05 - Making Decisions - Part 1

Dr. Robert Lowe

Division of Mathematics and Computer Science
Maryville College

Outline

Outline

Introduction to Branching

Introduction to Branching

- Branching is when a program selects between multiple paths of execution.

Introduction to Branching

- Branching is when a program selects between multiple paths of execution.
- Programs make choices according to statements which are either true or false.

Introduction to Branching

- Branching is when a program selects between multiple paths of execution.
- Programs make choices according to statements which are either true or false.

What is Truth?

Introduction to Branching

- Branching is when a program selects between multiple paths of execution.
- Programs make choices according to statements which are either true or false.

What is Truth?

In C++, 0 is false. All other values are true.

Introduction to Branching

- Branching is when a program selects between multiple paths of execution.
- Programs make choices according to statements which are either true or false.

What is Truth?

In C++, 0 is false. All other values are true.
We usually use expressions which logically evaluate to true or false as conditions

Comparison Operations

Relational Operators

Comparison Operations

Relational Operators

- $<$ Less Than

Comparison Operations

Relational Operators

- $<$ Less Than
- $>$ Greater Than

Comparison Operations

Relational Operators

- `<` Less Than
- `>` Greater Than
- `<=` Less Than or Equal To

Comparison Operations

Relational Operators

`<`	Less Than
`>`	Greater Than
`<=`	Less Than or Equal To
`>=`	Greater Than or Equal To

Comparison Operations

Relational Operators

- `<` Less Than
- `>` Greater Than
- `<=` Less Than or Equal To
- `>=` Greater Than or Equal To

Equality Operators

Comparison Operations

Relational Operators

- `<` Less Than
- `>` Greater Than
- `<=` Less Than or Equal To
- `>=` Greater Than or Equal To

Equality Operators

- `==` Equal

Comparison Operations

Relational Operators

- `<` Less Than
- `>` Greater Than
- `<=` Less Than or Equal To
- `>=` Greater Than or Equal To

Equality Operators

- `==` Equal
- `!=` Not Equal

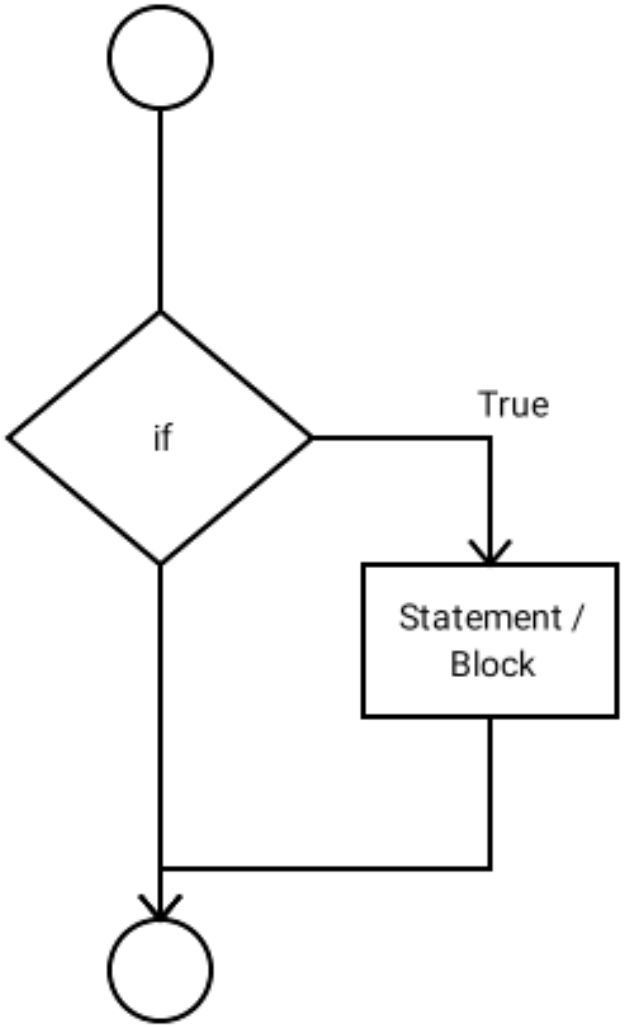
Operator Precedence (Thus Far)

