

dailyRoutines.cpp



C++ functions()

Robotics 102
Introduction to AI and Programming
University of Michigan and Berea College
Fall 2021

Done

hello
Hello World!
Chad is in Robotics 102"

calculator (Version 24)

```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 plus 7? 29
What is 22 minus 7 ? 15
What is 22 times 7 ? 154
What is 22 divided by 7 ? 3.14286
```

- Program Structure
- Compile/Execute
- Operators
- Data Types
- Variables
- User Input/Output
- Functions
- Branching
- Iterators
- Vectors
- Structs
- File Input/Output



Coming

wall_follower.cpp - Project 1

```
while (true) {
    LidarScan scan = readLidarScan(drv);

    if (min_idx == -1) {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = -1;
        float dist_to_wall = 100.0f;
        float dir_to_wall = 0.0f;
    }

    std::cout << "dist_to_wall: " << dist_to_wall << " dir_to_wall: " << dir_to_wall << std::endl;

    // Compute a vector that points towards the closest obstacle.
    Vector3D robot_to_wall_v;
    Vector3D forward_v = ...;

    // Create a vector that points up.
    Vector3D up_v = ...;

    // Get a vector that is perpendicular to the nearest obstacle.
    Vector3D forward_dir = forward_v - robot_to_wall_v;
    float vx = forward_dir.x;
    float vy = forward_dir.y;
    std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

    vx += ...;
    vy += ...;

    drive(vx, vy, 0);
}
```



calculator.cpp (Version 24)

```
#include <iostream>

/* Let's write a calculator program for real numbers with variables
   that takes numbers from user input (no more magic numbers!) */

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    char additionCharacter = '+'; // Character, for plus
    char subtractionCharacter = '-'; // Character, for minus
    char multiplicationCharacter = '*'; // Character, for times
    char divisionCharacter = '/'; // Character, for division

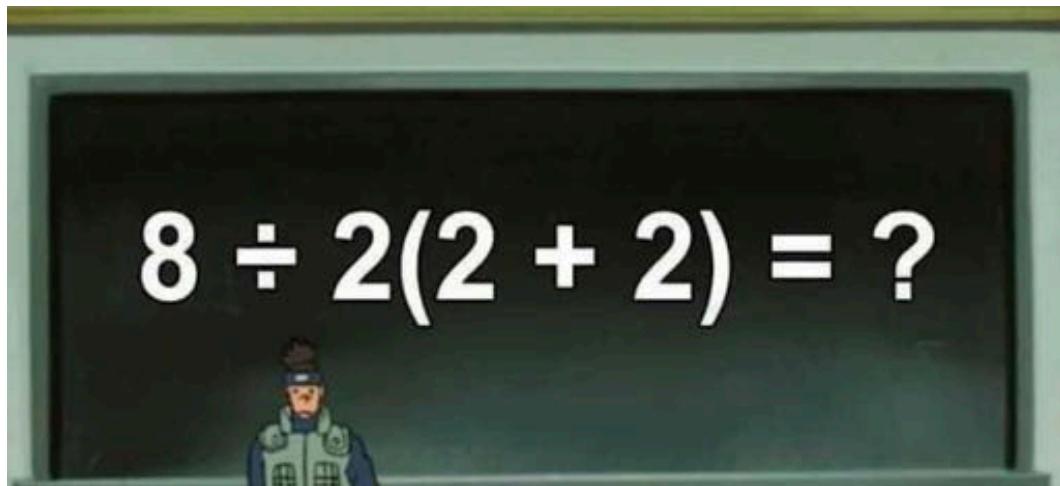
    // Perform all operations and output result to screen
    std::cout << myNumber << additionCharacter << myOtherNumber << "= "
        << myNumber + myOtherNumber << "\n";
    std::cout << myNumber << subtractionCharacter << myOtherNumber << "= "
        << myNumber - myOtherNumber << "\n";
    std::cout << myNumber << multiplicationCharacter << myOtherNumber << "= "
        << myNumber * myOtherNumber << "\n";
    std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
        << myNumber / myOtherNumber << "\n";
}
```

```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 plus 7? 29
What is 22 minus 7 ? 15
What is 22 times 7 ? 154
What is 22 divided by 7 ? 3.14286
```

Our main is not itself

Functions organize programs

A Brief Tangent



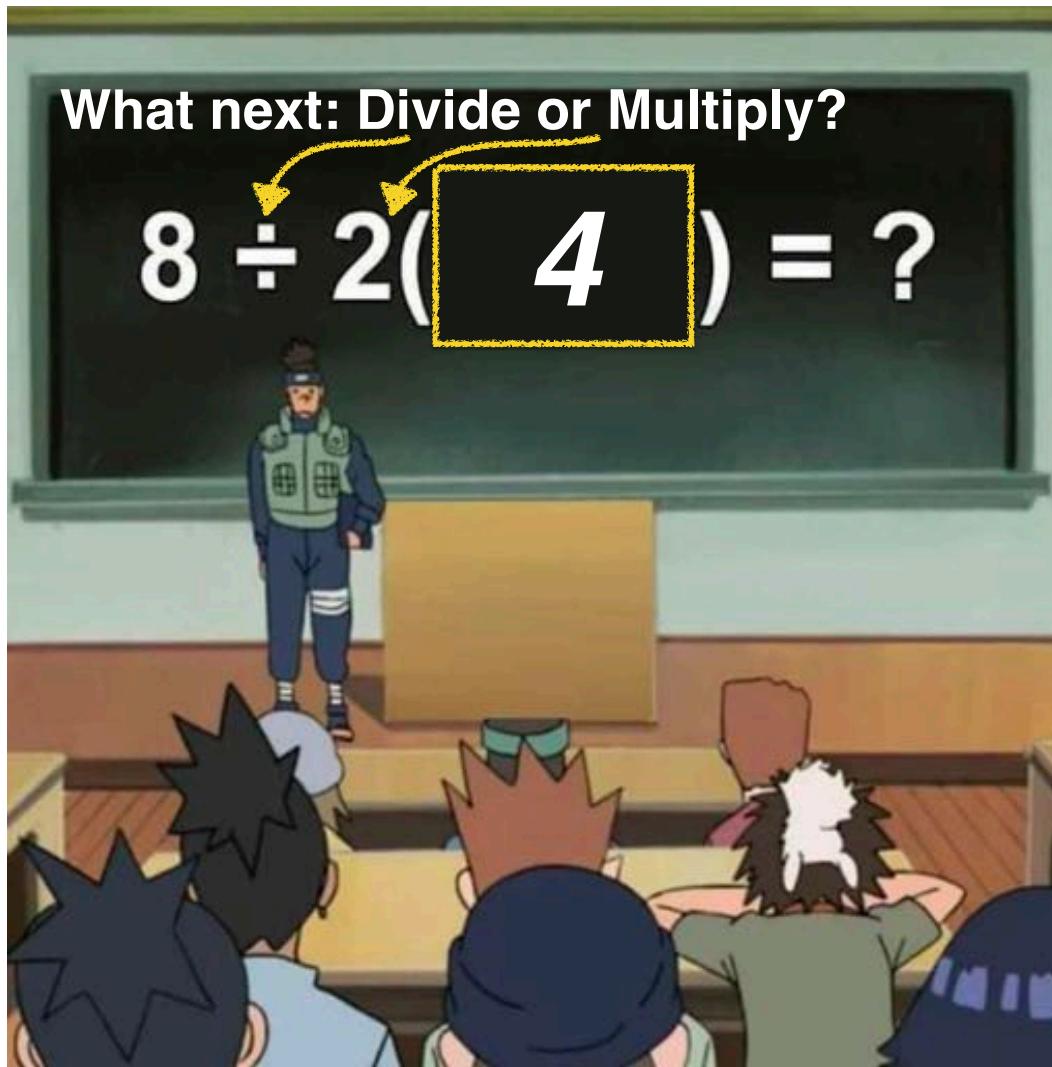
en.wikipedia.org/wiki/Order_of_operations

