LOOPS
Lab 1 - Recap

Lab 01 - Selection

For this lab you will adapt the program we wrote in class that calculates the volume of a cylinder, cone, or cube. The volume equations are shown below.

Volume of Sphere: \( V = \frac{4}{3} \pi R^3 \)
Volume of a Cylinder: \( V = \pi R^2 \times \text{height} \)
Volume of a Cone: \( V = \frac{1}{3} \pi R^2 \times \text{height} \)
Volume of a Cube: \( V = \text{height}^3 \)

Note, you must decide on the best way to allow the user to communicate which shape they would like to work with. You could pick any character to represent any shape (i.e. ask the user to press 'y' for a sphere).
while ( boolean expression ) {
    statements to execute....
}

while Loop Syntax
Designing While Loops

- while loops run **until** a condition **fails**
  - Step 1: Determine Ending condition
    - Expressed as a boolean expression
  - Step 2: Initialization
    - Determine what, if any, steps are needed to make sure the “first” execution works as expected.
  - Step 3: Ensure statements within loop will eventually meet end condition
    - Statements inside while loop **must** make some change to the variables in the boolean expression!
Pre-Test

- A while loop is a **pre-test** loop:
  - Before executing any statements within the loop, the boolean expression is checked.

- Often, we will want to execute **at least** once...
  - Execute the statements, *then* test to see if we should repeat...
Do - While Loop

do {
    statements...
} while ( boolean expression );

Always executes at least 1 iteration.
Exercise

- Ask the user to enter a series of numbers
- They can stop by entering “-1”
- Compute and display the sum of all the numbers entered.
Counting v.s. Conditional Loops

- Conditional Loops: We do not know how many times we will loop...
  - Sentinel (number entered != -1)
  - Continue? (y/n)
  - Input validation
Counting Loops

- Counting: We will loop some set number of times, known before loop is entered

```cpp
int x = 0;
while ( x < 30 ) {
    cout << x << endl;
    x++;
}
```

- All counting loops share common features.
for Loops

```c
for ( initialization; test; update) {
    statements...
}

for ( int i = 0; i < 30; i++) {
    statements...
}
```
Estimating PI

- PI can be estimated by computing an infinite series...

\[ \pi = 4 \times (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots) \]

Write a program to estimate PI...
How long does it take to get to 3.14159?
Nested for-loops

// Main Task
for ( int i = 0; i < 3; i++ ) {
    // Task A
    cout << “Hello” << endl;
    // Task B
    for ( int j = 0; j < 3; j++ ) {
        cout << “Goodbye” << endl;
    }
}
Prime Numbers

- Ask the user to enter a positive number between 1 and 50 (input validation) -> N
- Then find the first N prime numbers
  - A number is prime if it is not divisible by any numbers between 1 and the number

- **do-while** loop for validation
- **while** loop for computing enough prime numbers
- **for** loop for checking if the number is prime
The constant “e” is approximately 2.718 and has been calculated to 869,894,101 decimal places.

$e^x$ is approximated by the following series.

$$e^x = x^0/0! + x^1/1! + x^2/2! + x^3/3! + x^4/4! + \ldots + x^n/n!$$

Write a program that asks the user for a positive value for X. Display $e^x$ based on the above approximation where N is 1, 5, 25, and 125.

- $e^5$ (2 iterations) = 6.0000000000
- $e^5$ (6 iterations) = 91.4166666667
- $e^5$ (26 iterations) = 148.4131590981
- $e^5$ (126 iterations) = 148.4131591026

I used setprecision(10) and fixed

http://ncalculators.com/number-conversion/e-x-calculator.htm
Important Tips for the lab

- This can be a tough problem, since you will end up with 3 levels of nesting (at least)
- For each N (N = 1, 5, 25, 125 .... N^* = 5)
  - **EX = 0**;
  - For each i = 0 to N (inclusive)
    - Calculate A = \(x^i\) (use pow function)
    - Calculate B = \(i!\) (this will be another loop)
    - Add A/B to **EX**
- I highly recommend you do N = 5 first, without using any looping for N. Get that number to be correct.
- Print out i on each iteration – along with \(x^i\) and \(i!\). Make sure they are individually correct on each iteration.
- Then wrap your calculation in a loop for N = 1, 5, 25, 125
Next Class

- Please read chapter 5 in the text (FUNCTIONS)
  - We’ll cover functions for the next week or so.
  - Remember to complete the lab for next class (upload to appiversity before class starts.)