Programming Style Guidelines

There are many methods of styling C++ programs. Unfortunately, many of these are based on personal whim or a misguided desire to make programs as short as possible. The programs given out in this class will follow the style guidelines given below. You should also follow these style guidelines when writing programs for this course.

1) Identifiers (or Variable Names) representing constants should appear entirely in all upper case. This convention was started by C programmers and has been widely accepted.

   Example:
   ```
   const float PI = 3.1416;
   const int TABLE_SIZE = 10;
   ```

   Note how the underscore character is used to separate the individual words that make up a constant identifier.

2) Programmer defined identifiers for variables and subprograms (or functions) should begin with an uppercase letter. This makes it easy to distinguish them from standard identifiers or reserved ones, since these are always lowercase. Since there are no standard identifiers of length one, single character variable names such as "x" can be either upper or lower case.

3) If the identifier consists of several words "glued together", each word should begin with an upper case letter. For example, `HourlyRateOfPay` is to be used rather than `Hourlyrateofpay`. The identifier `Hourly_rate_of_pay` is also acceptable, but is harder to type in.

   See handout *Choosing Good Identifier Names* for detailed suggestions on choosing good names.

4) Source lines should not normally contain more then one statement.

5) Whitespace should be used to make programs more readable. For example, related statements can be grouped together and surrounded with a blank line above and below.

   Use a blank before and after operators. Example: `cout << N;` not `cout<<N;`

   Use a blank line below variable declarations.

   Use some whitespace to separate comments from code – don't "glue" comments to code, making the actual code statements harder to locate.

6) The purpose of indentation is to make the logical structure of the program easy to grasp. Indent only to show that the indented material is "subsidiary to" or "controlled by" a preceding line.

   Use following as example:

   ```
   if ( Length > MaxLength )     // My preferred indentation style
   {                             // makes it easy to match up braces
      MaxLength = Length;
      cout  >>  "Found longer string!";
   }
   ```

7) Comments above the main function should contain your name, date, program name (as saved on disk), description of what the program does, and any constraints or limits. Function comments will be discussed later.

8) You should add trailing comments to the right of hard to follow code. Never comment the obvious!