

**Basic Numerical Calculations:****Orbit Calculator**

This program is an intermediate step between the first one – where the entire program was given to you – and all the rest – where you have to write it from scratch. You will be given the skeleton of a program that contains all the code necessary to read input in from a data file, and write the results to an output file. Your assignment is to write the code necessary to perform the required calculations in the indicated locations.

This program will be worth 5% of your final grade. The grade will be broken into two parts – 90% will be for the correctness of output (the EAGS score) and 10% will be for style. As in the first program, you are expected to copy comments, indentation, etc from the given code. For code that you write, you should include a comment describing (briefly) what each computation does. Additionally, you are required to use named constants where appropriate, and provide a comment describing each.

**Equations:**

While the math required to launch a satellite or spacecraft into orbit can become exceedingly complicated, the basic equations require nothing more than simple algebra (and are often presented in high school physics classes). The two key components of an orbiting satellite that are intrinsically linked are **altitude** and **speed**. In order to stay in a circular orbit at a given altitude, a satellite must be moving at a specific velocity. Rather counter intuitively, the lower the altitude, the faster the satellite must be traveling. The relationship between the altitude and the required velocity is given by the equation:

$$V = \sqrt{\frac{\mu}{R}}$$

where:

- $V$  is the required velocity in km/s
- $\mu$  is the product of Newton's gravitational constant, and the mass of the orbiting and orbited bodies (in this case, the mass of the satellite is so small, it is ignored). For a satellite orbiting Earth, the value is **3.99e5** km<sup>3</sup>/s<sup>2</sup>.
- $R$  is the radius of the orbit – the radius of the planet plus the altitude of the orbit. The radius of Earth is **6371** km.

The second calculation we're concerned with is the *period* of the orbit – that is, how long it takes to make a complete circle. The period is given by the equation:

$$T = \frac{2\pi}{\sqrt{\frac{\mu}{R^3}}}$$

In this equation,  $T$  is in seconds. For the output,  $T$  will be in minutes, so you must make the seconds to minutes conversion.

**Math Functions:**

The equations used required the mathematical operations of square root and exponentiation. Neither of these is provided as a simple operator in C++ (that is, to raise a number to the third power, you can not simply write  $x^3$ ). However, both are provided as standard mathematical functions, and can be used by including the header file *cmath*. For example, you can take the square root of a number using something like:

$$y = \text{sqrt}(x);$$

In order to perform exponentiation, use the function `pow( )`:

```
z = pow(x, 4);    // This is x4
```

A more complicated example would be using the Pythagorean Theorem to find the length of the hypotenuse of a right triangle. The equation:

$$c = \sqrt{a^2 + b^2}$$

can be programmed as:

```
c = sqrt(pow(a, 2) + pow(b, 2));
```

More discussion on using functions can be found at the end of the Chapter 3 course notes.

### Input file description and sample:

Your program will read its input from a file `orbit-altitude.dat`. Each line of the input file will contain a single value for the altitude of the orbit:

```
100
150
225
500
1000
35785
```

### Output description and sample:

Your program will write output data to a file named `state.dat`. The output file for the previous input file would look like:

```
Orbital Calculator
Programmer: Chris Knestrick

Altitude (km)    Velocity (km/s)    Period (m)
=====
    100           7.85              86.25
    150           7.82              87.26
    225           7.78              88.77
    500           7.62              94.37
   1000           7.36             104.86
  35785           3.08            1434.20
```

### Notes:

Please read the program source code carefully - comments are provided in order to help guide you. There are two locations where you should add code - one for constants and one for calculations - both of which are clearly marked in the comments.

