Mathematical Functions

Mathematical Functions

- When you want to use a mathematical functions including:

  `sine`, `cosine`, `log`, and `square root`

- Use the math library
  - Include the following at the top of your program

```
#include <cmath>
```

Review: Math Functions

- A function has three parts:
  - Name
  - Input parameters
  - Evaluates to an output

- Remember `y = f(x)`

  - `f` is the name of the function
  - `x` is input parameter
  - `y` is the output

sqrt function

- The square root function
  - Name: `sqrt`
  - Input parameter: a single decimal number
  - Output: The square root of the input decimal number
- Function characteristics:
  - Descriptive name
  - 0 or more input parameters
  - One output
- ***Function call***: A function name and its parameters
  - Example: sqrt(81.0)

Example : Code Trace

```
double a = 81.0;
double b = sqrt(a);     <-- RHS is a function call
```

- The 2nd line will execute as follows:

```
double b = sqrt(a);     <-- Before execution

double b = sqrt(81.0);  <-- Evaluates variable a

double b = 9.0;         <-- Evaluates sqrt(81.0)

double b = 9.0;         <-- Assigns 9.0 to variable b
```

- **A function call is replaced by its output during execution!**

Example: Code Trace

```
double a = 81.0;
sqrt(a);
```

- The 2nd line will execute as follows:

```
sqrt(a);        <-- Before execution
sqrt(81.0);     <-- Evaluates variable a
9.0;       <-- Evaluates sqrt(81.0) and throws away the output!
```

- This is NOT a syntax error!
  - It is nonsensical
- You should do something with a ***function call***, for example

```
double a = 81;
cout << sqrt(a); // or
double b = sqrt(a);
```

Example: Code Trace

```
double a = 81.0;
double b = sqrt(a);
double c = sqrt(sqrt(a)) * b;
```

- The 3rd line will execute as follows:

```
double c = sqrt(sqrt(a)) * b;

double c = sqrt(sqrt(81.0)) * b;

double c = sqrt(9.0) * b;

double c = 3.0 * b;

double c = 3.0 * 9.0;

double c = 27.0;     <-- Assign variable c to 27.0
```

Expressions as Parameters

```
double a = 81.0;
double b = 10.45;
```

- An input parameter to a ***function call*** can be any arithmetic expression

```
        sqrt(sqrt(a) * b)
```

- The output of a function replaces its ***function call*** during execution

Your Turn

```
double a = 81.0;
double b = sqrt(a);
double c = sqrt(sqrt(a) * b) * b;
```

- What are the values of variables a, b, and c?

a = 81.0

b = 9.0

c = 81.0

pow function

- The power function, for example `pow(b, e)`
  - Name: `pow`
  - Input parameters: Two decimal numbers
    - The 1st parameter is the base
    - The 2nd parameter is the exponent
  - Output: The base raised to the exponent, i.e. b^e

- Example,

```
cout << pow(3.0, 4.0);
```

  - Output is base 3.0 raised to the exponent 4.0

- Separate more than one input parameter with commas

Your Turn

```
double x = 1.0;
cout << pow(sqrt(25.0) + 2.0, x + 1.0);
```

- What is displayed to the screen?

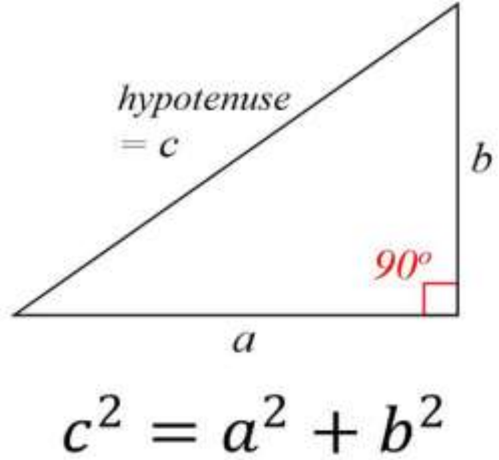
- Remember
  - An input parameter can be any ***arithmetic expression***
  - Function calls can be part of an ***arithmetic expression***

cmath library common math functions

Function	Returned Value	Data Type of Returned Value
abs(a)*	Absolute value of a	Data type of a
pow(b,e)	Value of b raised to the eth power	float or double Data type of b
sqrt(a)	Square root of a	double
sin(a)	Sine of a in radians	double
cos(a)	Cosine of a in radians	double
tan(a)	Tangent of a in radians	double
log(a)	Natural log of a	double
log10(a)	Common log (base 10) of a	double
exp(a)	e raised to the ath power	double