Operator	Description	Associativity
`a*b`, `a/b`, `a%b`	Multiply, Divide, Modulus	Left-to-Right
`a+b`, `a-b`	Addition and Subtraction	Left-to-Right
« , »	Insertion and Extraction	Left-to-Right
`<`, `<=` `>`, `>=`	Relational Operators	Left-to-Right
`==`, `!=`	Equality Operators	Left-to-Right

`=`,
`+=`, `-=`
`*=`, `/=`
`%=`
	Assignment and Assignment	Right-to-Left

If Statement

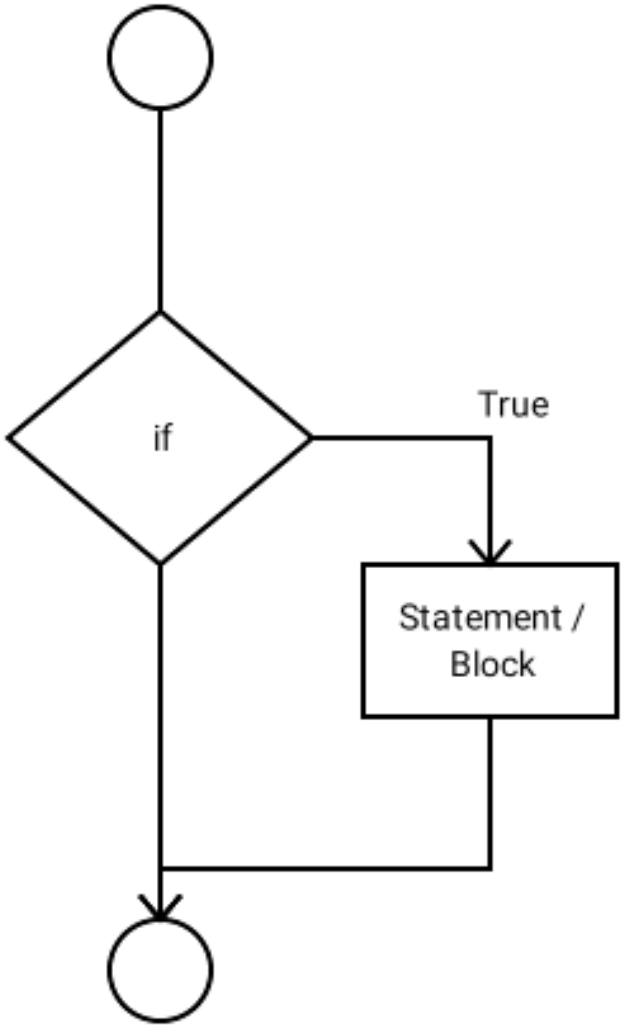
```
if( condition )
    statement/block
```



If Statement

```
if( condition )
    statement/block
```

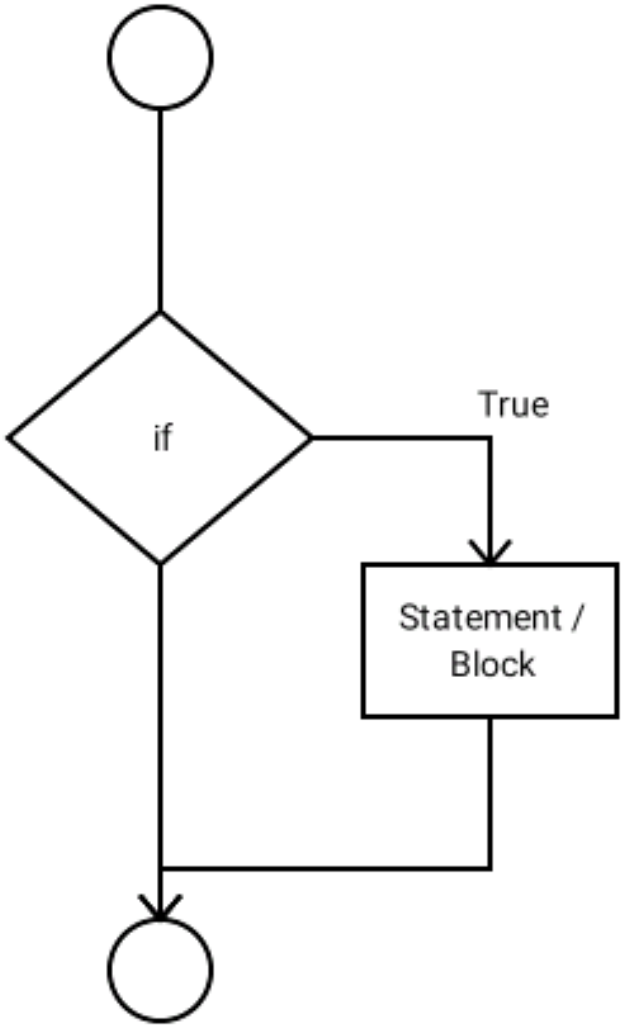
- If the *condition* is true, the *statement/block* will be executed.



