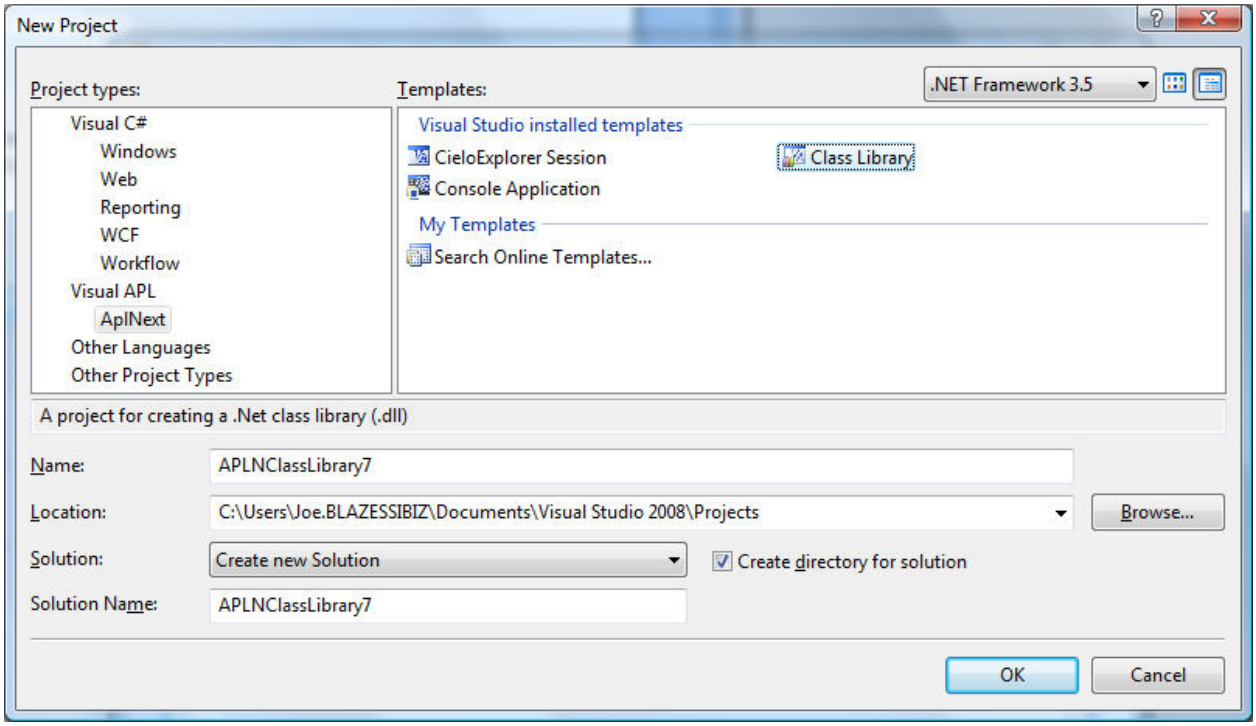
e. In the VisualAPL Cielo Explorer script window use the “Ctrl+E+E” keyboard shortcut to close the script editor and return to the Cielo Explorer session. Use the “)fns” command in the session to verify that the converted functions have been defined in the session from the script and try out the converted functions:



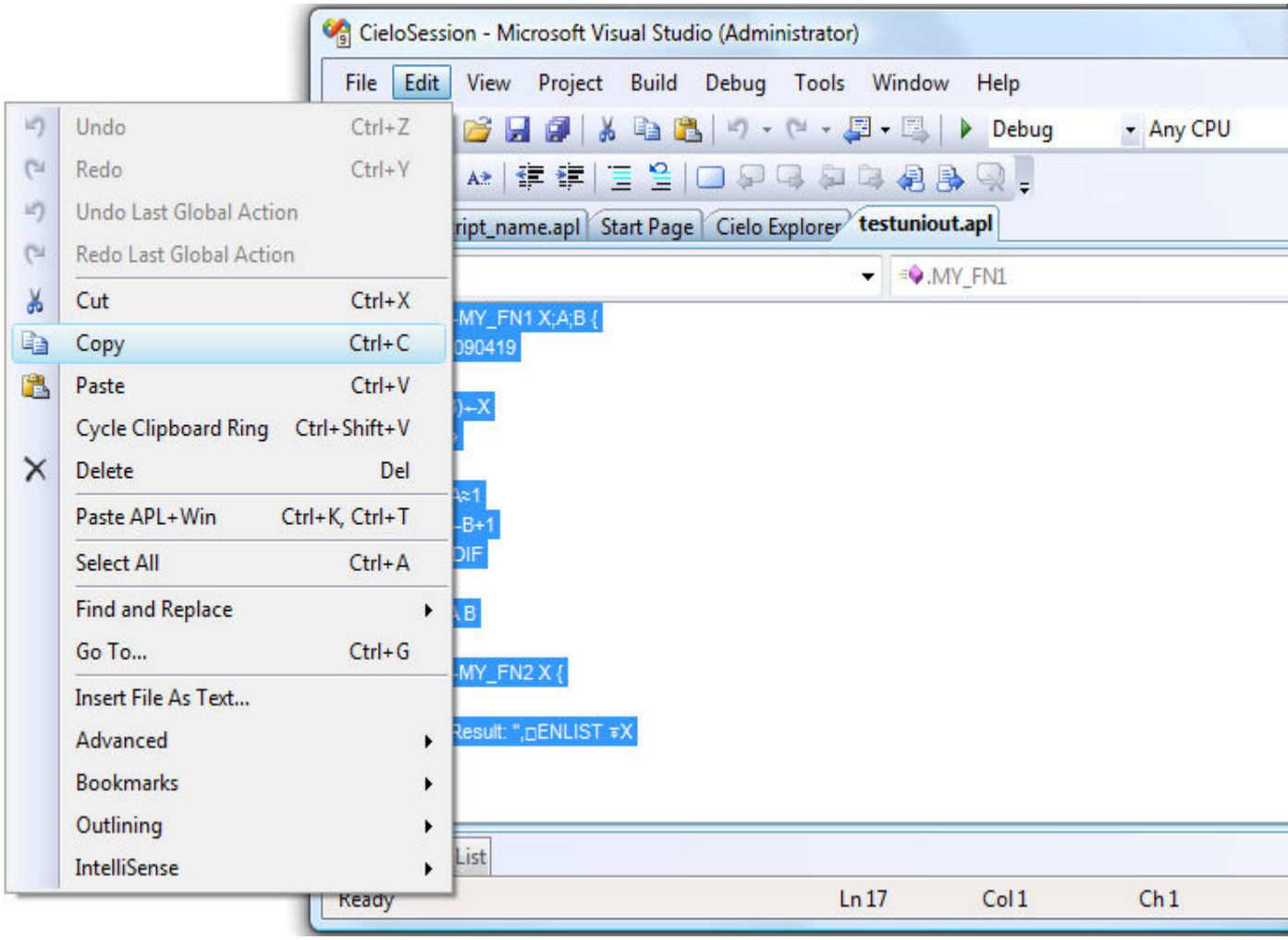
8. **To incorporate the converted APL+Win functions into a VisualAPL class library**
    a. **U**se the Visual Studio “File > New > Project” to present the VisualAPL new project dialog:



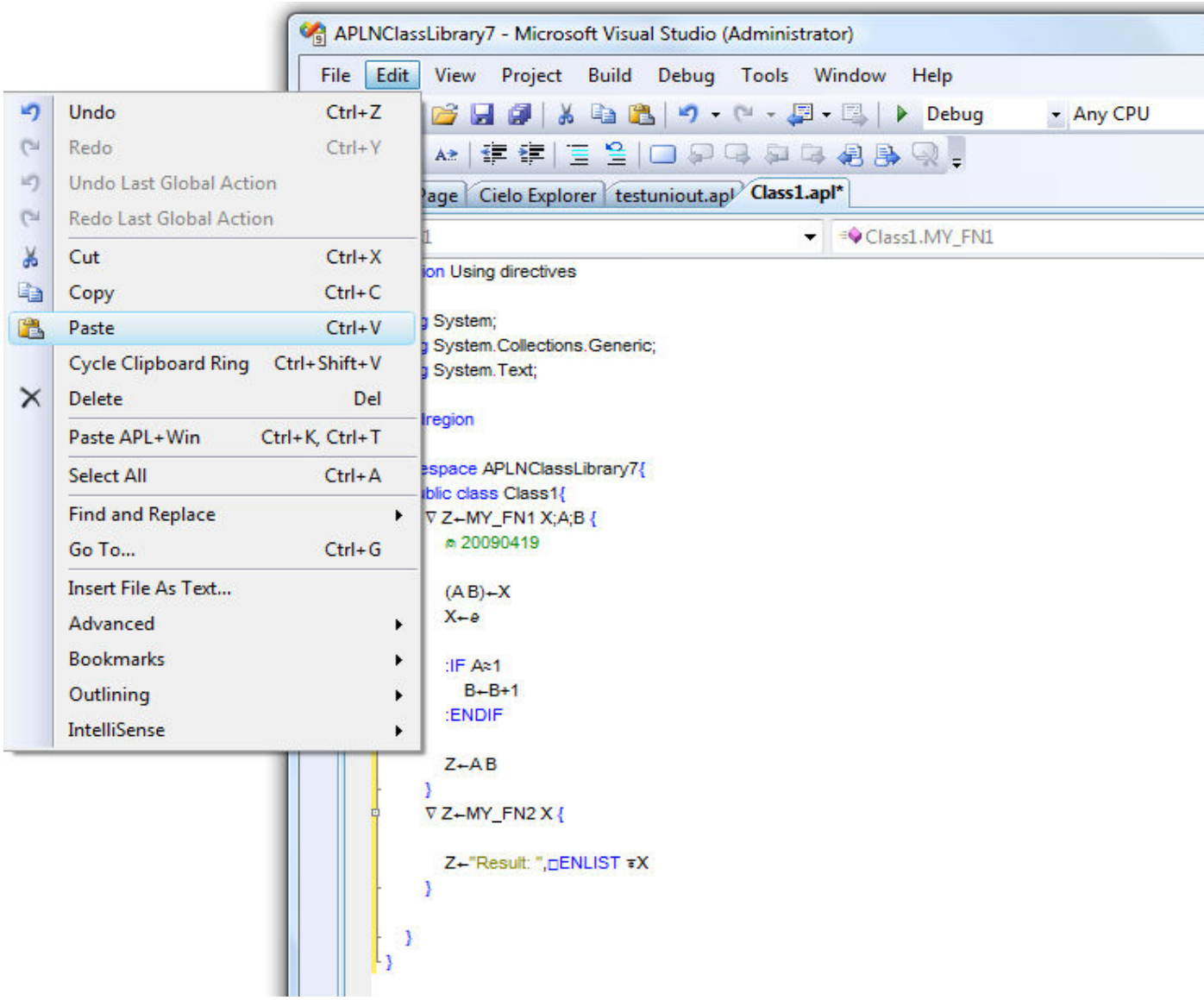
