Lab 4 - CMPS 1043- Computer Science I
Formatting Output using <iomanip>

Lesson objective: Demonstrate proper use of basic formatting manipulators
1) setw 2) setprecision 3) fixed 4) showpoint 5) left & right

Formatting refers to the manner in which data is displayed on the screen or in a printout. This includes such elements as spacing, number of decimal places, alignment, etc. The use of formatting manipulators allows the programmer to control the output to display as desired. Formatting manipulators MUST be placed in output commands (cout or file output statements).

#include <iomanip> file required to use the output manipulators listed above

Type in the following basic program then execute to ensure correctness:

```
#include <iomanip>
#include <iostream>
using namespace std;

int main ( )
{     double A = 132.364, B = 26.91, C = 307;
      cout << A/B;
         // will insert additional output statements here
      system("pause");   //causes black display screen to stay visible
      return 0;
}
```

SETWIDTH: **setw(#)** specifies the width of the field in which the next value will be displayed; this is the only one of the above manipulators that must be repeated for each item being printed.

- If the value requires fewer than the number of specified spaces, the value will be right justified with blank spaces padded to its left
- If the value required more than the number of specified spaces, the setw manipulator is "ignored" and the number of spaces needed is used.

Note: Each digit/character, including the decimal point, requires one space.

To the program above, insert the following statements and execute:

```
cout << setw(3) << B << '\n';
cout << setw(4) << B << '\n';
cout << setw(5) << B << '\n';
cout << setw(6) << B << '\n';  // first occurrence of a padded blank
cout << setw(7) << B << '\n';
cout << setw(8) << B << '\n';
```

When used successively on the same output statement, the next field begins immediately following the last digit of the previous value. Modify your code as shown below. Note how the data on successive lines are aligned in columns that are right justified.

```
cout << setw(8) << B << setw(9) << B << setw(10) << B << '\n';
cout << setw(8) << A << setw(9) << A << setw(10) << A << '\n';
cout << setw(8) << C << setw(9) << C << setw(10) << C << '\n';
```

SETPRECISION: **setprecision(#)** specifies the number of significant digits to be displayed (before AND after the decimal point); the value displayed is rounded, not truncated.

Add the following statements to your program and execute to see the effect of the setprecision manipulator.

```
cout << A << '\n';
cout << setprecision(2) << A << '\n';  //What happened??
cout << setprecision(3) << A << '\n';
cout << setprecision(4) << A << '\n';
cout << setprecision(5) << A << '\n';
cout << setprecision(6) << A << '\n';
cout << setprecision(7) << A << '\n';
cout << setprecision(8) << A << '\n';
```

Once the precision is set, it stays in effect until changed by another setprecision statement. It is unnecessary to use it for every value.

Add the following statements to your program and execute to see the effect of the setprecision manipulator as it remains in effect.

```
cout << B << '\n';
cout << setprecision(3) << A << '\n';
cout << setprecision(4) << A << '\n';
cout << A << '\n';
cout << A << '\n';
cout << A << '\n';
```

If the values are too large to be printed in the number of digits specified in setprecision, some systems will print the numbers in scientific (E) notation. You can test this by executing the following statement.

```
cout << setprecision(3) << 56789.432 << '\n';
```

FIXED: **fixed** causes all values to be printed in decimal or fixed point form, not scientific (E);
* **Most common use**: When **fixed** is used with **setprecision**, the precision determines the number of decimal places, not significant digits; stays in effect until changed by another manipulator command

Add the following statements to demonstrate the paired used.

```
cout << A << '\n';
cout << fixed << setprecision(1) << A << '\n';
cout << fixed << setprecision(2) << A << '\n';
cout << fixed << setprecision(3) << A << '\n';
cout << fixed << setprecision(4) << A << '\n';
cout << fixed << setprecision(5) << A << '\n';
cout << fixed << setprecision(6) << A << '\n';
```

Note: If you want all values to be printed with the same number of decimal places, you need only include fixed and setprecision in the first output statement. Demonstrate this with the following statements.

```
cout << fixed << setprecision(2) << A << '\n';
cout << A << '\n';
cout << A << '\n';
```

SHOWPOINT: **showpoint** causes the decimal point to be displayed for all floating point numbers even if it has no decimal digits. Demonstrate with the following statements.

```
cout << C << '\n';
cout << showpoint << C << '\n';
```

Normally, output is right justified within a field, with leading blanks when necessary. You can force data to be left justified if desired.

LEFT, RIGHT: **left, right** causes values to be printed left justified or right justified, respectively. Right is NEVER used alone. Right is the default position. Right is only used to "turn off" the effect of the left manipulator. That is, once the left manipulator is used in an output statement. Demonstrate with the following statements.

```
cout << setw(7) << C << '\n';
cout << left << setw(7) << C << '\n';
cout << setw(7) << C << '\n';
cout << right << setw(7) << C << '\n';
cout << setw(7) << C << '\n';
```

LAB 4 – Assignment

Write a program to print (to a file) the following 2 tables, with columns properly aligned. You can include your data in assignment statements. (I.E. No cin statements necessary.) Be sure to have your name printed to the output file.

```
          A          B          C
***********************************
X1         5         15         25
X10       50        150        250
X100     500       1500       2500
```

Declare & initialize the following variable values in your program then print the table exactly as shown below using the variables:

```
TotalColl = 26572.89087          Sales = 25068.7993
CountyTax = 501.76               StateTax = 1002.75212
TotalTax = 1504.12890
```

```
MONTH:  March 2008
------------------------------
Total Collected     $  26572.89
Sales               $  25068.80
County Tax          $    501.76
State Tax           $   1002.75
Total Tax           $   1504.13
```