If Statement

```
if( condition )
    statement/block
```

- If the *condition* is true, the *statement/block* will be executed.
- If the *condition* is false, the *statement/block* will be skipped.



Example: even.cpp

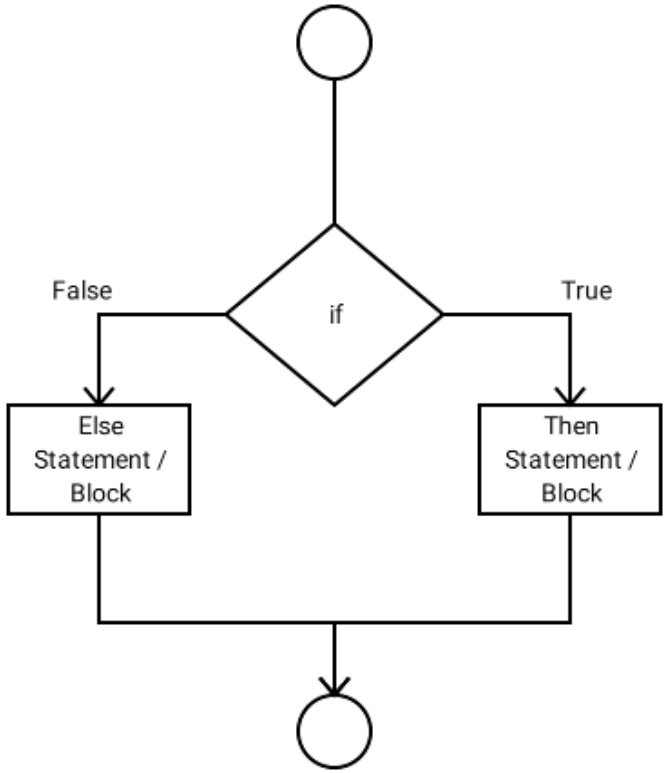
```
int main()
{
    int num;  //the number to test

    //get the number
    cout << "Enter a number: ";
    cin >> num;

    //inform the user if it is even
    if( num % 2 == 0 ) {
        cout << "The number " << num << " is even" << endl;
    }
}
```

If Else Statement

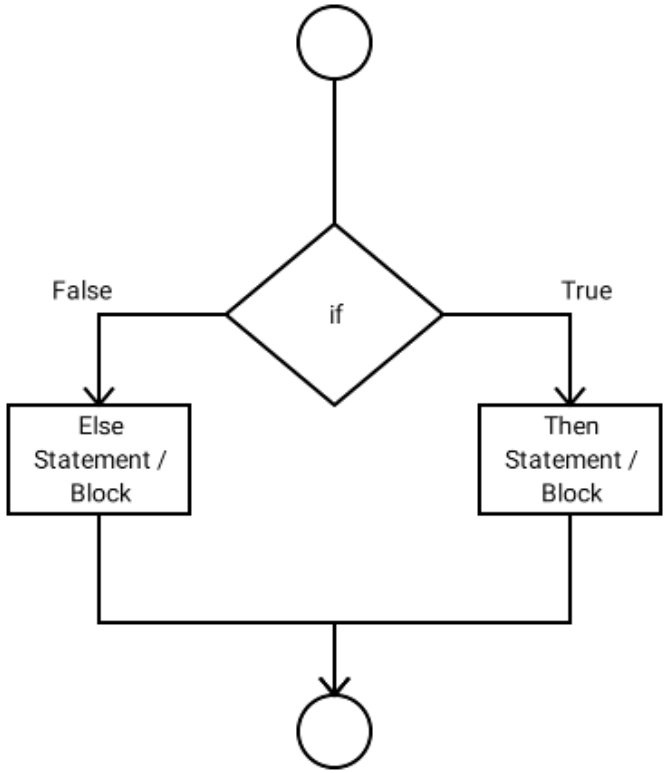
```
if( condition )
        then statement/block
else
        else statement/block
```