b. Create the new VisualAPL class library project, selecting the namespace name and the path where the class library project files will be saved:



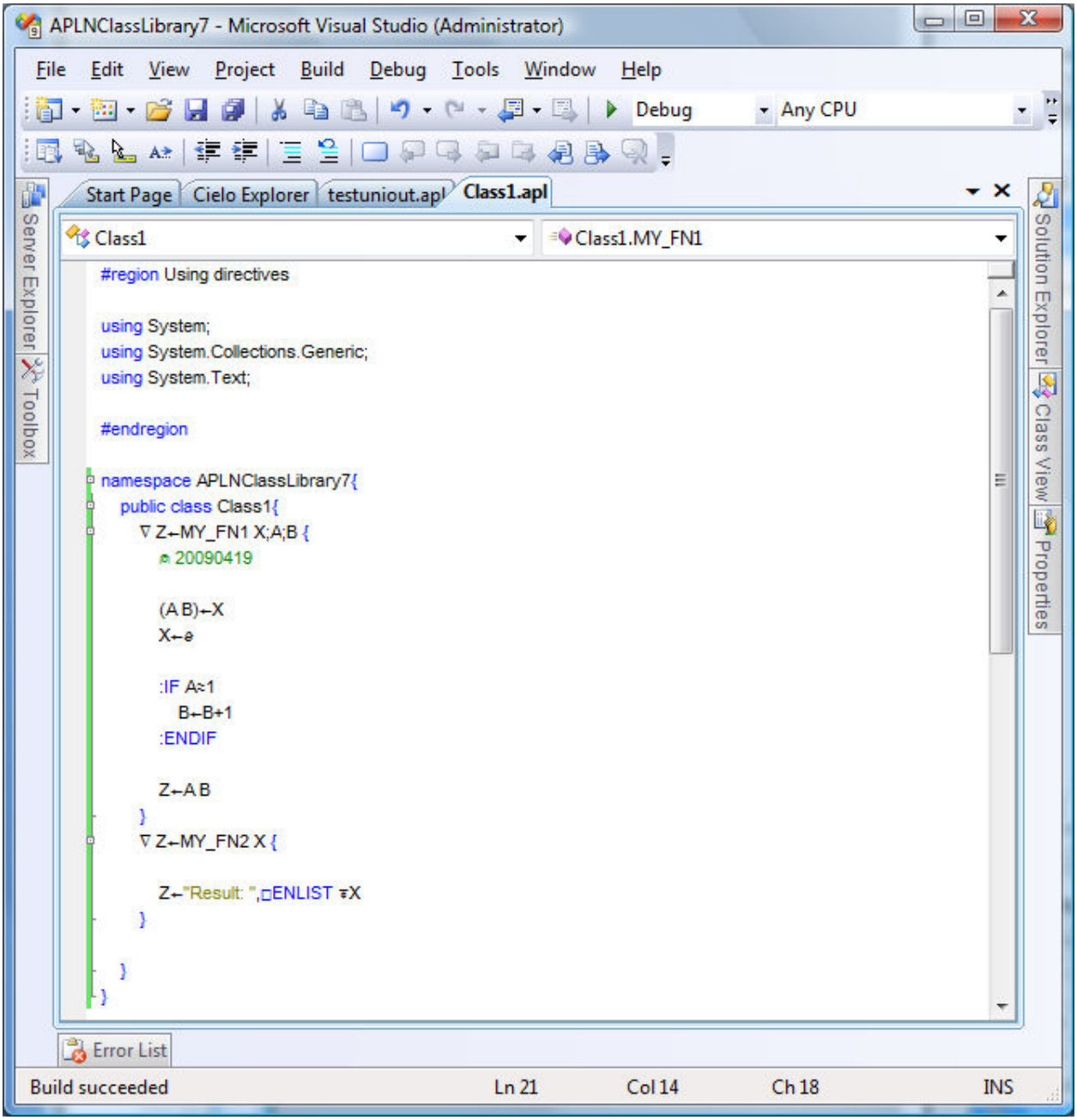
c. In the .apl-format file window containing the converted APL+Win functions, use the Visual Studio “Edit > Select All” and “Edit > Copy” menu items to place the selected Unicode text into the Windows Clipboard:



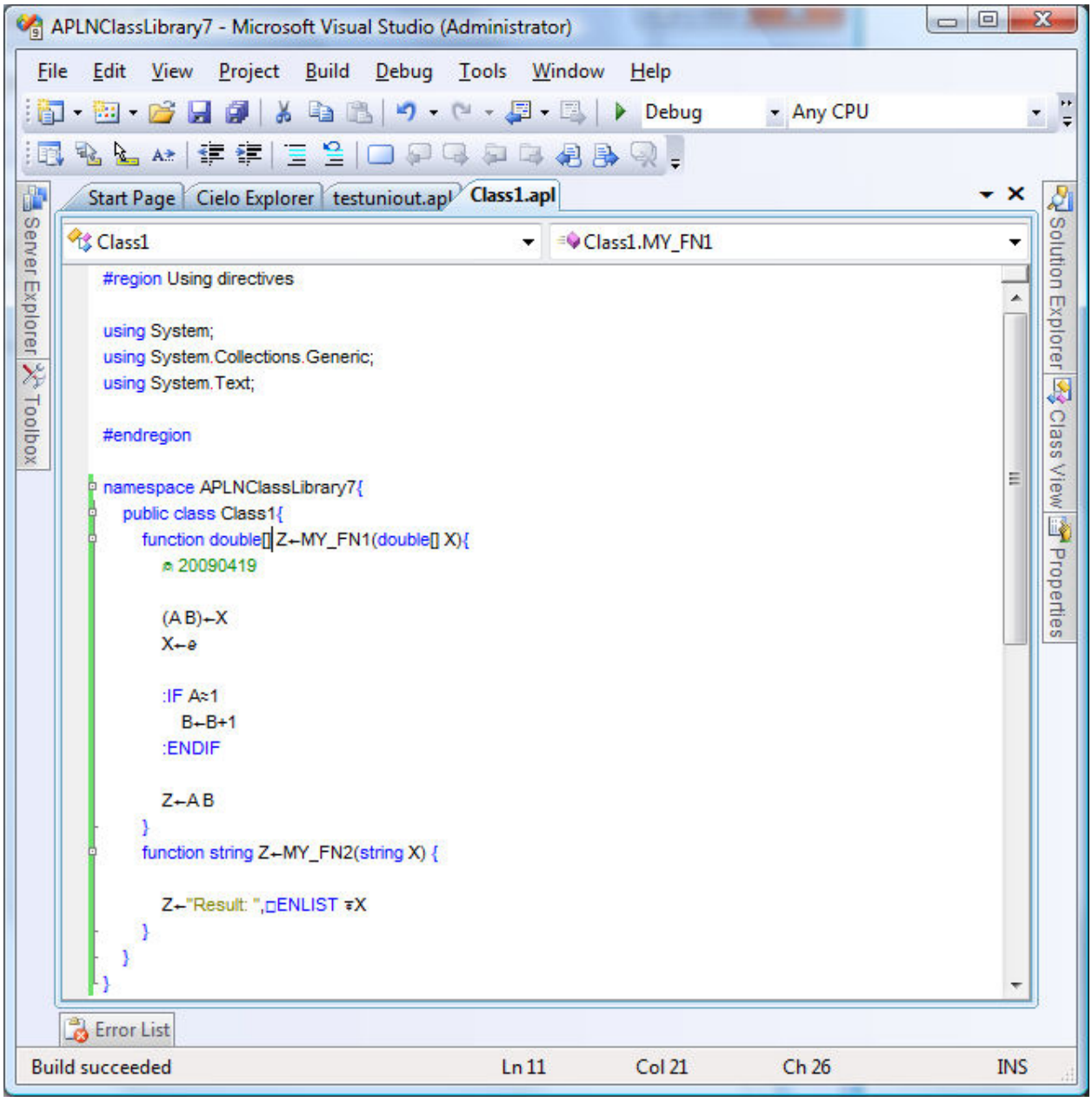
d. With the cursor inside the VisualAPL class definition code block of the VisualAPL class library source file window, use the Visual Studio “Edit > Paste” menu item to place the converted APL+Win functions into the class code block:



e. In the VisualAPL class library window, use the Visual Studio “Build” (“F6” keyboard shortcut) to build the compiled version of the class library which is indicated by the “Build succeeded” message in the Visual Studio status bar. These functions have the VisualAPL ‘operator’ function signatures, so they can be referenced by and used by any VisualAPL project.



f. To make the converted functions inter-operable with any .Net language, including VisualAPL:
    i. Create cover functions which use Visual Studio function signatures and have these cover functions call the converted functions which have traditional APL function signatures, or (as illustrated below)
    ii. Modify the function signatures of the converted functions to use the Visual Studio function signatures:
        - Use the Visual Studio 'function' keyword in the signature instead of the traditional APL "Alt+g" glyph
        - Convert the argument structure to the Visual Studio format enclosed in "(…)". Traditional APL functions with left arguments will have a second, comma-delimited argument in the "(…)" argument list.
        - Provide strong data types for the function arguments and results of the functions
        - Remove any ";local1;local2;…" specifications in the function headers as these variables are automatically local in VisualAPL


APLNClassLibrary7 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Tools Window Help
Debug
Any CPU
Start Page
Cielo Explorer
testuniout.apl
Class1.apl
Class1
Class1.MY_FN1
#region Using directives
using System;
using System.Collections.Generic;
using System.Text;
#endregion
namespace APLNClassLibrary7{
public class Class1{
function double[]Z←MY_FN1(double[] X){
⍝ 20090419
(A B)←X
X←⍵
:IF A≥1
B←B+1
:ENDIF
Z←A B
}
function string Z←MY_FN2(string X) {
Z←"Result: ",⍕ENLIST ⍕X
}
}
}
Server Explorer
Toolbox
Solution Explorer
Class View
Properties
Error List
Build succeeded
Ln 11
Col 21
Ch 26
INS