Use a .Net Standard Library with ⍂CSE

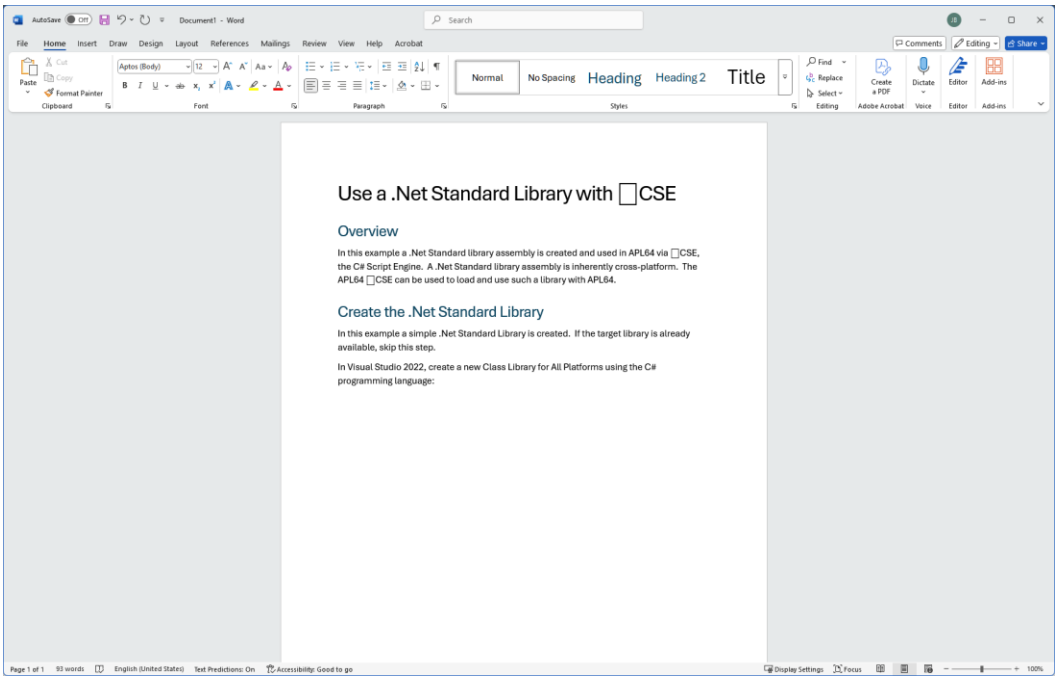
Contents

Overview

In this example, a .Net Standard library assembly is created and used in APL64 via ⍂CSE, the C# Script Engine. A .Net Standard library assembly is inherently cross-platform. The APL64 ⍂CSE can be used to load and use such a library with APL64.

