CIS 365 Lab03: Recursion, again

Adam Lewis

<2015-05-18 Mon>

Contents

1 Objectives

1. Continue our look at classes and methods
2. Review recursion
3. Continue to gain experience with VS and C#.

2 Background

Recursion is the process of repeating items in a self-similar way. In computer science, recursion refers to a method of defining functions in which the function being defined is applied within its own definition. From a function standpoint, we must define at least two cases when we use recursion: (1) a base termination case, and (2) a recursion step based upon values we have already computed.

Consider how one computes an exponent. If we want to compute the value of an exponent: $a^n = a * a^{n-1} = a * a * a^{n-2} = \ldots$. This means that we can write a recursive algorithm for this function:

```
Power(baseNumber, exponent):
   IF exponent equals 1 THEN return the baseNumber
   ELSE
      return baseNumber * Power(baseNumber, exponent -1)
   END
```

3 Instructions

So, we can create an app in C# that reads a number from the console, convert it to an integer, and then calls a method that implements our algorithm.

3.1 Implementing the solution

Now let's implement our solution using C#.

First thing that we need to do is to start Visual Studio.

Once you have Visual Studio started:

1. Select "New Project" or select *File>New>Project*
2. Select "Visual C#" in the templates menu
3. Select "Console Application" from the list of Project types
4. Set the project name to "ReverseApp"
5. Adjust the project location to where you want to store this project on your machine
6. Select "OK"

The IDE will grind for a few moments and then present you with a newly created project. Visual Studio works with "Solutions" and "Projects" and you can see the contents of the Solution in the "Solution Explorer" windows. By default, VS creates a C# *namespace* with a name that matches your solution name and a class named "Program" with a **main()** function. I like to rename this to something more appropriate for the project:

1. Select the file "Program.cs" in the Solution Explorer.

2. Right-click and select "Rename" from the context menu. Rename the file to something more appropriate such as "RecursiveExponentClass.cs". Note that VS will also change the name of the class in the source file.

3.2 So, let's do things in a top down fashion

That means we should start with our **main()** method. Add the following to your new class in Visual Studio:

```
public static void Main( string[] args )
{
   int baseNumber; // the base to raise to a power
   int exponent; // the power to raise to

   // prompt user for base and obtain value from user
   Console.Write( "Enter base: " );
   baseNumber = Convert.ToInt32( Console.ReadLine() );

   // prompt user for exponent and obtain value from user
   Console.Write( "Enter exponent: " );
   exponent = Convert.ToInt32( Console.ReadLine() );

   if ( exponent > 0 )
   {
      int result = Power( baseNumber, exponent );
      Console.WriteLine( "Value is {0}", result );
   } // end if
   else
      Console.WriteLine( "Invalid Exponent." );
} // end Main
```

Note the following:

1. The function **Power()** is where we'll implement our algorithm. Given how it is being called, we will treat it as a class method.

2. Note that we have to read the number using the **Console.ReadLine()** method and convert what has been entered into an Integer using **Convert.ToInt32()**

3.3 Now to do the heaving lifting

So, we need to pass the inputted integer to our function and then implement our algorithm as the body of the function. We'll return back the resulting integer:

```
public static int Power( int baseNumber, int exponent )
{
   if ( exponent == 1 )
      return baseNumber;
   else
      return baseNumber * Power( baseNumber, exponent - 1 );
} // end method Power
```

3.4 Build and test your application

Build your application in Visual Studio. Run your program and enter a few test cases. Set a breakpoint in **Power()** and watch what happens to the values in the local variables as your program runs.

4 Submission instructions

Combine a copy of your source code and a screen-shot of your program into a PDF file (you can use Microsoft Word to do this) and attach it to the assignment on Blackboard. Emacs 24.4.1 (Org mode 8.2.10)