If Else Statement

```
if( condition )
      then statement/block
else
      else statement/block
```

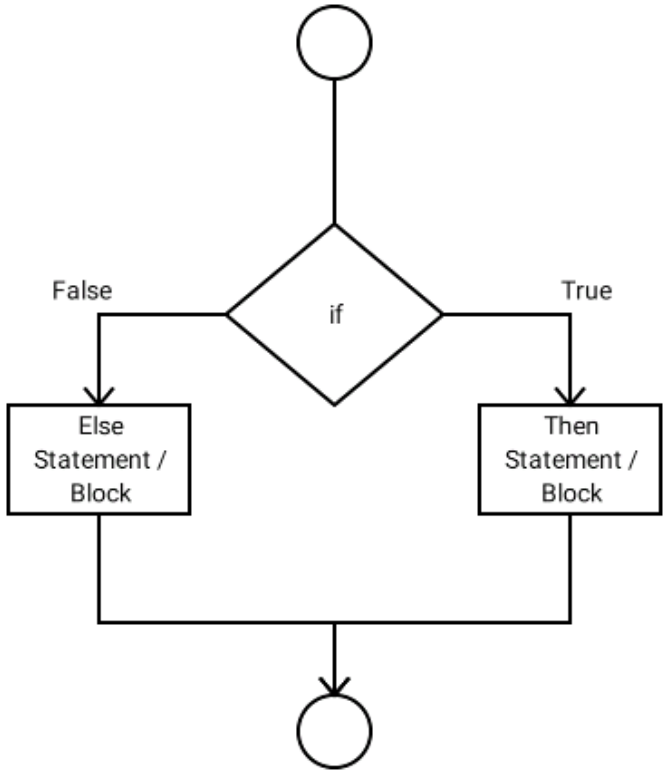
- If the *condition* is true, the *then statement/block* will be executed.



If Else Statement

```
if( condition )
      then statement/block
else
      else statement/block
```

- If the *condition* is true, the *then statement/block* will be executed.
- If the *condition* is false, the *else statement/block* will be executed.



Example: even-odd.cpp

```
int main()
{
    int num;  //the number to test

    //get the number
    cout << "Enter a number: ";
    cin >> num;

    //inform the user if it is even or odd
    if( num % 2 == 0 ) {
        cout << "The number " << num << " is even" << endl;
    } else {
        cout << "The number " << num << " is odd" << endl;
    }
}
```

Best Practices

- Curly braces are optional, but always use them.

Best Practices

- Curly braces are optional, but always use them.
- Do this:

```
if( x == 1 ) {
    cout << "Yes, x is 1" << endl;
}
```

Best Practices

- Curly braces are optional, but always use them.
- Do this:

```
if( x == 1 ) {
    cout << "Yes, x is 1" << endl;
}
```

- Not this:

```
if( x == 1 )
    cout << "Yes, x is 1" << endl;
```

Best Practices

- Curly braces are optional, but always use them.
- Do this:

```
if( x == 1 ) {
    cout << "Yes, x is 1" << endl;
}
```

- Not this:

```
if( x == 1 )
    cout << "Yes, x is 1" << endl;
```

- Be consistent about brace styles!

Best Practices

- Curly braces are optional, but always use them.
- Do this:

```
if( x == 1 ) {
    cout << "Yes, x is 1" << endl;
}
```

- Not this:

```
if( x == 1 )
    cout << "Yes, x is 1" << endl;
```

- Be consistent about brace styles!
- Always indent the contents of the if's block.