Your Turn: Distance between two lines (All Row 1's)

- Write a C++ program to compute the length of the hypotenuse of a triangle where one point is the origin (0,0)
  - Program should prompt user for (x,y)
  - Program should calculate distance from (0,0) to (x,y)
- How do we solve this?
  - Pythagorean Theorem
- Choose a partner & code this



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

Your Turn: Free Fall (All Row 2's)

- Write a C++ program to compute the time in seconds (*t*) for an object to fall a given height in feet (*h*)

$$g = \frac{2h}{t^2}$$

- Solve for *t* (time in seconds)
  - Need `sqrt` function
  - *g* = acceleration = 9.8 m/s^2 = 32.2 ft/s^2
  - Therefore: *t = sqrt(2h / 32.2)*

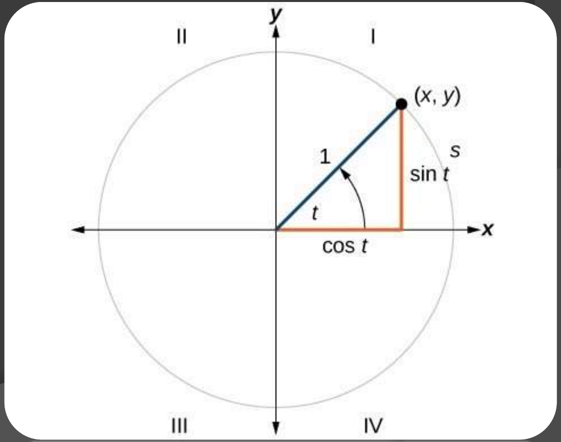
Your Turn: Money Doubles!

- Given an arbitrary `interest rate`, how many `years` will it take for my investment to double in value?
- Solve for `years`
- `years` = ln 2.0 / ln (1 + `interest rate %`)
- Write a program to ask for the interest rate, then solve for `years` until the money doubles

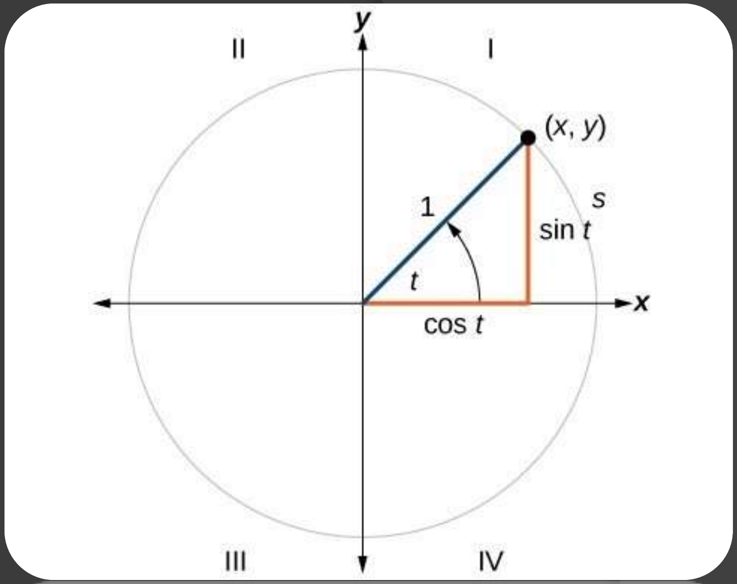
Trigonometric Math Functions

- The trigonometric math functions require the input parameters to be in ***radians***
  - For example: `sin`, `cos`, and `tan`
- Example problem:
  - If `point(1,0)` on a graph is rotated `angle radians`, what is the `new point`?
  - Calculate `new(x,y)` as

```
x = cos(angle)
y = sin(angle)
```