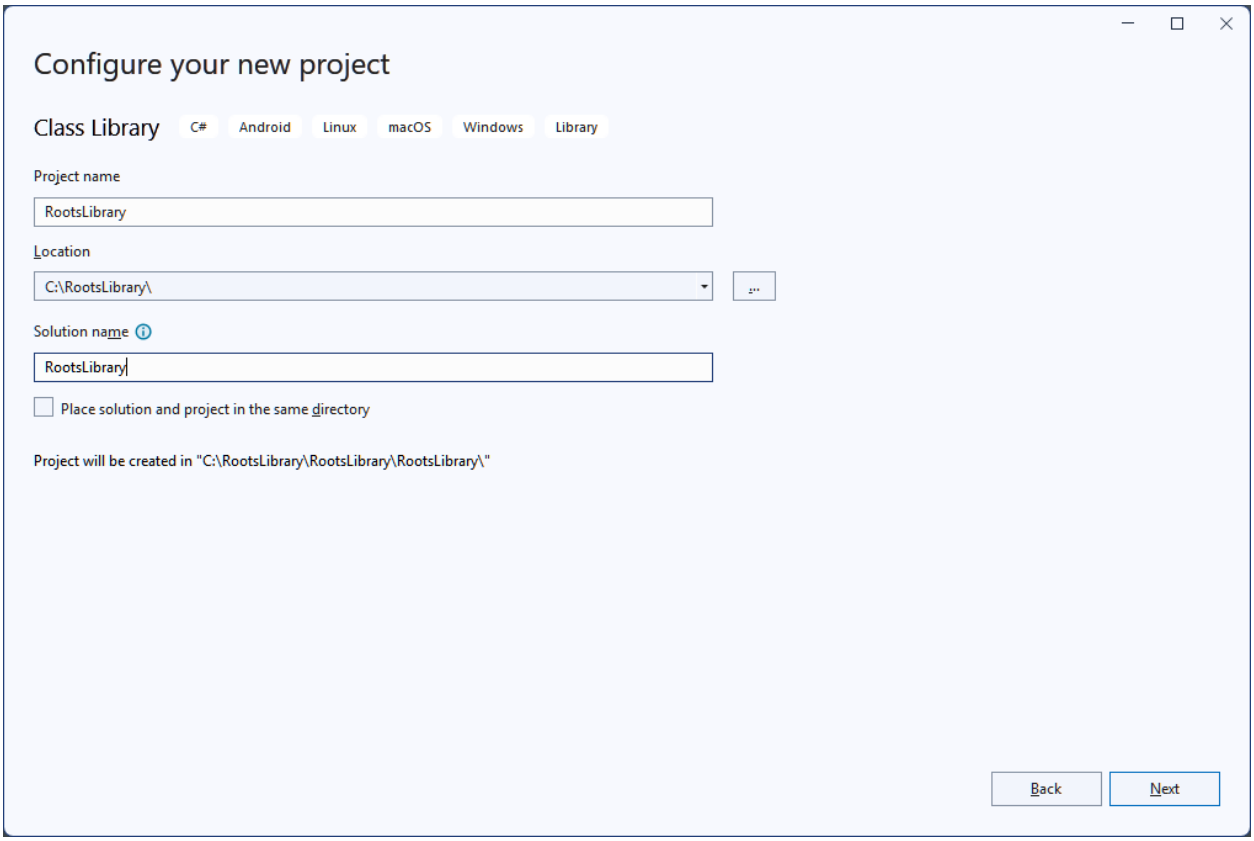
Create the .Net Standard Library

In Visual Studio 2022, create a new Class Library for All Platforms using the C# programming language:

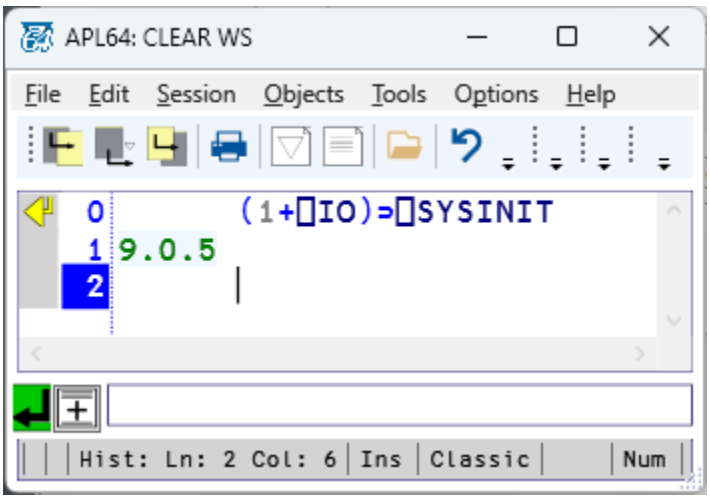


©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 1

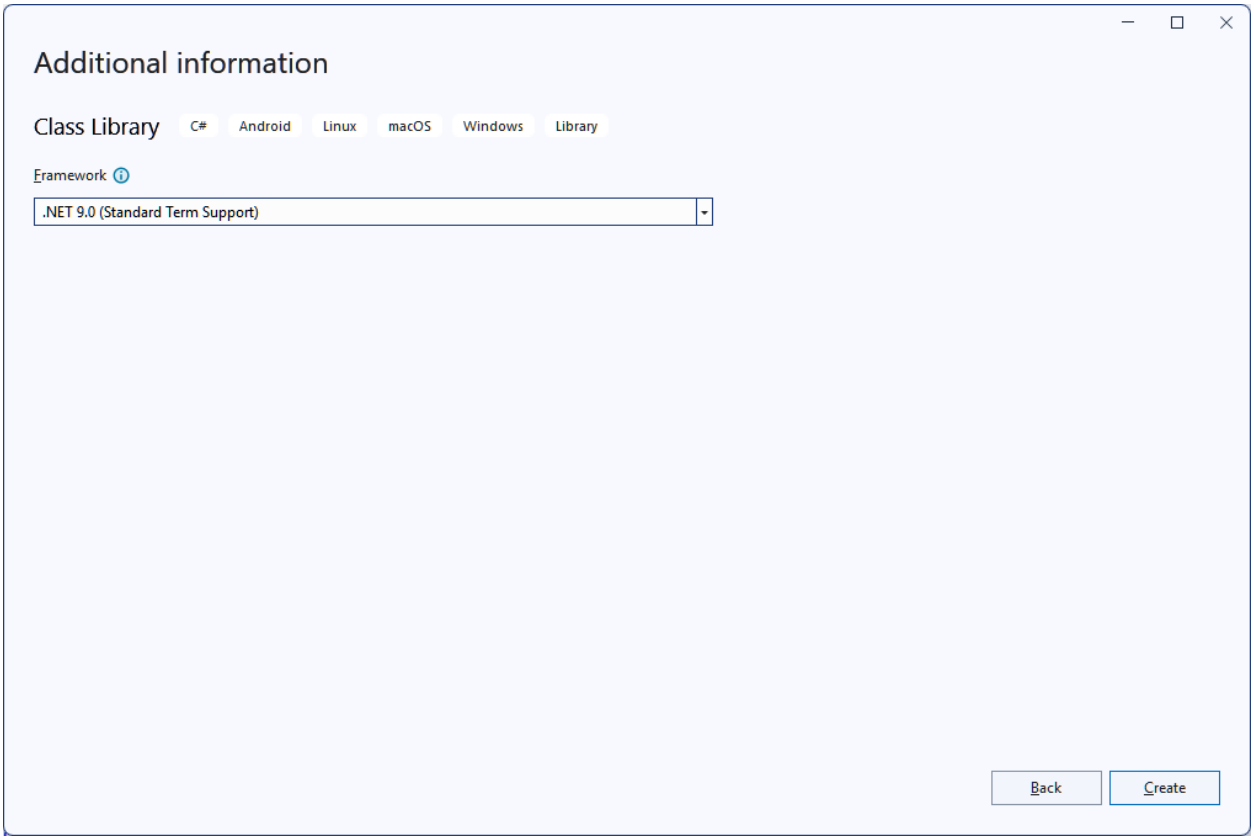
Configure the new project:



Select the .Net Standard version compatible with the current version of APL64. Use (1+⎕IO)⊃⎕SYSINIT to determine this version of .Net.



