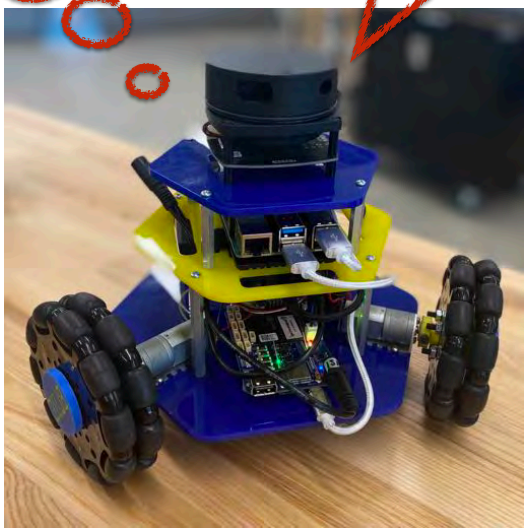
```
/* Robotics 102 - Fall 2021
   Introduction to AI and Programming

   C++ Operators and Variables */

shapeYCoordinate = sqrt(cos(x))*cos(300*x)
   +sqrt(abs(x))-0.7*(4-x*x)^0.01;
boundaryUpper = sqrt(6-x^2);
boundaryLower = -sqrt(6-x^2);
```

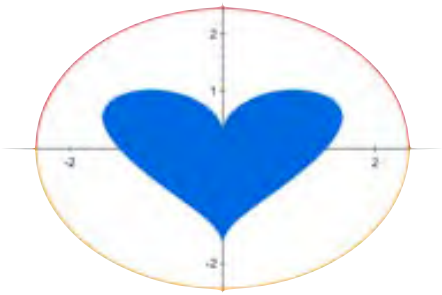


Open

Type

sqrt(cos(x))*cos(300x)+sqrt(abs(x))-0.7*(4-x*x)^0.01, sqrt(6-x^2), -sqrt(6-x^2) from -4.5 to 4.5

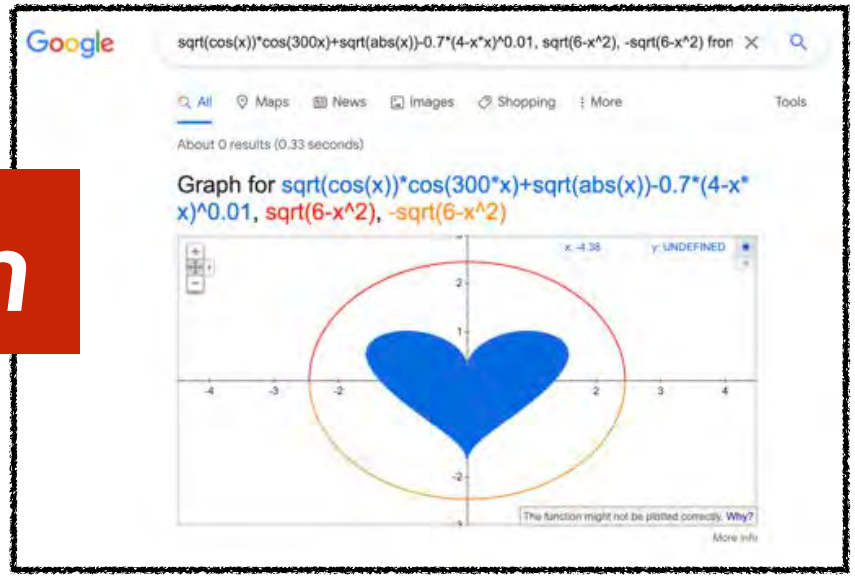
Your result ?



These are the same

(roughly)

Graph



Code

```
shapeYCoordinate = sqrt(cos(x))*cos(300*x)
   +sqrt(abs(x))-0.7*(4-x*x)^0.01;
boundaryUpper = sqrt(6-x^2);
boundaryLower = -sqrt(6-x^2);
```