Example: sin(), cos()



```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  double angle(0.0), x(0.0), y(0.0);

  cout << "Enter rotation angle (radians): ";
  cin >> angle;

  x = cos(angle);
  y = sin(angle);
  cout << "Point (1,0) rotates to point"
       << "(" << x << "," << y << ")" << endl;

  return 0;
}
```

In & Out always in radians

```
…
  cout << "Enter rotation angle (radians): ";
  cin >> angle;

  x = cos(angle);
  y = sin(angle);
  cout << "Point (1,0) rotates to point"
       << "(" << x << "," << y << ")" << endl;
…
```

```
> rotate_radians.exe
Enter rotation angle (radians): 1.57
Point (1,0) rotates to point (0.000796327,1)

> rotate_radians.exe
Enter rotation angle (radians): 0.78
Point (1,0) rotates to point (0.710914,0.703279)
```

Common Math Constants

- The `cmath` library file also defines commonly used *math constants*

- Use constants like variables
  - But you CANNOT assign a new value to them

Constant	Value
`M_PI`	π, 3.14159…
`M_E`	e, the base of natural logarithms
`M_LOG2E`	Base-2 logarithm of e
`M_LOG10E`	Base-10 logarithm of e
`M_LN2`	Natural log of 2
`M_LN10`	Natural log of 10

degrees2radians.cpp

```
#include <iostream>
#include <cmath>         // cmath contains definitions of math
   constants
using namespace std;

int main()
{
  double degrees(0.0), radians(0.0);

  cout << "Enter angle in degrees: ";
  cin >> degrees;

  radians = (degrees * M_PI) / 180.0; // Convert degrees to
   radians

  cout << "Angle in radians = " << radians << endl;

  return 0;
}
```

$$1^\circ = \frac{\pi}{180^\circ}$$

rotate_degrees.cpp

```
#include <iostream>
#include <cmath>  // cmath contains definitions of math constants
using namespace std;


int main()
{
  double degrees(0.0), radians(0.0), x(0.0), y(0.0);

  cout << "Enter rotation angle (degrees): ";
  cin >> degrees;

  radians = (degrees * M_PI) / 180.0;
  x = cos(radians);
  y = sin(radians);
  cout << "Point (1,0) rotates to point (" << x << "," << y <<
   ")" << endl;

 return 0;
}
```

```
…
   cout << "Enter rotation angle (degrees): ";
   cin >> degrees;

   radians = (degrees * M_PI) / 180.0;
   x = cos(radians);
   y = sin(radians);
   cout << "Point (1,0) rotates to point"
        << "(" << x << "," << y << ")" << endl;
…
```

```
> rotate_degrees.exe
Enter rotation angle (degrees): 90
Point (1,0) rotates to point (6.12323e-17,1)

> rotate_degrees.exe
Enter rotation angle (degrees): 45
Point (1,0) rotates to point (0.707107,0.707107)
```

log_2.cpp

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  double x(0.0), y(0.0);

  cout << "Enter number: ";
  cin >> x;

  y = log(x) / M_LN2;
  cout << "log(" << x << ") = " << log(x) << endl;
  cout << "log_2(" << x << ") = " << y << endl;

  return 0;
}
```

Arguments to Math Functions

- Input parameters to math functions should have type `double`
- If parameters is type `int`, it will be coerced to `double`
  - Or multiply `int` by 1.0 to get double
- For example,

```
int x(3);
double y(0.3), z(0.0);
z = sqrt(9);
z = sin(1.2);
z = sqrt(x);
z = log(3.2 * x);
z = pow(x / 0.5, 1.2 * y);
```

Mixed Mode Calculations

- Arguments to functions should always be double
  - Or multiply int by 1.0 to get double

- Mixed mode operations:

  (3.0 + 5) or (3.0 * 5) have type double;

- Operator precedence:

  Multiplication and division before addition and subtraction

Math in C++ Review

- Use `#include<cmath>` for math functions

- Common math functions:

  ```
  abs(a)*, pow(b,e), sqrt(a), sin(a),
  cos(a), tan(a), log(a), log10(a),
  exp(a) (there are others)
  ```

- Common math constants:

  ```
  M_PI, M_E, M_LN2, M_LN10
  ```

* cmath library no longer needed for abs( )