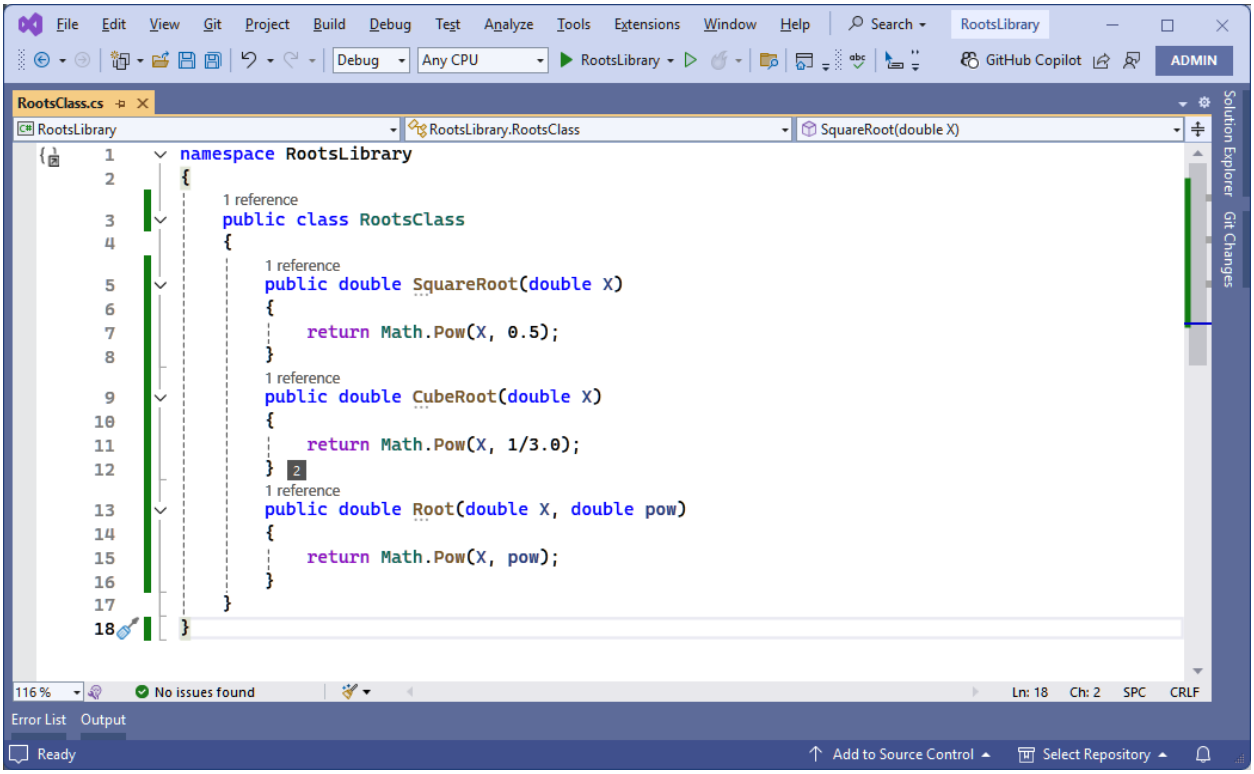
©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 2



Rename the default Class1.cs code file to RootsClass.cs, and replace the existing source code with:

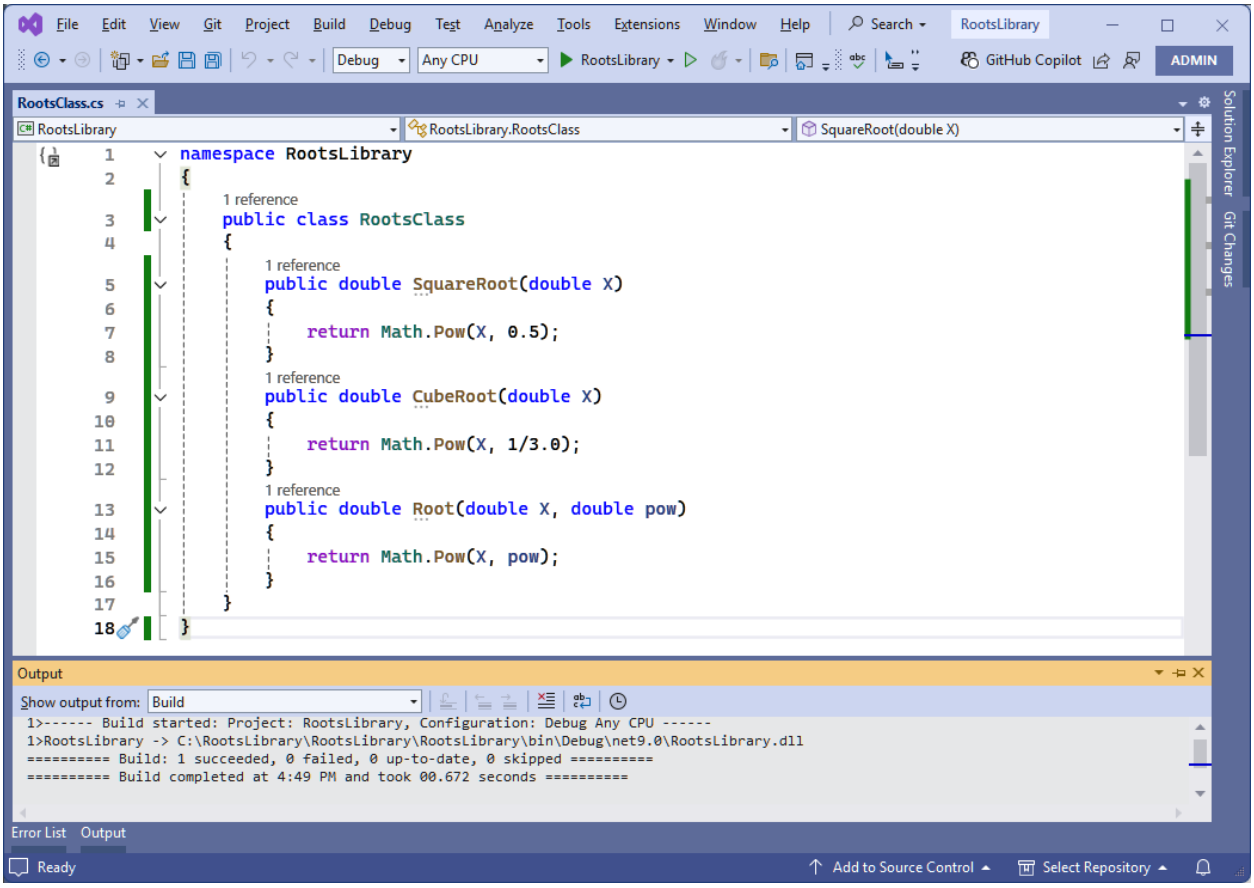
```
namespace RootsLibrary
{
   public class RootsClass
   {
      public double SquareRoot(double X)
      {
         return Math.Sqrt(X);
      }
      public double CubeRoot(double X)
      {
         return Math.Pow(X, 1/3.0);
      }
      public double Root(double X, double pow)
      {
         return Math.Pow(X, pow);
      }
   }
}
```

