16.216: ECE Application Programming

Fall 2012

Programming Assignment #2: Basic I/O and Operations

Due **Monday, 9/17/12**, 11:59:59 PM

1. Introduction

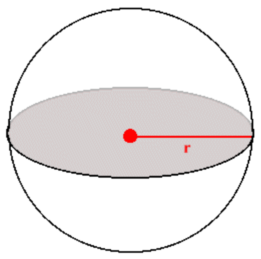
This assignment will give you some experience working with C input (using `scanf()`) and output (using `printf()`), as well as arithmetic operations with variables. You will read input values, use them to calculate some results, and print those values to the screen.

2. Deliverables

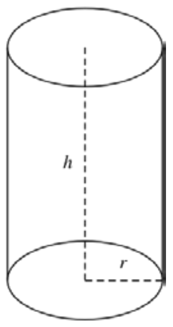
Submit your source file directly to Dr. Geiger (Michael_Geiger@uml.edu) as an e-mail attachment. Ensure your source file name—not your Visual Studio project name—is ***prog2_io.c***. Submit only the .c file. Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

3. Specifications

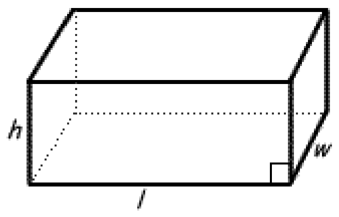
In this program, you will deal with three different types of 3-dimensional shapes: a sphere, a cylinder, and a rectangular prism:



Sphere with radius ***r***



Cylinder with height ***h***, radius ***r***



Rectangular prism with height ***h***, width ***w***, and length ***l***

Input: Your program should prompt the user to enter the dimensions of each shape (assume the units are cm). All dimensions for each shape should be on one line, separated by at least one space. (Note: remember that `scanf()` ignores whitespace when scanning anything other than a single character—you do not have to explicitly worry about varying numbers of spaces.)

A sample run of the program might produce the following first three lines (user inputs are underlined):

```
Enter radius of sphere (cm): 7
Enter radius and height of cylinder (cm): 3.5 0.5
Enter length, width, and height of prism (cm): 1.2 3.4 5.6
```

All input values should be treated as double-precision floating point values.

Output: Once all dimensions have been read, for each shape, your program should print the following:

- A blank line (to separate each shape)
- The name of the shape, in capital letters
- All dimensions (in cm) input for that shape, with one dimension per line
- The surface area of the shape (in square cm)
- The volume of the shape (in cubic cm)

It is assumed that you can find the appropriate formulas to calculate these values. The output lines that would follow the example shown above would be:

```
SPHERE
Radius: 7.000000 cm
Surface area: 615.752159 square cm
Volume: 1436.755039 cubic cm

CYLINDER
Radius: 3.500000 cm
Height: 0.500000 cm
Surface area: 87.964594 square cm
Volume: 19.242255 cubic cm

RECTANGULAR PRISM
Length: 1.200000 cm
Width: 3.400000 cm
Height: 5.600000 cm
Surface area: 59.680000 square cm
Volume: 22.848000 cubic cm
```

See Section 5: Test Cases for more sample program runs.

4. Formulating a solution—hints, tips, etc.

Symbolic constants: Recall that the `#define` directive can be used to assign a name to frequently-used constants to improve program readability. The value π (pi) is used in several area and volume formulas, making it a good candidate for a symbolic constant in this program.

Printing expressions: Values used only for output should not be assigned to a variable—doing so wastes memory space. Recall that `printf()` can print the value of any expression, not just variables. Each of the following lines is therefore a valid use of this function. Assume you have variables `int n` and `double x`:

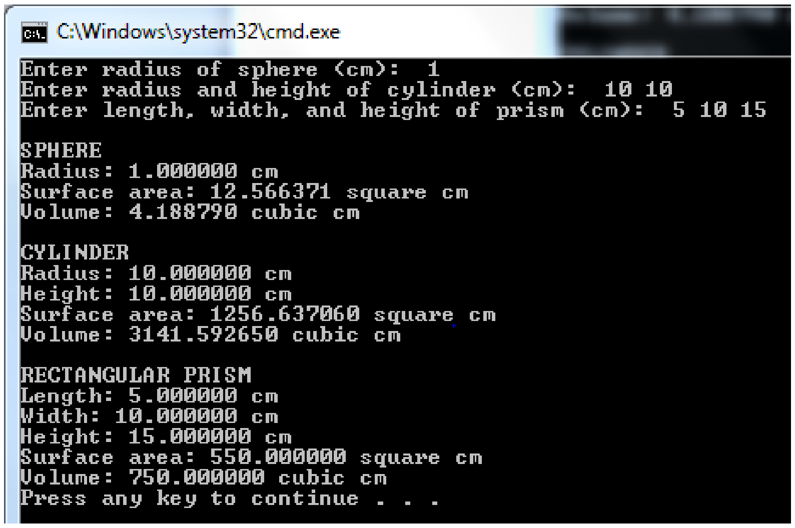
- `printf("n squared: %d, n cubed: %d\n", n * n, n * n * n);`
- `printf("17/x + 35n = %lf\n", (17 / x) + (35 * n));`

-
  ```
  printf("Rectangle with length %d, width %lf has area %lf\n",
     n, x, n * x);
  ```

