**Creating and Using Header Files**

**Why Use Header Files**

In C++, it is common to have a class in two parts. The first is the Header file, which declares the class and all the items in it. The second part is the Source file, which states what all the functions in that class do. This is useful as we can create all our declarations in one place, and then include the Header file in as many Source files as necessary. When you do something like:



You are including the header file of String class. Let’s try using Visual Studio to create a class for us and split it into these two files. Right-click the ‘Source Files’ folder in your Solution Explorer and select:

**Add** -> **Class…**

This will bring up a Window for adding a class to the project. Select:

**Visual** **C++** -> **C++** -> **C++** **Class**

Then click **Add** at the bottom right of the window. This will bring up another Window where you can enter the name of the class. Give it any name you want; it will auto-fill the **.h file** and **.cpp file** fields. Select **Finish**.

This will add two files to your project, one in the **Source Files** and one in the **Header Files** part of the Solution Explorer. The Header file will contain the declaration of the class you just created, along with an automatically generated Constructor and Destructor (we will get to the Destructor soon). The Source file will contain the definitions of these functions. If you want to add a new function to the class, first we add it in the classes Header file like so:



Then we add the following to the Source File:



The code above is telling the compiler what the function called ***another\_function*** inside CyberPet does. We can now fill this with whatever code we want.

Now we need to add the proper include to our other code files and we can access the class we have just created. In the file where you main is defined, add at the top an include to the Header file you created. Like:



Notice how we are using quotation marks “” rather than <> around the header file. This is due to how the pre-processor searches for the include files. Items that are part of the Standard Template Library have a pre-determined location set by Visual Studio, and are accessed with <>. Items included with “” are looked for in the same directory as the file attempting the include.

**Exercises**

1. If you haven’t already, create a CyberPet class. Then add as a private variable, a string to describe the pets name, and a number of describe how hungry it is.
2. Create some public functions to you can set and get the name and hunger variables.
3. Create an instance of the CyberPet class in the main.
4. Have the user input a value for the name and hunger of each CyberPet.

**Summary**

* Declare classes in their own Header file, and their code in a separate Source file.
* When creating classes using the Class Wizard, it will automatically generate the files with a Constructor and Destructor for the class.
* When including Header files, remember that one’s part of the Standard Template Library are included with a <name>; whereas ones you have locally in your project use “name”.
* You don’t **HAVE** to use the Header/Source file format for classes (You can create them wherever you want), but it is very useful for keeping your project clean.