Floating-point Output Formats Solutions

Floating-point output

- Write a program that executes the code below

```
double pi {3.1415926535};
cout << pi << endl;                    // Displays 3.14159

double c {299'792'458};
cout << c << endl;                     // Displays 2.997925e+008
```

- Explain the results
  - By default, floating-point numbers are displayed to six significant figures around the decimal point
  - Hence, only the first five digits of pi after the decimal point appear
  - If the number has more than six digits, and they are all before or after the decimal point, scientific notation is used

Scientific Notation

- The following numbers are in scientific notation
- What are their values in "everyday" notation?

```
2.99792E+008        // represents 2.99792 * 10^8 which is 299792000
1.00000E-006        // represents 1.00000 * 10^-6 which is 0.000001
3.14159E+000        // represents 3.14159 * 10^0 which is 3.14159
```

Scientific Manipulator

- What effect does the "scientific" manipulator have?
  - The scientific manipulator causes the stream to display output using scientific notation
- What effect does the "uppercase" manipulator have?
  - The uppercase manipulator causes the stream to use an uppercase 'E' in scientific notation
- What output does the following code give?

```
double pi {3.1415926535};
cout << scientific << pi << endl;                  // Displays 3.141593e+000
cout << scientific << uppercase << pi << endl;     // Displays 3.141593E+000
```

Fixed-point Output

- What is meant by "fixed-point" output?
  - In fixed-point output, the number will always be displayed with six decimal places
  - If the number cannot be displayed in this way, it will be truncated, or padded with 0's as necessary
- What output does the following code give?

```
double c {299'792'458};
cout << fixed << c << endl;          // Padded - displays 299792458.000000
double e {1.602e-19};
cout << fixed << e << endl;          // Truncated - displays 0.000000
```

Restoring Floating-point Defaults

- Why is it important to restore the defaults for floating-point output?
  - These manipulators are all "sticky": they permanently change the behaviour of the stream
  - If other programmers use floating-point output later on in the program, they may not get the results they expect
  - When we have finished our output, we should use the "defaultfloat" manipulator to restore the stream to its original settings
- How do we restore these defaults?
  - We use the "defaultfloat" manipulator

Floating-point Precision

- What is meant by floating-point precision?
  - Floating-point precision determines the number of digits that are displayed
- How can we find the current value of the floating-point precision?
  - Calling the precision member function with no arguments will return the current precision
- How can we set the floating-point precision?
  - Calling precision() with an argument will set the precision to that value