The order of operations, which is used throughout mathematics, :

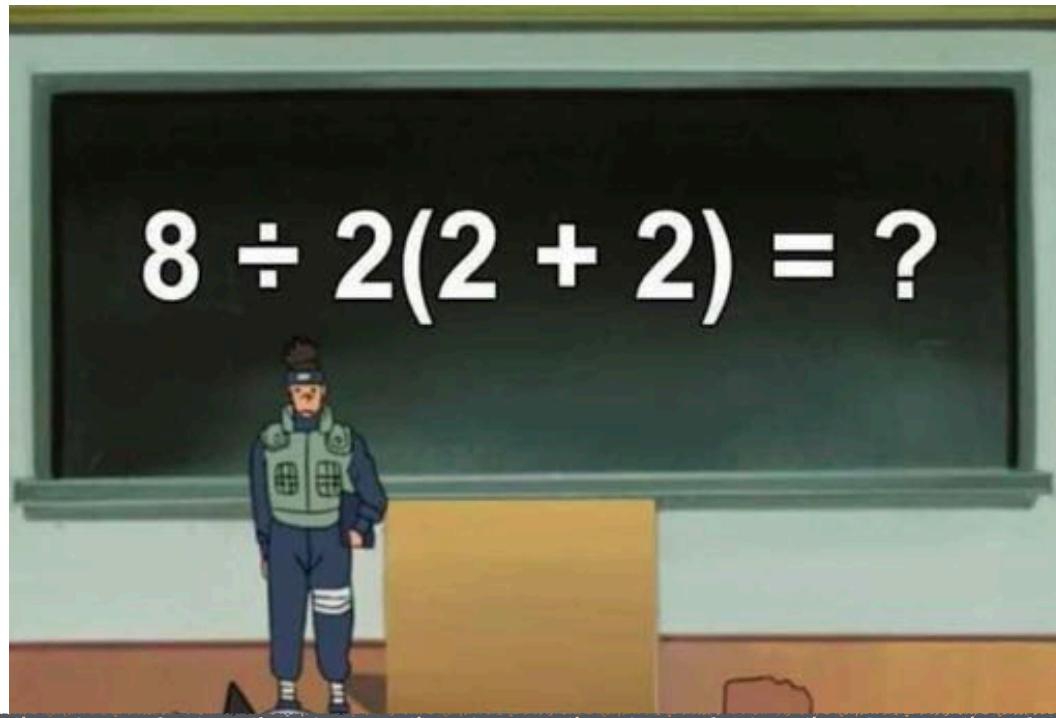
1. **exponentiation** and **root extraction**
2. **multiplication** and **division**
3. **addition** and **subtraction**

In the United States, the acronym **PEMDAS** is common.

It stands for *Parentheses, Exponents, Multiplication/Division, Addition/Subtraction.*



16



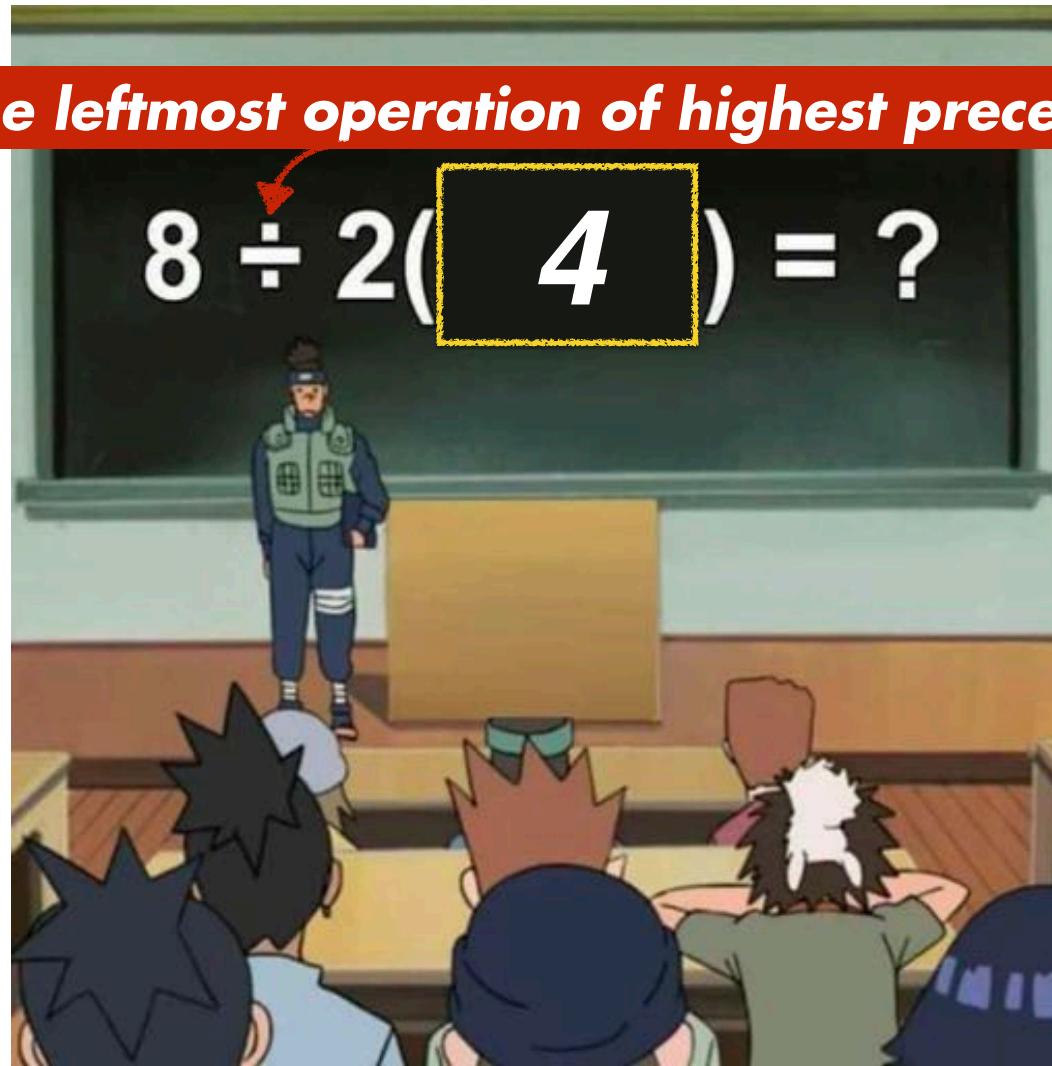
Hello Iruka Sensei!
The answer is: 16

naruto.cpp

```
#include <iostream>

int main()
{
    std::cout << "Hello Iruka Sensei!" << "\n";
    std::cout << "The answer is: " << 8/2*(2+2) << "\n";
}
```

