Homework:
Bring to your next lab. I will also send the following by email.

Write a complete C++ program that does the following:
1) Prints on the console: "Please enter an angle in degrees: "
2) Reads from the console a double number.
3) Prints out "The sine of the angle is X and the cosine of the angle is Y" where X and Y are the sine and cosine of the number.

Make sure your program is complete! Review TCS 3. You do not need to run the program; type it or handwrite it and bring it to lab.

For Thurs., read TCS 4.

Random Numbers:
Random number are useful for many things in computing:
- Game programs (shuffling a deck of cards, rolling dice,…)
- Simulations of natural phenomena (whether it rains or not, when a car enters the highway,…)
- Statistical sampling
- Some algorithms depend on randomness

It's quite difficult to get a computer to generate a true random number. The reason is that by design computers are deterministic. That means given the same input, they will always give the same output.

What we do have are pseudo-random number generators, which generate a sequence of pseudo-random numbers, that is, numbers that look random, but are perfectly predictable. This is what we usually use when we need random numbers.

Notice that X % Y always gives you a number in the range 0, 1, 2, … Y-1.

So rand() % N is a random number in the range 0, …, N-1

For example rand() % 12 is a random integer in the range 0, …, 11

But we usually don't count months from 0, we count them from 1.
So how do I get a random integer in the range 1, …, 12?

Use 1 + rand() % 12

So the modulus controls the number of possibilities, the number determines where I start counting.

How would I generate random integers in the range X, …, Y?

X + rand() % (Y - X + 1)

Think about this very carefully. Leaving out the "+1" is a common off-by-one error. Off-by-one errors are very common logic errors.

So to generate random integers in the range -10,. …, 10.

Use -10 + rand() % (10 - -10 + 1)
or rand() % 21 - 10 // generate random numbers in -10, …, 10

Every time we use rand() we will get the same random number sequence, unless we "seed" the random number generator with some number that it not control of

the program (because if it is completely in the program's control, we will get the same numbers every time).

We can use the function srand(X) to see the random number generator to X.

One thing you can do is to get the seed from some unpredictable source like the time.

If you need genuine random numbers, you can use radioactive decay, thermal noise, etc.

Conditions:
Conditions test things and return a truth value (known in CS as a Boolean value, named for the logician George Boole).

For example, the condition X > 2 returns 1 (meaning true), if X is greater than 2) and returns 0 (meaning false), if X is not greater than 2.

You can use conditions in while-loops, for-loops, and other places.

```
for (int player = 1; player <= 4; player++) {….}

while (X > 0) { …. }

X == Y  // X is equal to Y
X != Y  // X is not equal to Y
X > Y
X < Y
X >= Y
```

X <= Y

Warning! Don't confuse "X = Y" (assignment) and "X == Y" (condition).

Due to the rules of C++, the compiler will not detect if you wrote one of these and meant the other. Your program will simply be incorrect (have a logic error).

Program Design:

We can design programs:
Top-down
Bottom-up
A combination of both

Top-down: Start with a goal, and divide into subgoals (tasks).
These are divided into sub-sub-tasks, and so on, until we get to stuff that's built-in or provided (e.g. Myro functions).

Bottom-up: Assemble things we've got into more useful things. Until we get what we need.

On the polygon program, think about input checking!
How many sides make sense?
How long are the sides, what makes sense?
On any program it is important to test whether the inputs make sense.