Outline

Pythagoras

- Pythagoras lived in Croton around 500 BC.

URL: `https://www.youtube.com/watch?v=Vn4rwks1eMk`

Pythagoras

- Pythagoras lived in Croton around 500 BC.
- Taught that everything could be quantified by whole units (integers).

URL: `https://www.youtube.com/watch?v=Vn4rwks1eMk`

Pythagoras

- Pythagoras lived in Croton around 500 BC.
- Taught that everything could be quantified by whole units (integers).
- Founded a mystery cult, called the Pythagoreans.

URL: `https://www.youtube.com/watch?v=Vn4rwks1eMk`

Pythagoras

- Pythagoras lived in Croton around 500 BC.
- Taught that everything could be quantified by whole units (integers).
- Founded a mystery cult, called the Pythagoreans.
- The Pythagoreans kept all their math secret.

URL: `https://www.youtube.com/watch?v=Vn4rwks1eMk`

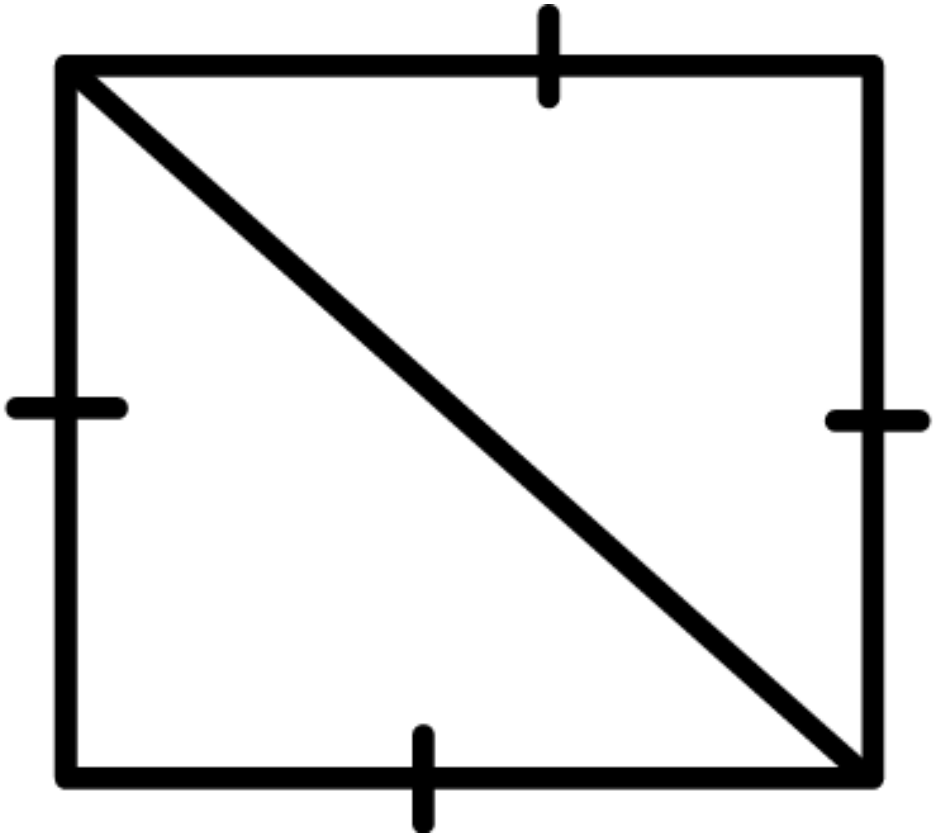
Pythagoras

- Pythagoras lived in Croton around 500 BC.
- Taught that everything could be quantified by whole units (integers).
- Founded a mystery cult, called the Pythagoreans.
- The Pythagoreans kept all their math secret.
- The Pythagoreans died under siege by burning themselves to death on a funeral pyre made of their mathematical works.

URL: `https://www.youtube.com/watch?v=Vn4rwks1eMk`

The Problem

One day, a student discovered that irrational numbers necessarily exist, even in simple observable shapes!