Perform the leftmost operation of highest precedence



Operators and Precedence

- A subset of C++ operators in order of precedence

- grouping:

/*	*/
open comment	close comment

//
comment to end of line

()
open parenthesis	close parenthesis
- increment/decrement:
 ++ --
 increment variable decrement variable
- arithmetic:

*	/	%
multiplication	division	modulus

+	-
addition/concatenation	subtraction
- comparison:
 < <= > >= == != && ||
 less than less than or equal greater than greater than or equal equality inequality logical AND logical OR
- assignment:

=
assignment

+=
add to variable

*=
multiply to variable

Be careful with precedence

calculator.cpp (Version 24)

```
#include <iostream>

/* Let's write a calculator program for real numbers with variables
   that takes numbers from user input (no more magic numbers!) */

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    char additionCharacter = '+'; // Character, for plus
    char subtractionCharacter = '-'; // Character, for minus
    char multiplicationCharacter = '*'; // Character, for times
    char divisionCharacter = '/'; // Character, for division

    // Perform all operations and output result to screen
    std::cout << myNumber << additionCharacter << myOtherNumber << "= "
        << myNumber + myOtherNumber << "\n";
```

calculator.cpp (Version 24 - Branch 01)

```
#include <iostream>

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    // Compute the average of two numbers?
    std::cout << "What is the average of " << myNumber << " and "
        << myOtherNumber << "? "
        << myNumber + myOtherNumber / 2 << "\n"; // Not average of two numbers
}
```

*A branch of our code for a digression
about operator precedence*

calculator.cpp (Version 24 - Branch 01)

```
#include <iostream>

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    // Compute the average of two numbers?
    std::cout << "What is the average of " << myNumber << " and "
        << myOtherNumber << "? "
        << myNumber + myOtherNumber / 2 << "\n"; // Not average of two numbers
}
```

Please type a number and press enter: 22
Please type another number and press enter: 7
What is the average of 22 and 7? 25.5

That Ain't Right

calculator.cpp (Version 24 - Branch 02)

```
#include <iostream>

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    // Parenthesis containing groupings of C++ operations
    std::cout << "What is the average of " << myNumber << " and "
        << myOtherNumber << "? "
        << (myNumber + myOtherNumber) / 2 << "\n";
}
```

Please type a number and press enter: 22
Please type another number and press enter: 7
What is the average of 22 and 7? 14.5

Program output correct

calculator.cpp (Version 24 - Branch 03)

```
#include <iostream>
#include <cmath> // include cmath library for more math functions

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    // Parenthesis containing groupings of C++ operations
    std::cout << "What is the average of " << myNumber << " and "
        << myOtherNumber << "? "
        << (myNumber + myOtherNumber) / 2 << "\n";

    // Function calls to math functions (we will discuss functions next)
    std::cout << "What is " << myNumber << " to the power of " << myOtherNumber
        << "? " << pow(myNumber, myOtherNumber) << "\n";
    std::cout << "What is the cosine of " << myNumber << "? "
        << cos(myNumber) << "\n";
}
```

calculator.cpp (Version 24 - Branch 03)

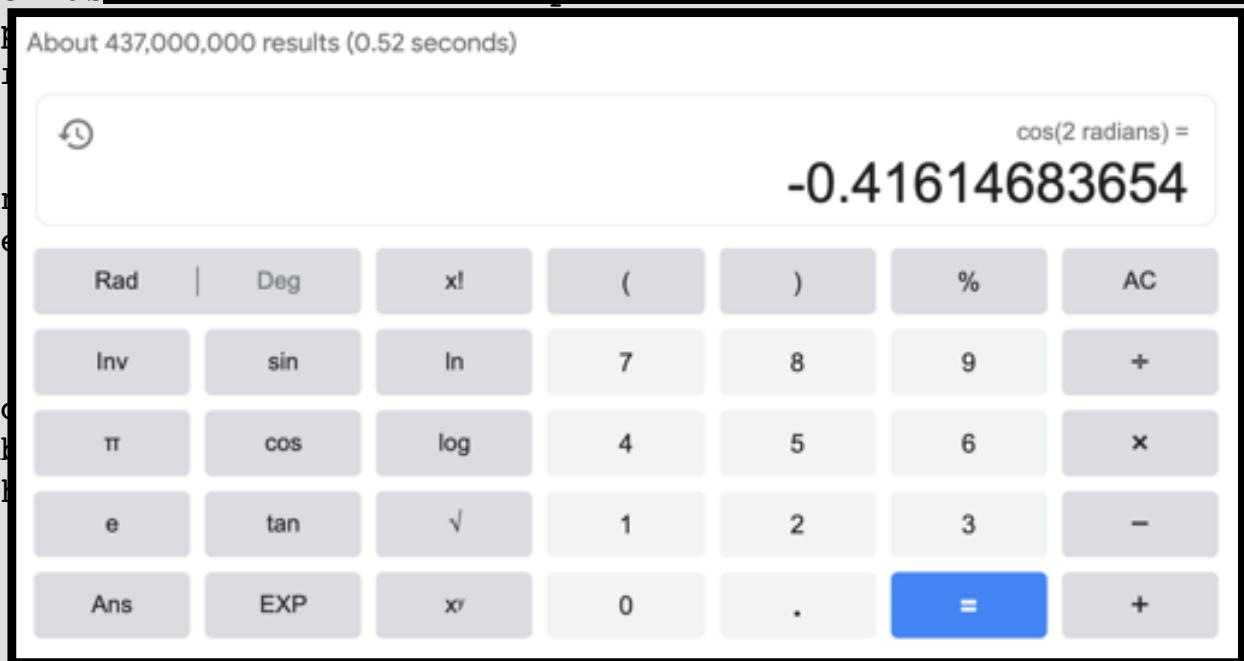
```
#include <iostream>
#include <cmath> // include cmath library for more math functions

int main()
{
    // Ask the user to give us two numbers
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user input
    // Ask the user for our second operation
    std::cout << "Please type another number and press enter: ";
    std::cin >> myOtherNumber;

    // Parenthesis containing grouping symbols
    std::cout << "What is the average of ";
    std::cout << myNumber << "? ";
    std::cout << (myNumber + myOtherNumber) / 2;

    // Function calls to math functions
    std::cout << "What is " << myNumber << "? ";
    std::cout << pow(myNumber, myOtherNumber);
    std::cout << "What is the cosine of ";
    std::cout << cos(myNumber) << "\n";
}
```

```
Please type a number and press enter: 2
Please type another number and press enter: 3
What is the average of 2 and 3? 2.5
What is 2 to the power of 3? 8
What is the cosine of 2? -0.416147
```



calculator.cpp (Version 24 - Branch 03)

```
#include <iostream>
#include <cmath> // include cmath library for more math functions
```

```
int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber;
    // Ask the user for another number
    std::cout << "Please type another number and press enter: ";
    std::cin >> myOtherNumber;
```

Includes a wealth of mathematical functions

What is a function ?

pow

raises a number to the given power (z^y)
(function)

cos

computes cosine (cos z)
(function)

```
/>
std::cout << "What is " << myNumber << " to the power of " << myOtherNumber
    << "? " << pow(myNumber, myOtherNumber) << "\n";
std::cout << "What is the cosine of " << myNumber << "? "
    << cos(myNumber) << "\n";
```

Function calls

What is a function ?