Integer division: Recall that, when converting a floating-point value to an integer, the fractional part of the number is discarded. This property is particularly important in integer division: for example, in C, `1 / 2 = 0`, since both 1 and 2 are integers. To avoid this issue, one of the two values needs to be a floating-point value. Either `1.0 / 2` or `1 / 2.0` will give the desired result (`0.5`).

5. Test Cases

Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your lab, but will also generate additional cases that will not be publicly available. Note that these test cases may not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.

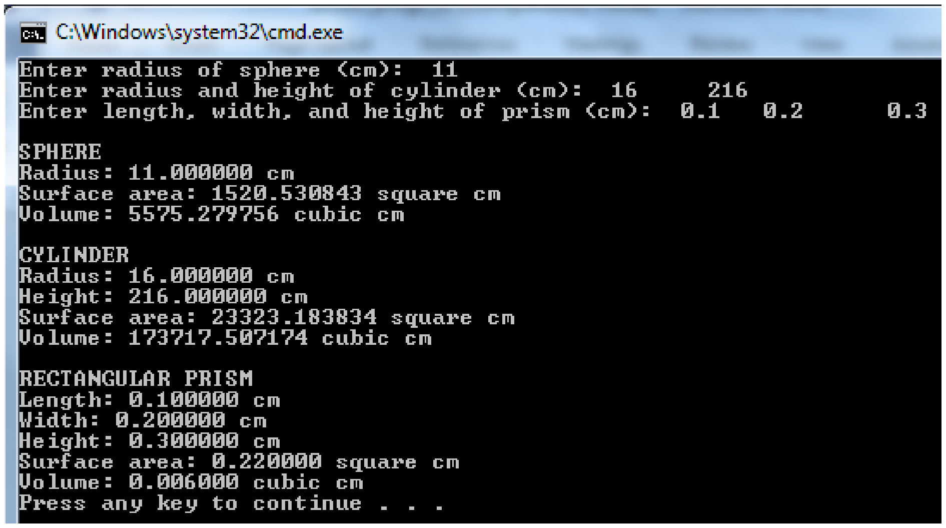


```
Enter radius of sphere (cm):  1
Enter radius and height of cylinder (cm):  10 10
Enter length, width, and height of prism (cm):  5 10 15

SPHERE
Radius: 1.000000 cm
Surface area: 12.566371 square cm
Volume: 4.188790 cubic cm

CYLINDER
Radius: 10.000000 cm
Height: 10.000000 cm
Surface area: 1256.637060 square cm
Volume: 3141.592650 cubic cm

RECTANGULAR PRISM
Length: 5.000000 cm
Width: 10.000000 cm
Height: 15.000000 cm
Surface area: 550.000000 square cm
Volume: 750.000000 cubic cm
Press any key to continue . . .
```



```
Enter radius of sphere (cm):  11
Enter radius and height of cylinder (cm):  16     216
Enter length, width, and height of prism (cm):  0.1   0.2      0.3

SPHERE
Radius: 11.000000 cm
Surface area: 1520.530843 square cm
Volume: 5575.279756 cubic cm

CYLINDER
Radius: 16.000000 cm
Height: 216.000000 cm
Surface area: 23323.183834 square cm
Volume: 173717.507174 cubic cm

RECTANGULAR PRISM
Length: 0.100000 cm
Width: 0.200000 cm
Height: 0.300000 cm
Surface area: 0.220000 square cm
Volume: 0.006000 cubic cm
Press any key to continue . . .
```

Remember, if you are using Visual Studio or Visual C++ Express, to get your program to terminate with a message saying, “Press any key to continue …”, use the **Start Without Debugging** command (press Ctrl + F5) to run your code.