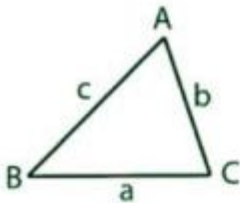


A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with diagonal stripes.

# ProgTeam Spring Week 3

Geometry

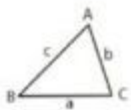
# Important properties:



Law of sines

$$\frac{\sin(A)}{a} = \frac{\sin(B)}{b} = \frac{\sin(C)}{c}$$

Law of Cosines



$$c^2 = a^2 + b^2 - 2ab \cos(C)$$

$$a^2 = b^2 + c^2 - 2bc \cos(A)$$

$$b^2 = a^2 + c^2 - 2ac \cos(B)$$



# Floating Points (The Short Version)

- Floating points will never be exact
- Two workarounds:
  - Comparing with epsilon, ex. `eps = 1e-7`
    - Ex. two points, P and Q
    - **Don't:** `if(P == Q){`
    - **Do:** `if(dist(P,Q) < eps){`
  - Keep everything integers
    - **Don't:** `if(sqrt(dx * dx + dy * dy) < r){`
    - **Do:** `if(dx * dx + dy * dy < r * r){`



# Example: Binary Search

- When binary searching with floating points, don't use !=:
- Approach 1:

```
double lo = 0, hi = 100;
while(hi - lo > eps){
    double mid = (lo + hi) / 2;
```
- Approach 2:

```
double lo = 0, hi = 100;
for(int i = 0; i < 100; i++){
    double mid = (lo + hi) / 2;
```
- $2^{100} > 30$  digits of precision, so this is pretty much always enough



## Note: formatting output

- Unless problem says otherwise, print more digits to avoid rounding errors (10-20 is plenty)
- Example in different languages, print to 10 decimal places
- Java:
  - `println( String.format("%.10f", 3 / 10.0)) );`
- Python:
  - `print('% .10f' % (3 / 10.0))`
- C++:
  - `cout << fixed << setprecision(10) << 3 / 10.0 << endl;`