Intuitively, similar to a mathematical function

`cos(myNumber)`

`pow(myNumber, myOtherNumber)`

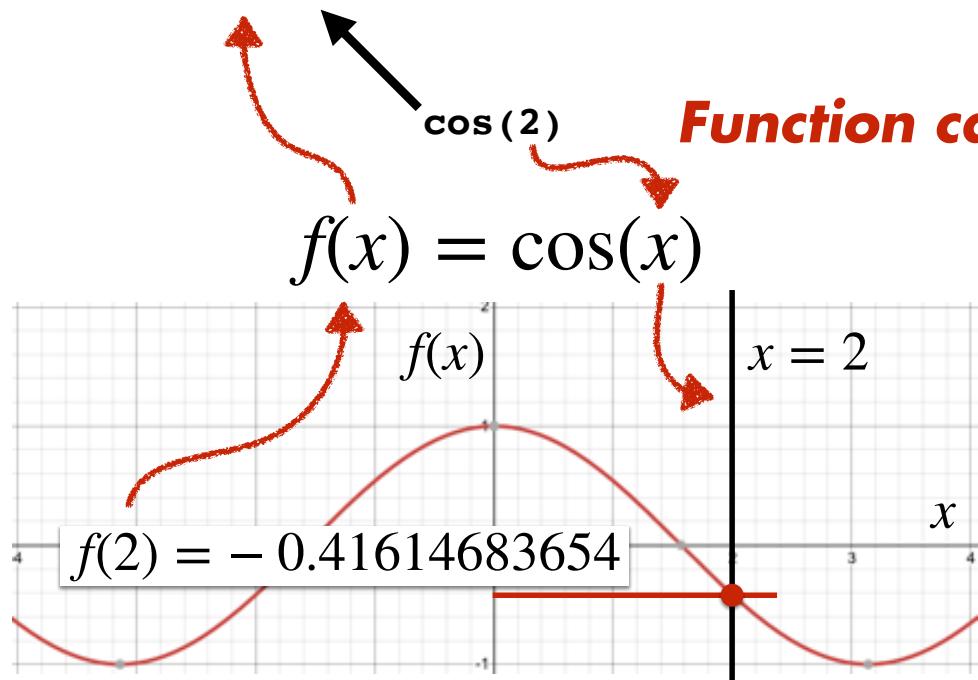
$$f(x) = \cos(x)$$

$$f(x) = x^y$$

What is a function ?

Function returns a value

-0.41614683654



Function called with input argument

`pow(myNumber, myOtherNumber)`

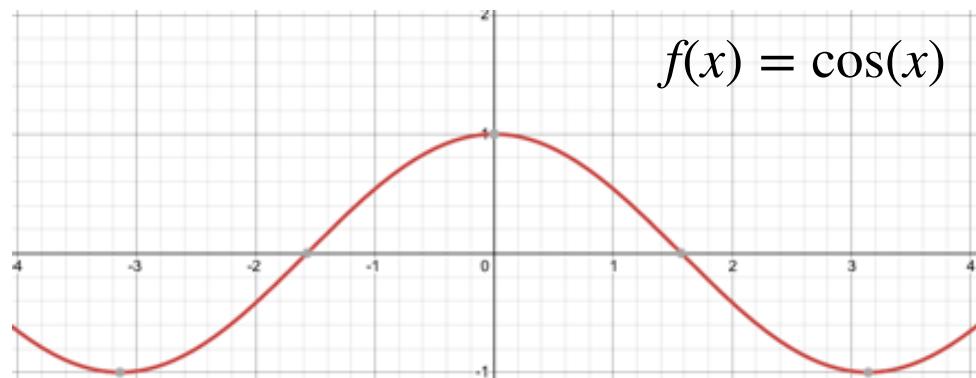
$$f(x) = x^y$$

Function evaluated for input argument

What is a function ?

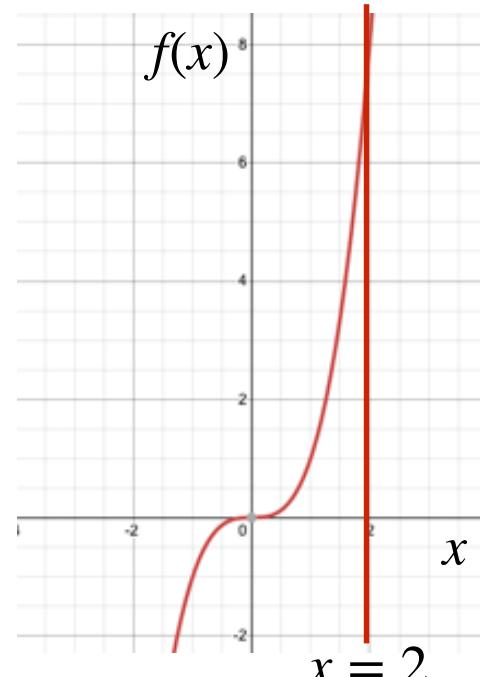
Intuitively, similar to a mathematical function