©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 3



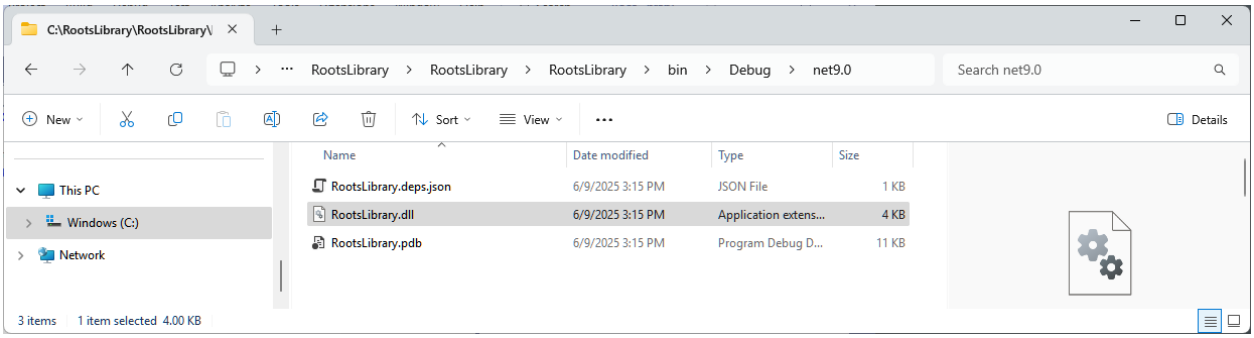
Build the solution and save the project changes to the source folder:

©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 4



The .Net Standard library is in the target folder for the solution build:

C:\RootsLibrary\RootsLibrary\RootsLibrary\bin\Debug\net9.0



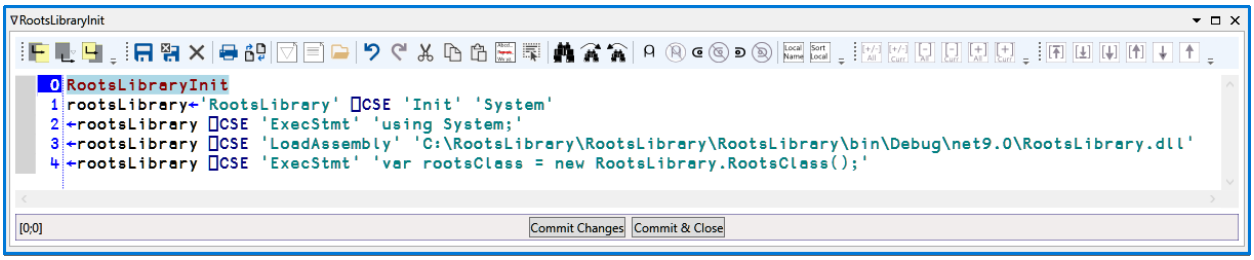
Create the APL Functions using the CSE

Create a developer version instance of APL64 and create the RootsLibraryInit, RootsLibraryGetRoot, and RootsLibraryClose APL64 developer-designed functions.

The RootsLibraryInit will be run once when the APL64 application instance starts:

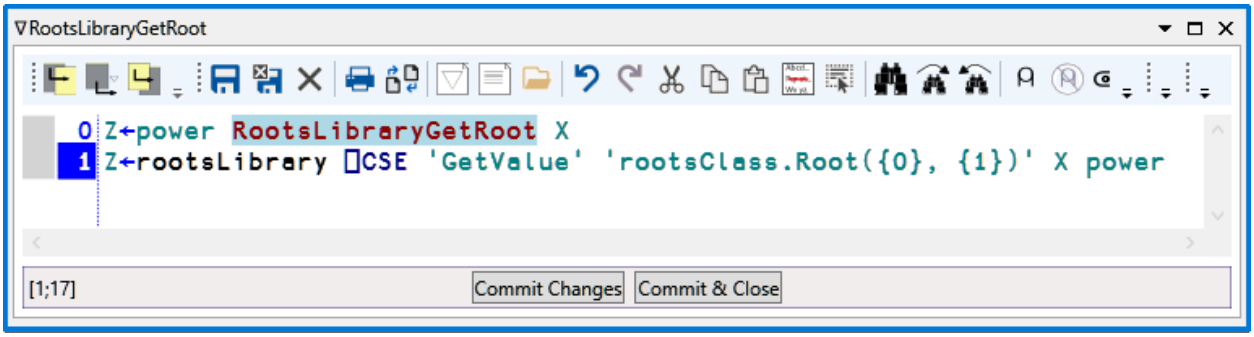
©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 5

```
RootsLibraryInit
rootsLibrary←'RootsLibrary' ⎕CSE 'Init' 'System'
←rootsLibrary ⎕CSE 'ExecStmt' 'using System;'
←rootsLibrary ⎕CSE 'LoadAssembly'
'C:\RootsLibrary\RootsLibrary\RootsLibrary\bin\Debug\net9.0\RootsLibrary.dll'
←rootsLibrary ⎕CSE 'ExecStmt' 'var rootsClass = new RootsLibrary.RootsClass();'
```




```
Z←power RootsLibraryGetRoot X
Z←rootsLibrary ⎕CSE 'GetValue' 'rootsClass.Root({0}, {1})' X power
```

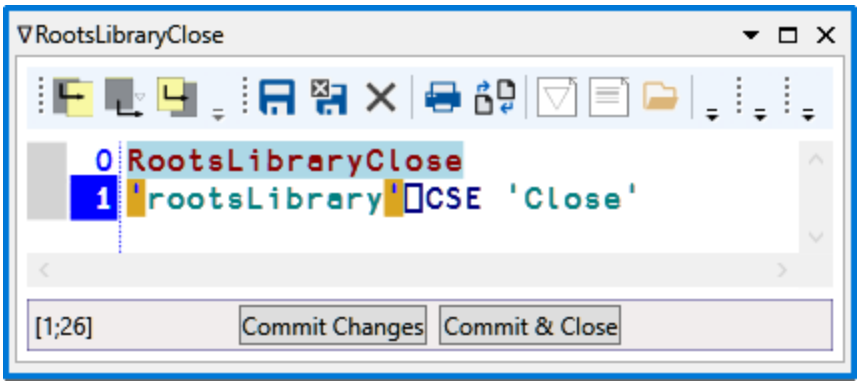
The RootsLIbraryGetRoot function will be used in the APL64 application whenever its functionality is needed. If the .Net Standard library includes additional methods or properties, these can be exposed as APL64 programmer-designed functions.