The Solution

This caused Pythagoras a lot of distress.

The Solution

This caused Pythagoras a lot of distress.
Luckily, his students found a solution!

The Solution

This caused Pythagoras a lot of distress.
Luckily, his students found a solution!

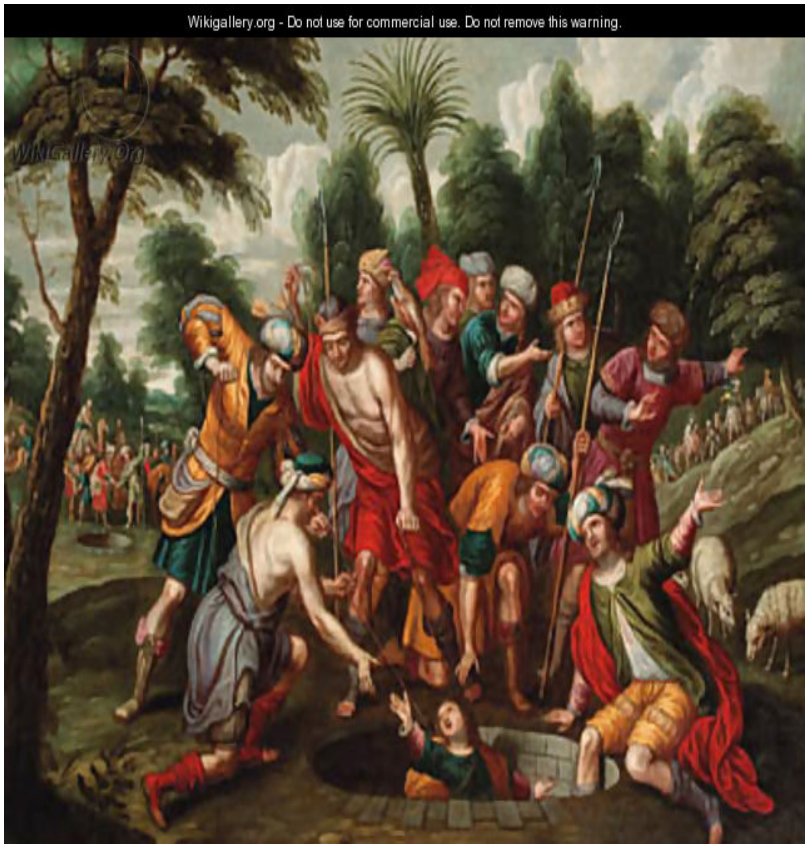
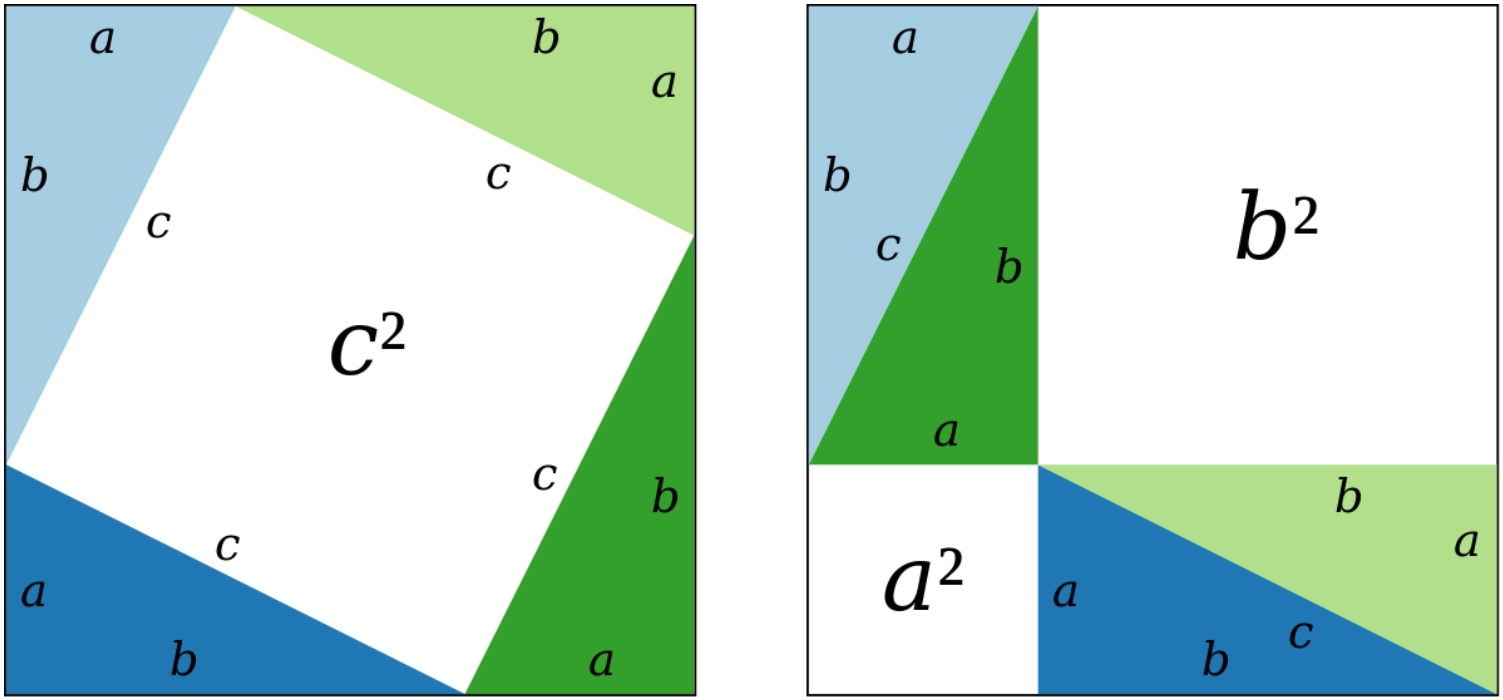