`cos (myNumber)`



$$f(x) = \cos(x)$$

`pow(2, 3)`

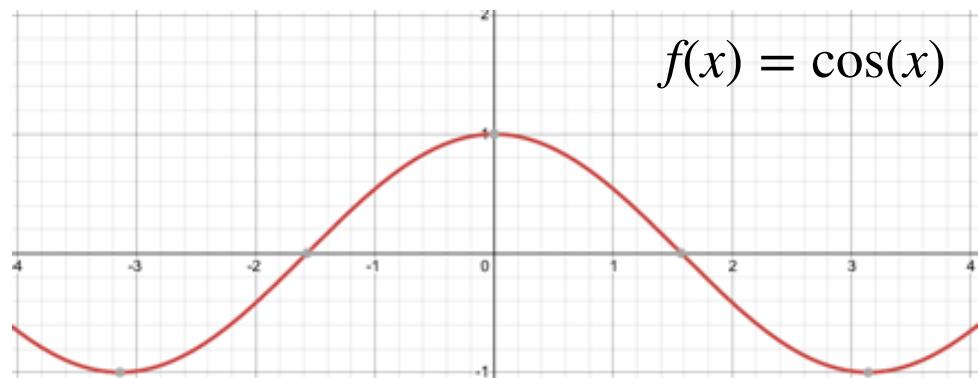


$$f(x) = 2^3$$

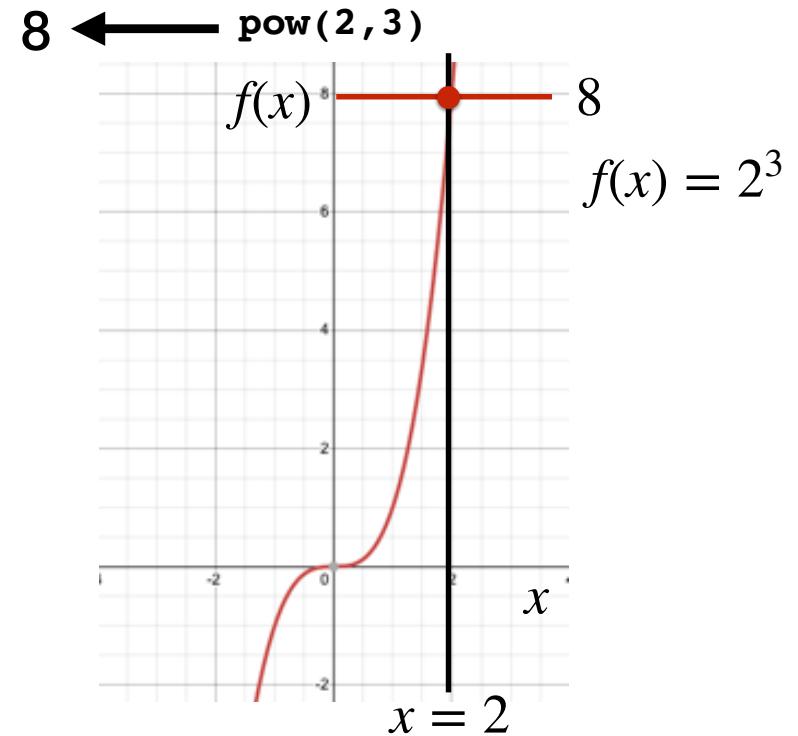
What is a function ?

Intuitively, similar to a mathematical function

`cos (myNumber)`



$$f(x) = \cos(x)$$



8 ← `pow(2, 3)`

$$f(x)$$

$$f(x) = 2^3$$

$$x = 2$$

What is a function ?

Intuitively, similar to a mathematical function

Function calls

`cos(myNumber)`

Mathematically

$$f(x) = \cos(x)$$

`pow(myNumber, myOtherNumber)`

$$f(x) = x^y$$

`myFunction(myNumber)`

$$f(x) = x^2 + 3$$

Make our own functions

What is a function ?

C++ functions are not necessarily mathematical functions

Function calls

`cos (myNumber)`

`pow(myNumber, myOtherNumber)`

`myFunction(myNumber)`

`printThisNumber(numberY)`

Mathematically

$$f(x) = \cos(x)$$

$$f(x) = x^y$$

$$f(x) = x^2 + 3$$

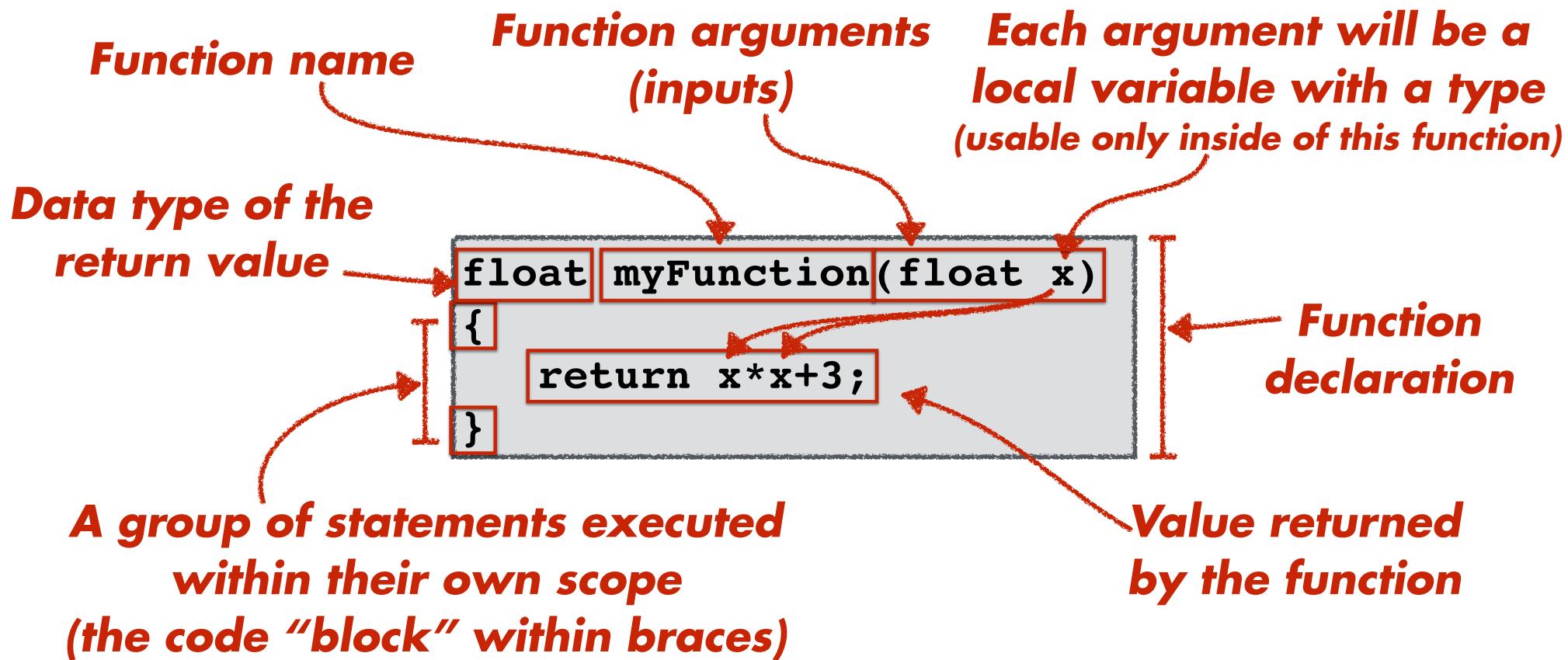
```
float myFunction(float x) {  
    return x*x+3;  
}
```

```
void printThisNumber(float y) {  
    std::cout << y;  
}
```

What is a C++ function ?

```
float myFunction(float x)
{
    return x*x+3;
}
```

What is a C++ function ?



What is a C++ function ?

A defined function can be invoked (or called) from another part of the program

Function call

```
int main() {  
    float numberY = myFunction(2.0);  
}
```

Function input parameter

```
float myFunction(float x)  
{  
    return x*x+3;  
}
```

What is a C++ function ?

A defined function can be invoked (or called) from another part of the program

Function call

```
int main() {  
    float numberY = myFunction(2.0);  
}
```

Function input parameter

Program execution passed to function

```
float myFunction(float x)  
{  
    return x*x+3;  
}
```

Local variable
 $x=2.0$

What is a C++ function ?

