Practical class 2

General comments

The objectives of this second lab session are:

(i) To practise declaring variables and calculating arithmetic expressions.
(ii) To practise formatted reading and writing from stdin/stdout.
(iii) To learn how to read from and write to files.
(iv) To understand the concept of overflow for integer variables.
(v) To investigate the maximum values that can be held in various integer types.

Instructions

1. First, make sure that you've read and understood Self-study package 1 on "Input and output", which can be found under Week 2 on ELE.

2. Then using your preferred IDE (Code::Blocks, Xcode, etc.), write a piece of code that calculates and prints to screen the surface areas and volumes of three geometric figures: a sphere, a cylinder and a cuboid. To do this:

   a) Declare the necessary variables to accommodate all the information you need (radius, length, width, height, surface area and volume).

   b) Ask the user for, and then read in, the information you need. For example, for the sphere you could print something like "Enter the radius of the sphere" and then read in the radius.

   c) Calculate the surface area and volume, and then print the results to screen. Use some text when printing the results, don't simply dump the numbers to the screen.

3. Modify the above code to instead read and write from text files. To do this:

   a) Create three text files called "sphere.txt", "cylinder.txt" and "cuboid.txt" that contain the required geometric parameters for each figure (put all the numbers in a single line to make things simpler).

   b) Following the instructions in Self-study package 1, read in the information from the text files and store in appropriate variables.

   c) After performing the calculations, write the results to a new file called "results.txt".

4. The below code can be used to investigate the maximum value of a signed char: when ch1 is the maximum value, ch1+1 will overflow and be negative. Use similar code to investigate overflow of the char, short, int and long integer types. Consider both signed and unsigned versions. Hence determine the maximum value that each type can take on your system. (NB: you can also get these maximum values from limits.h.)

```c
#include <stdio.h>
int main(void) {
  char ch1 = 127;
  char ch2 = ch1 + 1;
  printf("ch1 = %d, ch2 = %d\n", ch1, ch2);
  return 0;
}
```

5. OPTIONAL, ADVANCED EXTRA. Why does the below attempted simplification of the above code not give the expected output? Hint: read about integer promotion.

```c
#include <stdio.h>
int main(void) {
  char ch1 = 127;
  printf("ch1 = %d, ch2 = %d\n", ch1, ch1+1);
  return 0;
}
```