### **Programming Standards:**

You'll be expected to observe good programming/documentation standards. A copy of *Elements of Programming Style* is included with the course notes — if you don't have a copy I strongly suggest you read the on-line edition (available from the course web page). Some specifics:

- You must include header comments specifying the compiler and operating system used and the date completed.
- Use named constants instead of literal values where appropriate. Each constant must include a comment describing what it is.
- Precede every computation with a description of what it does.
- All real number constants must be of type **double**

**The Source Code:**

```
////////////////////////////////////
// Identification
// (You should substitute the correct information in the following
// lines.)
//
// Title:           Orbital Calculator
// Programmer:      Chris Knestrick
// ID Number:       999-99-9999
// Compiler:        Visual C++ version 6.0
// Platform:        Pentium II 400 / Windows NT
// Last Modified:   July 7, 2001
//
////////////////////////////////////
// Honor Pledge
//   On my honor:
//
//   - I have not discussed the C++ language code in my program with
//     anyone other than my instructor or the teaching assistants
//     assigned to this course.
//
//   - I have not used C++ language code obtained from another student,
//     or any other unauthorized source, either modified or unmodified.
//
//   - If any C++ language code or documentation used in my program
//     was obtained from another source, such as a text book or course
//     notes, that has been clearly noted with a proper citation in
//     the comments of my program.
//
//   - I have not designed this program in such a way as to defeat or
//     interfere with the normal operation of the Automated Grader.
//
////////////////////////////////////
// Purpose of the program:
//
//   This program reads in altitudes for a satellite orbiting the Earth
//   from the file "orbit-altitude.dat". It then calculates the velocity
//   required to maintain an orbit of that altitude as well as the period
//   of the orbit. It prints out a labeled table of results to the file
//   "state.dat", showing velocity and period for each altitude.
//
////////////////////////////////////
//
//   The is the #include section. These statements tell the compiler
//   to read and use variables and functions declared in the specified
//   files.
#include <iostream>
#include <fstream>
#include <iomanip>           // For setw() and setprecision()
#include <cmath>            // For pow() and sqrt()

using namespace std;

int main() {
```

```
////////////////////////////////////
//
// Student Modify - Insert all necessary constants here //
//
////////////////////////////////////

// Variable declarations and initializations - all variables must be
// declared before they can be used.

int    Radius = 0,           // The radius of the orbit
       Altitude = 0;        // The altitude of the orbit.
                               // The program will read the altitude of the
                               // orbit from the input file and store it
                               // here. This variable should then be used in
                               // your computations.

double Velocity = 0,        // The velocity required to maintain an orbit
       Period = 0,         // The period of the orbit
       Temp = 0;           // A temporary variable that you may use to
                               // store any intermediate results of
                               // computations. The program can be correctly
                               // written with or without needing
                               // to use this variable - it is here for your
                               // convience.

ifstream inFile;           // Input filestream
ofstream outFile;         // Output filestream

inFile.open("orbit-altitude.dat"); // Open the input filestream
outFile.open("state.dat");         // Open the output filestream
outFile << fixed << showpoint;     // Formatting for decimals - always
                                   // include.

// Output the header. Since the main while loop prints out results to the
// output file, this must be done before we enter the loop. Otherwise, the
// header would not be printed at the top of the output file

outFile << "Orbital Calculator" << endl;
outFile << "Programmer: Chris Knestricks" << endl << endl;
outFile << "Altitude (km)    Velocity (km/s)    Period (m)" << endl;
outFile << "===== " << endl;

// Read the first value from the output file. This is the "priming read"
inFile >> Altitude;

// Enter the main loop. Inside the loop, the calculations for an orbit with
// a given altitude are performed and the result printed to the output file.
// The next value is then read in, and the process repeats until all the
// values have been read from the file.

while (inFile) {           // While the input file is not empty
```

```
////////////////////////////////////
//
// Student Modify - Insert all necessary computations here //
//
////////////////////////////////////

// Write the results to the output file
outFile << setw(8) << Altitude;
outFile << setw(16) << setprecision(2) << Velocity;
outFile << setw(19) << setprecision(2) << Period;
outFile << endl;

// Get the next value from the input file
inFile >> Altitude;

} // End of while

// We're all out of data, close the filestreams
inFile.close();
outFile.close();

return 0; // Return success

} // End of main
```