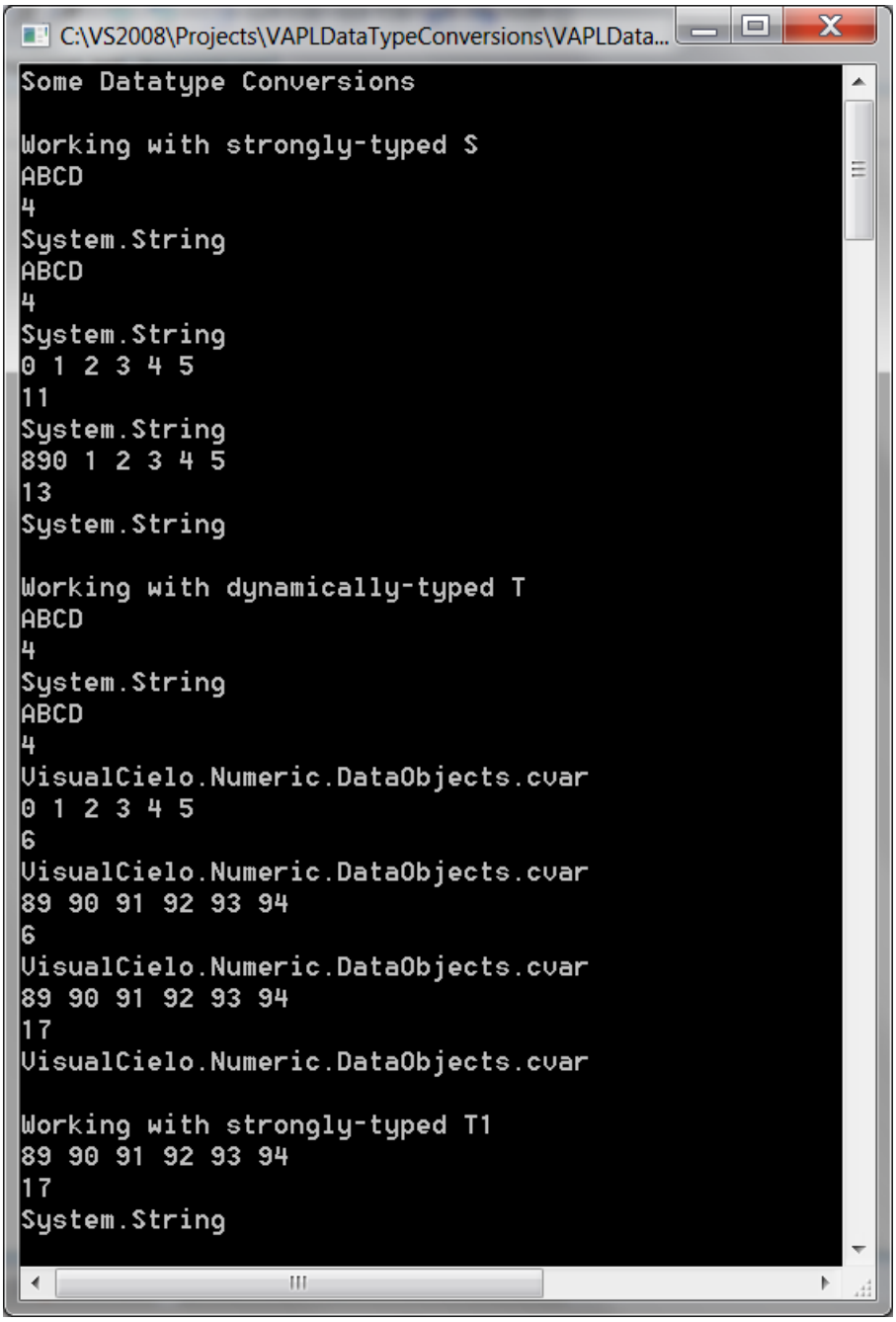
VisualAPL: Working with Strongly- and Dynamically-typed Objects

In this simple VisualAPL VS2008 console project S is strongly-typed (string) and T is dynamically-typed (Cielo):

```
#region Using directives
using System;
using System.Collections.Generic;
using System.Text;
#endregion
namespace VAPLDataTypeConversions{
    public class Program{
        public static void Main(){
            Console.WriteLine("Some Datatype Conversions");
            Console.WriteLine();

            Console.WriteLine("Working with strongly-typed S");
            string S = "ABCD";
            Console.WriteLine(S);
            Console.WriteLine(⍴S);
            Console.WriteLine(S.GetType());
            S←"ABCD";
            Console.WriteLine(S);
            Console.WriteLine(⍴S);
            Console.WriteLine(S.GetType());
            S←⍳6;
            Console.WriteLine(S);
            Console.WriteLine(⍴S);
            Console.WriteLine(S.GetType());
            S←89+S
            Console.WriteLine(S);
            Console.WriteLine(⍴S);
            Console.WriteLine(S.GetType());

            Console.WriteLine();
            Console.WriteLine("Working with dynamically-typed T");
            T = "ABCD";
            Console.WriteLine(T);
            Console.WriteLine(⍴T);
            Console.WriteLine(T.GetType());
            T←"ABCD";
            Console.WriteLine(T);
            Console.WriteLine(⍴T);
            Console.WriteLine(T.GetType());
            T←⍳6;
            Console.WriteLine(T);
            Console.WriteLine(⍴T);
            Console.WriteLine(T.GetType());
            T←89+T
            Console.WriteLine(T);
            Console.WriteLine(⍴T);
            Console.WriteLine(T.GetType());
            T←(string) T.ToString();
            Console.WriteLine(T);
            Console.WriteLine(⍴T);
            Console.WriteLine(T.GetType());

            Console.WriteLine();
            Console.WriteLine("Working with strongly-typed T1");
            string T1← T
            Console.WriteLine(T1);
            Console.WriteLine(⍴T1);
            Console.WriteLine(T1.GetType());



            Console.WriteLine();
            Console.WriteLine("Click the Enter key to end this application");
            Console.ReadLine();
        }
    }
}
```

When this project is run, the result is:



Notice how the initial strong typing of S as string is preserved throughout subsequent re-assignments. Especially interesting is how VisualAPL preserves the string datatype in the re-assignments of S by converting the numeric vector ‘0 1 2 3 4 5’ to a formatted string and subsequently where the ‘+’ glyph is treated as the .Net string catenation operator.

Notice how the first (dynamic) assignment of T, line [34], results in a .Net string datatype and that that subsequent assignments of T continue to result in dynamically-typed Cielo datatypes, even until the final assignment of T which explicitly assigned to T the value of T cast as a string, yet T remains a dynamically-typed Cielo datatype.

Finally, the value of T can be assigned to the strongly-typed (string) T1.