PREPARING FOR PROGRAMMING ASSIGNMENT #2

Re-Read Tic-Tac-Toe

Programming assignment #2 is the game "Connect 4" where you drop checkers into a vertical grid until someone gets 4 in a row. This is generally similar to the Tic-Tac-Toe example from earlier in this module. Spend some time reviewing how that program was created before starting on the assignment. You'll find there are lots of similarities. **[0407] Top-Down Design Example (Tic-Tac-Toe) (https://lwtech.instructure.com/courses/1841516/pages/0407-top-down-design-example-tic-tac-toe)**

Generating Random Numbers

C#'s "System.Random()" can be used to generate random integers for game programs like this. (It is not good enough for cryptography applications, but we don't need that level of randomness for our game.) The important things to remember are:

- Only declare one instance of Random() in your program. Declaring a new instance each time you need a random number will often lead (paradoxically) to non-randomness.
- At the top of your program, add "**static Random rng = new Random();**" (You can name it anything you want. I use "rng" which stands for "Random Number Generator."
- You can then use "**rng.Next(x);**" anywhere in your application to get a random integer between 0 (inclusive) and x (not inclusive).

While vs Do/While

If your code is going to go through the loop at least once, try using a Do/While loop and see if it makes things simpler. It usually will, but not always.

Create Methods Intelligently

Remember the key reasons for creating a method instead of just including code in-line:

- Adds self-documentation by giving a name to a section of code
- Creates a new scope for local variables
- Fosters code reused in the case of general-purpose methods
- Reduces duplicate code
- Potentially provides an API for other programs to use
- Allows parts of the code creation task to be given to other programmers
- Helps separate the User Interface code from the program logic

For this exercise, we only care about SOME of these things, not all of them. Only create a new method when it significantly improves the readability, maintainability and future flexibility of your program.

Displaying 2-D Arrays on the Console

It is important to remember how rows and columns in a 2-D array map onto characters on the console screen. When displaying the contents of a 2-D array, you need 2 for loops (why?) however the outer for loop needs to be used for the second dimension (the vertical columns, aka "y" (or "j")). The inner for loop is for the first dimension (the horizontal rows (x/i). Here is an example:

```
private static void DisplayArray(int[,] grid)
{
   for (int y = 0; y < 10; y++)
   {
      for (int x = 0; x < 20; x++)
      {
         Console.Write(grid[x, y] + " ");
      }
      Console.WriteLine();
   }
   Console.WriteLine();
}
```

About Pseudocode

According to WIkipedia:

> **Pseudocode** is an informal **high-level** (https://en.wikipedia.org/wiki/High-level_programming_language) description of the operating principle of a **computer program** (https://en.wikipedia.org/wiki/Computer_program) or other **algorithm** (https://en.wikipedia.org/wiki/Algorithm).
>
> It uses the structural conventions of a normal **programming language** (https://en.wikipedia.org/wiki/Programming_language), but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as **variable declarations** (https://en.wikipedia.org/wiki/Variable_declaration), system-specific code and some **subroutines** (https://en.wikipedia.org/wiki/Subroutine). The programming language is augmented with **natural language** (https://en.wikipedia.org/wiki/Natural_language) description details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and **scientific publications** (https://en.wikipedia.org/wiki/Scientific_publications) that are documenting

various algorithms, and also in the planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program. Pseudocode resembles, but should not be confused with, **skeleton programs (https://en.wikipedia.org/wiki/Skeleton_(computer_programming))** which can be **compiled (https://en.wikipedia.org/wiki/Compiler)** without errors.

(Yes, professors can quote Wikipedia. It's totally allowed.)

In my way of thinking about it, pseudocode is a description of how a program works that can be understood by both programmers and non-programmers. It is a way to ensure that the code that a programmer ultimately writes does what the customer expects.

It is important to remember that pseudocode should not have any *implementation-specific details* in it. You should try to keep your pseudocode as high-level as possible for as long as possible. So, for instance, the fact that you are storing data in an array should not be mentioned (directly) in pseudocode. The fact that you are printing your output on a Console is also an implementation-specific detail that you don't want to focus on until you are actually writing the code.

By delaying the introduction of implementation-specific details until later in the process, you can be sure that your design is as flexible and maintainable as possible. It also helps ensure that you are focusing on the STRUCTURE of your program first rather than the lower-level details.

Remember, as programmers, you are now starting to work on bigger programs that you may not be able to fit inside of your head all at once. Top-down design (using pseudocode) allows you to focus on your program in ways that let you keep things understandable. You abstract out the low-level details and focus on high-level structure knowing that, later, you can come back and create the low-level stuff module-by-module without having to worry about the program as a whole.

There are usually 4 or 5 "phases" of pseudocode that you should create before diving into actual coding. The first phase is simply a statement of what the program does. The second phase focuses on the high-level flow-of-control for the program (i.e., what your Main() method will probably look like). It includes things like where does the program start, how does it exit, what are the big loops that it has and what are the high-level tasks it needs to accomplish. Phase Three takes the high-level tasks that were identified in phase two and breaks them down into smaller tasks. etc. etc. etc.