The RootsLibraryClose function will be run once when the APL64 application instance ends:

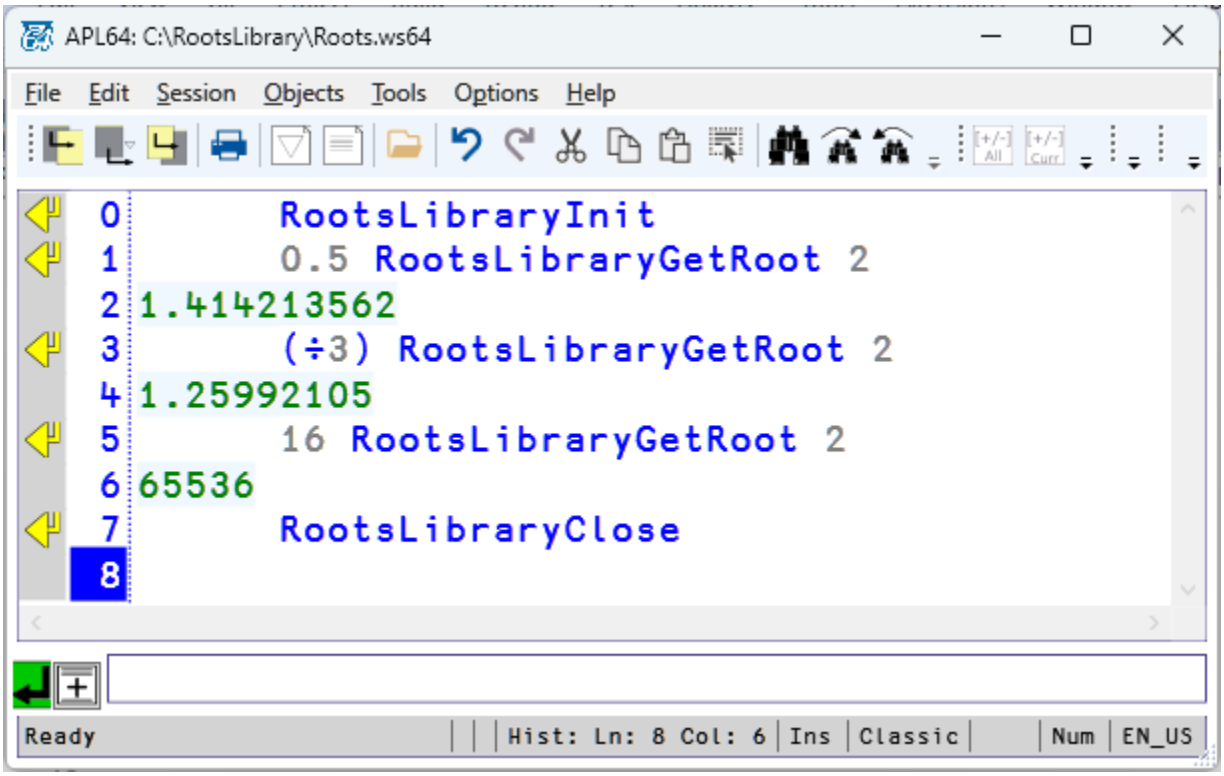
```
RootsLibraryClose
'rootsLibrary'⎕CSE 'Close'
```

©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 6



Use the APL Functions

```
RootsLibraryInit
0.5 RootsLibraryGetRoot 2
(÷3) RootsLibraryGetRoot 2
16 RootsLibraryGetRoot 2
RootsLibraryClose
```



©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 7

Production Environment

In this example the solution was build in Debug mode. In a production version of the APL64 application, the .Net Standard library should be built in Release mode.

In a production version of the APL64 application, the .Net Standard library file would be distributed with the other APL64 application files.

The example the .Net Standard library has scalar argument and result methods, which could be enhanced to support arguments which come from APL64 array variables.

The .Net Standard library methods or the APL64 programmer-developed functions should incorporate exception handling.

©Copyright APLNext LLC – All Rights Reserved - 6/19/2025 - Pg: 8