A defined function can be invoked (or called) from another part of the program

Function call

```
int main() {  
    float numberY = myFunction(2.0);  
}
```

Assignment
numberY=7.0

**Return value and program
execution passed back to
calling function**

Function input parameter

**Program execution
passed to function**

```
float myFunction(float x)  
{  
    return x*x+3;  
}
```

**Local
variable**
x=2.0

Statements execute

squaredPlusThree.cpp (Version 00)

```
float myFunction(float x) {
    return x*x+3;
}

int main() {
    float numberY = myFunction(2.0);
}
```

[No errors]

[No output]

squaredPlusThree.cpp (Version 01)

```
#include<iostream>

void printThisNumber(float y) {
    std::cout << y;
}

float myFunction(float x) {
    return x*x+3;
}

int main() {
    float numberY = myFunction(2.0);
    printThisNumber(numberY);
}
```

[No errors]

Let's write our calculator operations as functions

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

**Functions can take
multiple input arguments
(separated by commas)**

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

Subtraction

```
float subtractTwoNumbers(float operand1, float operand2) {  
    float difference = operand1 - operand2;  
    return difference;  
}
```

**Variables can be declared
within a function's scope**

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

Subtraction

```
float subtractTwoNumbers(float operand1, float operand2) {  
    float difference = operand1 - operand2;  
    return difference;  
}
```

Multiplication Will not work properly

```
void multiplyTwoNumbers(float operand1, float operand2, float product) {  
    product = operand1 * operand2;  
}
```



Function arguments are “pass by value”

**Modifying the variable in the function will not
change its value outside the function**

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

Subtraction

```
float subtractTwoNumbers(float operand1, float operand2) {  
    float difference = operand1 - operand2;  
    return difference;  
}
```

Multiplication

Will work properly

```
void multiplyTwoNumbers(float operand1, float operand2, float &product) {  
    product = operand1 * operand2;  
}
```



Adding & to front makes function argument “pass by reference”

**Modifying the variable in the function will
change its value outside the function**

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

Subtraction

```
float subtractTwoNumbers(float operand1, float operand2) {  
    float difference = operand1 - operand2;  
    return difference;  
}
```

Multiplication

```
void multiplyTwoNumbers(float operand1, float operand2, float &product) {  
    product = operand1 * operand2;  
}
```

*If no value is returned,
the function should use
void as the return type*

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

Subtraction

```
float subtractTwoNumbers(float operand1, float operand2) {  
    float difference = operand1 - operand2;  
    return difference;  
}
```

Multiplication

```
void multiplyTwoNumbers(float operand1, float operand2, float &product) {  
    product = operand1 * operand2;  
}
```

Division

```
bool divideTwoNumbers(float operand1, float operand2, float &quotient) {  
    quotient = operand1 / operand2;  
    return false;
```

Return false if no errors occurred

Addition

```
float addTwoNumbers(float operand1, float operand2) {  
    return operand1 + operand2;  
}
```

Subtraction

```
float subtractTwoNumbers(float operand1, float operand2) {  
    float difference = operand1 - operand2;  
    return difference;  
}
```

Multiplication

```
void multiplyTwoNumbers(float operand1, float operand2, float &product) {  
    product = operand1 * operand2;  
}
```

Division

```
bool divideTwoNumbers(float operand1, float operand2, float &quotient) {  
    quotient = operand1 / operand2;  
    return false;
```

calculator.cpp (Version 32)

```
#include <iostream>

/* Let's write a calculator program for real numbers with variables
   that takes numbers from user input (no more magic numbers!) */

float addTwoNumbers(float operand1, float operand2) {
    return operand1 + operand2;
}

float subtractTwoNumbers(float operand1, float operand2) {
    float difference = operand1 - operand2;
    return difference;
}

void multiplyTwoNumbers(float operand1, float operand2, float &product) {
    product = operand1 * operand2;
}

bool divideTwoNumbers(float operand1, float operand2, float &quotient) {
    quotient = operand1 / operand2;
    return false;
}

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
```

calculator.cpp (Version 32)

```
#include <iostream>

/* Let's write a calculator program for real numbers with variables
   that takes numbers from user input using functions for modularity */

float addTwoNumbers(float operand1, float operand2) {
    return operand1 + operand2;
}

float subtractTwoNumbers(float operand1, float operand2) {
    float difference = operand1 - operand2;
    return difference;
}

void multiplyTwoNumbers(float operand1, float operand2, float &product) {
    product = operand1 * operand2;
}

bool divideTwoNumbers(float operand1, float operand2, float &quotient) {
    quotient = operand1 / operand2;
    return false;
}

int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
```

Function declarations

calculator.cpp (Version 32)

```
float addTwoNumbers(float operand1, float operand2) {
    return operand1 + operand2;
}

float subtractTwoNumbers(float operand1, float operand2) {
    float difference = operand1 - operand2;
    return difference;
}

void multiplyTwoNumbers(float operand1, float operand2, float &product) {
    product = operand1 * operand2;
}

bool divideTwoNumbers(float operand1, float operand2, float &quotient) {
    quotient = operand1 / operand2;
    return false;
```

Function declarations

```
int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    // Ask the user for our second operand and assign it to "myOtherNumber"
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    char additionCharacter = '+'; // Character, for plus
    char subtractionCharacter = '-'; // Character, for minus
```

calculator.cpp (Version 32)

```
float addTwoNumbers(float operand1, float operand2) {
    return operand1 + operand2;
}

float subtractTwoNumbers(float operand1, float operand2) {
    float difference = operand1 - operand2;
    return difference;
}

void multiplyTwoNumbers(float operand1, float operand2, float &product) {
    product = operand1 * operand2;
}

bool divideTwoNumbers(float operand1, float operand2, float &quotient) {
    quotient = operand1 / operand2;
    return false;

// Perform all operations and output result to screen
std::cout << myNumber << additionCharacter << myOtherNumber << "= "
    << addTwoNumbers(myNumber,myOtherNumber) << "\n";
std::cout << myNumber << subtractionCharacter << myOtherNumber << "= "
    << subtractTwoNumbers(myNumber,myOtherNumber) << "\n";
float productNumber; // local variable only usable in main function
multiplyTwoNumbers(myNumber,myOtherNumber,productNumber); // function call
std::cout << myNumber << multiplicationCharacter << myOtherNumber << "= "
    << productNumber << "\n";
std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
    << myNumber / myOtherNumber << "\n";
float quotient; // NOT the same variable as quotient in function definition
divideTwoNumbers(myNumber,myOtherNumber,quotient); // function call
std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
    << quotient << "\n";
}
```

addTwoNumbers()

subtractTwoNumbers()

multiplyTwoNumbers()

divideTwoNumbers()

main()