Keep breaking things down UNTIL you are sure you have identified all the important tasks your program needs to perform. Remember, Top-Down Design is all about focusing on the processes that your program will perform (i.e., its methods). Once you are sure you have all the important processes identified (and you have your pseudocode validated by the business/design people you work with), THEN you can start to convert it into actual code.

Note: Top-Down Design (and pseudocode) generally ignores dealing with how data will be stored and manipulated by your program. While you probably will have a sense of what data types and data structures your program will use, those are implementation details that should be left out of the design in

order to maximize implementation flexibility. This also means that one of the first things you should do when you start to convert your pseudocode *to real* code *is to think hard about how the data will be stored and used by your implementation.*

Top-Down Design with pseudocode takes practice to learn! If this concept is still confusing, go back and re-read the Tic-Tac-Toe example, then - without looking at what I wrote - try to create your own pseudocode to that simpler program. Compare what you came up with to what I wrote. Let me know if you are still confused.

IMPORTANT! Creating and Turning In Your Programming Assignment

Remember that you need to zip up the entire directory that Visual Studio created for your assignment. **It is also important that you don't rename or move any files inside of your project directory!** You really want to enter the correct name for your project at the time you create it. Changing it later can easily break things.

Here are the steps for creating a project and then zipping it up so it can be submitted in Canvas:

FOR WINDOWS:

1. Start Visual Studio. In the "New project" area of the Start Page, click "Console App"
2. On the "New Project" dialog box, click "Get Started", then click "Console App".
   In the "Name" box, enter "LWTech.ChipAnderson.Assignment2" (but replace my name with yours - no spaces!)
   In the "Location" box, leave things alone. (It is usually pre-set to something like "C:\Users\Chip Anderson\source\repos")
   In the "Solution Name" box, leave things alone. (It should be the same as what you typed in the "Name" box)
3. Make sure the "Create directory for solution" checkbox is checked, then click "OK" (Do not check the "Add to source control" checkbox)
4. Your new project will be created in the location you specified and a simple Program.cs file should appear on your screen. (Note that the namespace is automatically set correctly!)
5. Delete the additional "using" statements that Visual Studio added. For now, just keep "using System;"
6. You can now modify Program.cs as needed in order to implement the assignment. If you need to add a new file to the project, go into the "Solution Explorer" panel, right click on the "LWTech.ChipAnderson.Assignment2" node and select "Add" and then "New Item..." from the popup menu.

Once your program is complete and working, follow these steps to ZIP it up and submit it:

1. Click "File/Close Solution" from Visual Studio's menu bar then **SHUT DOWN VISUAL STUDIO!**
2. Open the "Windows File Explorer" and navigate through the folder in your computer until you find the "repos" directory. Open up that directory in File Explorer.

3. Inside that directory should be a folder with the name "LWTech.ChipAnderson.Assignment2" (If not, you're in the wrong place. Go find out what "Location" was set to when you created your project.)
4. Right click on "LWTech.ChipAnderson.Assignment2" and then click "Send To..." Finally, click "Compressed (zipped) folder"
5. The file "LWTech.ChipAnderson.Assignment2.zip" should appear in that directory. The size of the file should be near or over 150KB.
6. Upload that file into Canvas as your submission for Assignment #2.

Congrats!!! You're done!

FOR MAC:

1. Start Visual Studio. Click the "New Project..." button on the "Get Started" page.
2. Click "Console Application" then click "Next"
3. Leave the "Target Framework" set to ".NET Core 2.1" and click "Next"
4. In the "Project Name" box, enter "LWTech.ChipAnderson.Assignment2"
   Leave the "Solution Name" box alone - it should also have "LWTech.ChipAnderson.Assignment2" in it
   Leave the "Location" box alone too - it should be set to something like "/Users/chipa/Projects"
5. Make sure the "Create a project directory" checkbox is checked, then click the "Create" button
6. Your new project will be created in the location you specified and a simple Program.cs file should appear on your screen. (Note that the namespace is automatically set correctly!)
7. You can now modify Program.cs as needed in order to implement the assignment. If you need to add a new file to the project, go into the "Solution Explorer" panel, right click on the "LWTech.ChipAnderson.Assignment2" node and select "Add" and then "New File..." from the popup menu.

Once your program is complete and working, follow these steps to ZIP it up and submit it:

1. Click "File/Close Solution" from Visual Studio's menu bar then **SHUT DOWN VISUAL STUDIO!**
2. Open a Finder window on your Mac's desktop and navigate to the directory you had Visual Studio create your project in. You should see a folder labeled "LWTech.ChipAnderson.Assignment2" in that directory. (If not, you're in the wrong place. Go find out what "Location" was set to when you created your project.)
3. Right-click on that folder and select Compress 'LWTech.ChipAnderson.Assignment2" from the menu that appears.
4. The file "LWTech.ChipAnderson.Assignment2" should appear in your directory. It should be around 330KB in size.
5. Upload that file into Canvas as your submission for Assignment #2.

Congrats! You are done!