Programming Fundamentals
CS 1336 – C++

Formatting Instructions on Assignments

Comments and Formatting

We are going to enforce some comment and general formatting requirements throughout the semester. These are designed to help your code be as neat and professional looking as possible.

Program Headers

Please begin each program with a comment header that contains the following information: (a) the author's name, (b) the class name and section, (c) the date (of submission, last edit, etc), (d) the assignment being submitted, (e) the compiler used, and (f) a description of what the program does.

Here's the example given in class:

```
// Author:      Charles O. Shields, Jr.
// Course:      CS1336.XXX
// Date:        2/4/2014
// Assignment:  Lecture Homework #1 Exercise #2
// Compiler:    Visual C++ 2010

// Description:
// This program demonstrates how a string value
// can be printed to the console.

int main ()
{
   cout << "Hello there." << endl;

   return 0;
} // end function main()
```

Please put the words "Author:", "Course:", "Date:", "Assignment:", and "Compiler:" in the header (so the information you provide will be identified and make sense), and make sure everything lines up well and is neat. Thus, you will line up the first letter of the name ("C" in this case) with the course "CS1336", and that with the date, etc., exactly as it is indicated here. This is a good but simple example of an acceptable program header for this class.

Alignment requirements

In this class, we'll use the same alignment pattern for the braces that the book authors use. That is, left braces will appear beneath the first character of the given class, function (or later, "for"

loop, etc). (Thus, the left brace for the `main()` function is under the small "`i`" in "int.") Right braces will line up vertically with the left braces. This style is indicated in the above example.

To be explicit about it, we do not want to use the common method of placing braces in which the left brace appears on the same line as the class or function header, and the right brace is lined up with the first character. It is true that this is an extremely common style. But for this semester, I'd like us all to standardize on one good style, i.e., the one the book uses, for the sake of clarity and professionalism. If you want to do something different when you leave this class, then of course, you are perfectly welcome to do so.

Indentation Requirements

Each new level in your code should be indented by 3 spaces. Thus, in the example, the "`cout << "Hello there." << endl;`" statement is indented 3 spaces from the braces of the function `main()`. If, in the future, there are deeper structures in the code (e.g., a "`for`" loop or something comparable), those will be indented 3 spaces as well from whatever is in their outer scope. (We will discuss those issues as they arise.)

Note that instructions for configuring Visual C++ 2010 to the specifications we will be using this semester are found in the file "Creating a Project in VC++". This file is in the "Supplemental Files" folder on eLearning, and the configuration instructions are the very last item in the file. These instructions describe how to: (a) turn on line numbers in the editing window, (b) set the tab size to 3, (b) set the indentation size to 3, (d) insert spaces instead of tabs. Please make sure that your compiler has been configured in this way.

Commenting ending braces

As we've said in class, all right ("ending") braces terminate some structure, either a function, a "`for`" loop, an "`if`" statement, or some other structure. Please put a small single line comment on the ending braces on major structures like classes and functions. (Later on, we'll do the same thing for significant loops and decision structures.) The comment should indicate what structure has been terminated. As illustrated above, it can be something simple, like "`end function main()`". The object is to make it clear to what structure the right brace is attached.

Variable declarations in `main()`

Please declare any variables used in the `main()` function according to the requirements listed below for functions in general.

Naming Conventions on Homework Programs

Most of our assignments this semester will consist of more than one program. To help us easily identify a given program, let's use the following naming convention this semester for our source code files: "`HWX_ExY.cpp`", where "X" is the homework number and "Y" is the program within that homework. "HW" stands for "HomeWork", and "Ex" stands for "Exercise".

For example, let's say you are submitting the second program in homework number 3. Then the submitted source code file should be: "`HW3_Ex2.cpp`".

Formatting and Comment Requirements in Functions

Variable Declarations and Comments

Please declare all variables used in a function at the top of the function, with a comment describing the purpose of each variable. This necessarily means that only one variable will be declared per line.

An example from the top of a function in a recent C++ program:

```
string binaryToHexadecimal(string inputNumber)
{
   const int SPLIT_SIZE = 4;   // number of digits for the binary split
   string workingStr = "";     // final output Hex String
   string subStr   = "";       // holds the 4 bit binary substring
   int firstPos = 0;           // marks first index in binary substring
   int secondPos = 0;          // marks second index in binary substring
   int binaryResult = 0;       // result of 4 bit binary-decimal calculation
   bool keepGoing = false;     // controls the operation of the while loop
}
```

Function Headers

Please comment all functions with headers that give the following information: (a) the function name, (b) a description of the function and what it does, (c) the input variables and what they do, (d) the return type, if the function returns a value, and what the return value does. An easy way to do that is to use the C++ "multi-line" comment format ("/* --- */") described in class and illustrated below.

An example of a complete function header is given for the function referenced above...

```
/*****************************************************************************
* Function: binaryToHexadecimal (string inputNumber)
* Descr:    This function takes a binary string as input, then calculates
*           and returns its hexadecimal equivalent.  It works by splitting
*           the input string into 4-bit substrings (starting from the
*           right), obtaining the decimal value for that substring from
*           "binaryToDecimal()", and then prepending the appropriate
*           Hex character to the output String "workingStr".
* Input:    inputNumber
*           The binary string input.
* Return:   A string corresponding to the Hex value of the input string.
*****************************************************************************/
string binaryToHexadecimal(string inputNumber)
```

Grading Guidelines

The programs will be evaluated according to the following guidelines:

Component	Percentage of final grade
Comments and Indentation	20%
Compilation	30%
Output and other issues	50%

Output is the most important part, so it is the largest percentage. But you will get some points for a program if it compiles, even if it doesn't run properly. Note that a C++ program will not produce any output if it has compile time errors.

As the semester goes on, we may change these percentages. If we do, I'll certainly inform you.

I hope this helps clarify the stylistic requirements for these assignments. Please let me know if you have any questions at all. Best….