**Function
calls**

calculator.cpp (Version 32)

```
float addTwoNumbers(float operand1, float operand2) {
    return operand1 + operand2;
}
float subtractTwoNumbers(float operand1, float operand2) {
    float difference = operand1 - operand2;
    return difference;
}
void multiplyTwoNumbers(float operand1, float operand2, float &product) {
    product = operand1 * operand2;
}
bool divideTwoNumbers(float operand1, float operand2, float &quotient) {
    quotient = operand1 / operand2;
    return false;
```

```
// Perform all operations and output result to screen
std::cout << myNumber << additionCharacter << myOtherNumber << " = "
    << addTwoNumbers(myNumber,myOtherNumber) << "\n";
std::cout << myNumber << subtractionCharacter << myOtherNumber << " = "
    << subtractTwoNumbers(myNumber,myOtherNumber) << "\n";
float productNumber; // local variable only usable in main function
multiplyTwoNumbers(myNumber,myOtherNumber,productNumber); // function call
std::cout << myNumber << multiplicationCharacter << myOtherNumber << " = "
    << productNumber << "\n";
std::cout << myNumber << divisionCharacter << myOtherNumber << " = "
    << myNumber / myOtherNumber << "\n";
float quotient; // NOT the same variable as quotient in function definition
divideTwoNumbers(myNumber,myOtherNumber,quotient); // function call
std::cout << myNumber << divisionCharacter << myOtherNumber << " = "
    << quotient << "\n";
```

addTwoNumbers()

subtractTwoNumbers()

multiplyTwoNumbers()

divideTwoNumbers()

main()

```
Please type a number and press enter: 22
Please type another number and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```

Great!

calculator.cpp (Version 32)

**Functions available
in this source file**

Functions

```
addTwoNumbers()  
subtractTwoNumbers()  
multiplyTwoNumbers()  
divideTwoNumbers()
```

main()

**Function scope for the
code snippet below**

```
// Perform all operations and output result to screen
std::cout << myNumber << additionCharacter << myOtherNumber << "= "
    << addTwoNumbers(myNumber,myOtherNumber) << "\n";
std::cout << myNumber << subtractionCharacter << myOtherNumber << "= "
    << subtractTwoNumbers(myNumber,myOtherNumber) << "\n";
float productNumber; // local variable only usable in main function
multiplyTwoNumbers(myNumber,myOtherNumber,productNumber); // function call
std::cout << myNumber << multiplicationCharacter << myOtherNumber << "= "
    << productNumber << "\n";
std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
    << myNumber / myOtherNumber << "\n";
float quotient; // NOT the same variable as quotient in function definition
divideTwoNumbers(myNumber,myOtherNumber,quotient); // function call
std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
    << quotient << "\n";
}
```

calculator.cpp (Version 32)

```
int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    char additionCharacter = '+'; // Character, for plus
    char subtractionCharacter = '-'; // Character, for minus
    char multiplicationCharacter = '*'; // Character, for times
    char divisionCharacter = '/'; // Character, for division

    // Perform all operations and output result to screen
    std::cout << myNumber << additionCharacter << myOtherNumber << "= "
        << addTwoNumbers(myNumber,myOtherNumber) << "\n";
    std::cout << myNumber << subtractionCharacter << myOtherNumber << "= "
        << subtractTwoNumbers(myNumber,myOtherNumber) << "\n";
    float productNumber; // local variable only usable in main function
    multiplyTwoNumbers(myNumber,myOtherNumber,productNumber); // function call
    std::cout << myNumber << multiplicationCharacter << myOtherNumber << "= "
        << productNumber << "\n";
    std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
        << myNumber / myOtherNumber << "\n";
    float quotient; // NOT the same variable as quotient in function definition
    divideTwoNumbers(myNumber,myOtherNumber,quotient); // function call
    std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
        << quotient << "\n";
    return 0; ← Return false if no errors occurred
}
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
```

calculator.cpp (Version 32)

```
int main()
{
    // Ask the user to give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    char additionCharacter = '+'; // Character, for plus
    char subtractionCharacter = '-'; // Character, for minus
    char multiplicationCharacter = '*'; // Character, for times
    char divisionCharacter = '/';

    // Perform all operations and output them
    std::cout << myNumber << additionCharacter << myOtherNumber << "="
        << addTwoNumbers(myNumber,myOtherNumber) << "\n";
    std::cout << myNumber << subtractionCharacter << myOtherNumber << "="
        << subtractTwoNumbers(myNumber,myOtherNumber) << "\n";
    float productNumber; // local variable only usable in main function
    multiplyTwoNumbers(myNumber,myOtherNumber,productNumber); // function call
    std::cout << myNumber << multiplicationCharacter << myOtherNumber << "="
        << productNumber << "\n";
    std::cout << myNumber << divisionCharacter << myOtherNumber << "="
        << myNumber / myOtherNumber << "\n";
    float quotient; // NOT the same variable as quotient in function definition
    divideTwoNumbers(myNumber,myOtherNumber,quotient); // function call
    std::cout << myNumber << divisionCharacter << myOtherNumber << "="
        << quotient << "\n";
    return 0;
}
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
main()
```

Still very disorganized code :(

calculator.cpp (Version 32)

```
int main()
{
    // Ask the user to Give us two numbers for our operands
    float myNumber, myOtherNumber;
    std::cout << "Please type a number and press enter: ";
    std::cin >> myNumber; // Wait for user to enter a first operand
    std::cout << "Please type another number and press enter: "; // Second operand
    std::cin >> myOtherNumber;

    char additionCharacter = '+'; // Character, for plus
    char subtractionCharacter = '-'; // Character, for minus
    char multiplicationCharacter = '*'; // Character, for times
    char divisionCharacter = '/'; // Character, for division

    // Perform all operations and Output results to screen
    std::cout << myNumber << additionCharacter << myOtherNumber << "= "
        << addTwoNumbers(myNumber,myOtherNumber) << "\n";
    std::cout << myNumber << subtractionCharacter << myOtherNumber << "= "
        << subtractTwoNumbers(myNumber,myOtherNumber) << "\n";
    float productNumber; // local variable only usable in main function
    multiplyTwoNumbers(myNumber,myOtherNumber,productNumber); // function call
    std::cout << myNumber << multiplicationCharacter << myOtherNumber << "= "
        << productNumber << "\n";
    std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
        << myNumber / myOtherNumber << "\n";
    float quotient; // NOT the same variable as quotient in function definition
    divideTwoNumbers(myNumber,myOtherNumber,quotient); // function call
    std::cout << myNumber << divisionCharacter << myOtherNumber << "= "
        << quotient << "\n";
    return 0;
}
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
main()
```

What is this code doing ?

calculator.cpp (Version 41)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
    // Let's declare our variables
    float myNumber, myOtherNumber; // Calculation operands
    float sumNumber, differenceNumber, productNumber, quotientNumber;

    // Ask the user for the first operand
    getNumber(myNumber); ←

    // Ask the user for the second operand
    getNumber(myOtherNumber); ←

    // Perform all operations and store results in variables
    performOperations(myNumber,myOtherNumber,sumNumber,differenceNumber,
                      productNumber,quotientNumber);

    // Output operation results to screen
    outputResults(myNumber,myOtherNumber,sumNumber,differenceNumber,
                  productNumber,quotientNumber);

    return 0;
}
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
main()
```

Give us two numbers for our operands

Perform all operations

Output results to screen

Only 4 function calls!

calculator.cpp (Version 41)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
    // Let's declare our variables
    float myNumber, myOtherNumber; // Calculation operands
    float sumNumber, differenceNumber, productNumber, quotientNumber;