Image Source: `http://www.wikigallery.org/wiki/painting_285033/Peeter-Sion/Joseph-lowered-into-the-well-by-his-brothers`

The Real Solution



$$c^2 = a^2 + b^2$$

Image Source: `https://commons.wikimedia.org/wiki/File:Pythagoras-proof-anim.svg`

Pythagoras Design

Let's design a program that can compute any side of a right triangle!
It can find a leg:

```
I know how to find:
  1.) A Leg
  2.) A Hypotenuse
Which do you want? 2
A=3
C=5
A=3
B=4
C=5
```

Pythagoras Design

It can find the hypotenuse:

```
I know how to find:
  1.) A Leg
  2.) A Hypotenuse
Which do you want? 2
A=3
B=4
A=3
B=4
C=5
```

pythagoras.cpp: Variables

```
double a, b, c;   //triangle sides
int choice;       //the user's choice
```

pythagoras.cpp: Getting the Choice

```
//get the user's choice
cout << "I know how to find: " << endl
     << "  1.) A Leg" << endl
     << "  2.) A Hypotenuse" << endl
     << "Which do you want? ";
cin >> choice;
```

pythagoras.cpp: Making the Decision

```
//perform the user's choice
if(choice == 1) {
    //find a leg
} else {
    //find the hypotenuse
}
```

pythagoras.cpp: Find a Leg

```
//get A and C
cout<< "A=";
cin >> a;
cout << "C=";
cin >> c;

//solve for b
b = sqrt(c*c - a*a);
```

pythagoras.cpp: Find the Hypotenuse

```
//get A and B
cout << "A=";
cin >>  a;
cout << "B=";
cin >> b;

//solve for c
c = sqrt(a*a + b*b);
```

pythagoras.cpp: Display the Results

```
//print the results
cout << "A=" << a << endl
     << "B=" << b << endl
     << "C=" << c << endl;
```

Challenge: Modify `quadratic.cpp`

Make the following changes to the behavior of the quadratic program we wrote last time.

1. Only display the roots if the equation has real roots.
2. If there are no real roots of the equation, display the message "No real roots."
3. If both roots are the same, only display one root. (Do not repeat roots.)

Week 3 Lab Requirements

For full credit you must have:

1. `circle.cpp`
2. `pythagoras.cpp`
3. `quadratic.cpp` (as modified on the previous slide).

Be sure to add, commit, and push!