    // Ask the user for the first operand
    getNumber(myNumber); ←

    // Ask the user for the second operand
    getNumber(myOtherNumber); ←

    // Perform all operations and store results in variables
    performOperations(myNumber,myOtherNumber,sumNumber,differenceNumber,
                      productNumber,quotientNumber);

    // Output operation results to screen
    outputResults(myNumber,myOtherNumber,sumNumber,differenceNumber,
                  productNumber,quotientNumber);

    return 0;
}
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
main()
```

Give us two numbers for our operands

```
bool getNumber(float &number) {
```

Perform all operations

```
bool performOperations(float operand1, float operand2, float &sum,
                      float &difference, float &product, float &quotient) {
```

Output results to screen

```
bool outputResults(float operand1, float operand2, float sum,
                   float difference, float product, float quotient) {
```

calculator.cpp (Version 41)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
    // Let's declare our variables
    float myNumber, myOtherNumber; // Calculation operands
    float sumNumber, differenceNumber, productNumber, quotientNumber;

    // Ask the user for the first operand
    getNumber(myNumber);

    // Ask the user for the second operand
    getNumber(myOtherNumber);

    // Perform all operations and store results in variables
    performOperations(myNumber,myOtherNumber,sumNumber,differenceNumber,
        productNumber,quotientNumber);

    // Output operation results to screen
    outputResults(myNumber,myOtherNumber,sumNumber,differenceNumber,
        productNumber,quotientNumber);

    return 0;
}
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
getNumber()
performOperations()
outputResults()
main()
```

```
Please type a number and press enter: 22
Please type another number and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```

Still works!

Our calculator still needs more

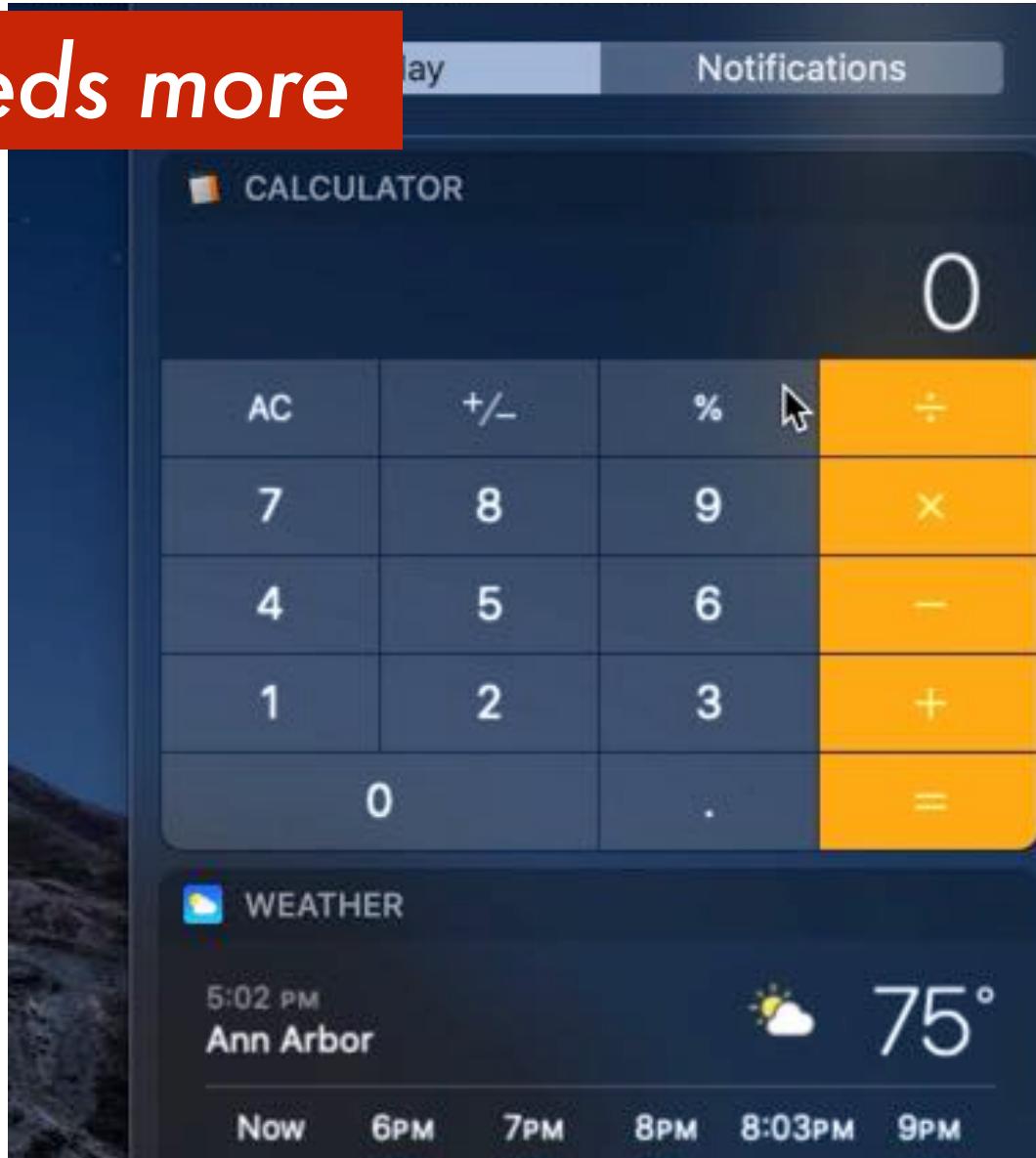
calculator (Version 41)

```
Please type a number and press enter: 22  
Please type another number and press enter: 7  
22+7= 29  
22-7= 15  
22*7= 154  
22/7= 3.14286
```

Select a single operator to perform?

Perform multiple operations in succession?

Can we divide by zero?



Done

hello
Hello World!
Chad is in Robotics 102"

calculator (Version 24)

```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 plus 7? 29
What is 22 minus 7 ? 15
What is 22 times 7 ? 154
What is 22 divided by 7 ? 3.14286
```

calculator (Version 41)

- Program Structure
- Compile/Execute
- Operators
- Data Types
- Variables
- User Input/Output
- Functions
- Branching
- Iterators
- Vectors
- Structs
- File Input/Output



Coming

wall_follower.cpp - Project 1

```
while (true) {
    LidarScan scan = readLidarScan(drv);

    if (min_idx == -1) {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = -1;
        float dist_to_wall = 100.0f;
        float dir_to_wall = 0.0f;
    }

    std::cout << "dist_to_wall: " << dist_to_wall << " dir_to_wall: " << dir_to_wall << std::endl;

    // Compute a vector that points towards the closest obstacle.
    Vector3D robot_to_wall_v;
    Vector3D forward_v = ...;

    // Create a vector that points up.
    Vector3D up_v = ...;

    // Get a vector that is perpendicular to the nearest obstacle.
    Vector3D forward_perp_v = ...;

    float vx = ...;
    float vy = ...;
    std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

    vx += ...;
    vy += ...;

    drive(vx, vy, 0);
}
```