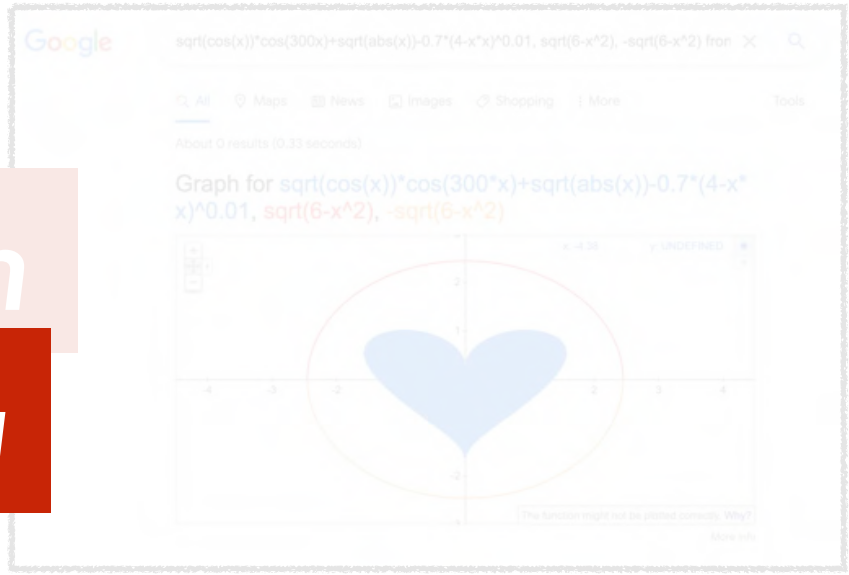
Algebra

$$\sqrt{\cos(x)} * \cos(300 * x) + \sqrt{|x|} - 0.7 * (4 - x * x)^{0.01}$$

These are the same

Graph

Code "is" Algebra




```
shapeYCoordinate = sqrt(cos(x))*cos(300*x)
   +sqrt(abs(x))-0.7*(4-x*x)^0.01;
boundaryUpper = sqrt(6-x^2);
boundaryLower = -sqrt(6-x^2);
```

Variables (x)

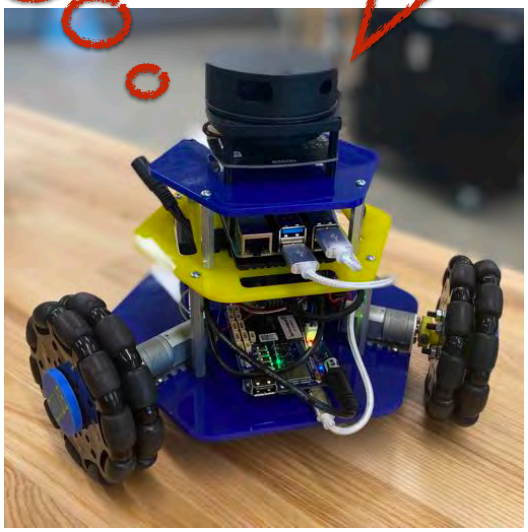
Arithmetic Operations (+ - * /)

$$\sqrt{\cos(x)} * \cos(300 * x) + \sqrt{|x|} - 0.7 * (4 - x * x)^{0.01}$$

```
/* Robotics 102 - Fall 2021
   Introduction to AI and Programming
```

Arithmetic and "Algebra" in C++ `*/`

```
shapeYCoordinate = sqrt(cos(x))*cos(300*x)
   +sqrt(abs(x))-0.7*(4-x*x)^0.01;
boundaryUpper = sqrt(6-x^2);
boundaryLower = -sqrt(6-x^2);
```

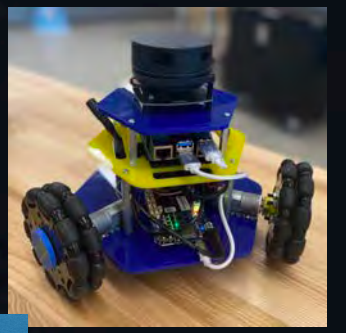


- [ ] Program Structure
- [ ] Compile/Execute
- [ ] Operators
- [ ] Data Types
- [ ] Variables
- [ ] User Input/Output
- [ ] Functions
- [ ] Branching
- [ ] Iterators
- [ ] Vectors
- [ ] Structs
- [ ] File Input/Output

wall_follower.cpp - Project 1

```
while [redacted] {
    LidarScan scan = readLidarScan(drv);

    if ([redacted]) {
        // Get the index of the shortest ray, and save
        // the angle of the ray.
        int min_idx = [redacted]
        float [redacted]
        float [redacted]

        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;
        [redacted]

        // Create a vector that points up.
        [redacted]

        // Get a vector that is perpendicular to the nearest obstacle.
        Vector3D forward_v = [redacted]

        float vx = [redacted]
        float vy = [redacted]
        std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

        [redacted]

        vx += [redacted]
        vy += [redacted]

        [redacted]

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

Coming



Done

hello.cpp - Last Lecture

```cpp
#include <iostream>
/* Hello World - A first C++ Program
   Copyright 2021 Odest Chadwicke Jenkins at the University of Michigan
   Licensed under Michigan Honor License in the LICENSE file and
   available at to view at https://autorob.org/MichiganHonorLicense.txt
*/

int main()
{
   std::cout << "Hello World" << "\n";  // A single-line comment
   std::cout << "Chad is in Robotics 102";  // "\n" creates a new line
}
```

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

Coming

wall_follower.cpp - Project 1

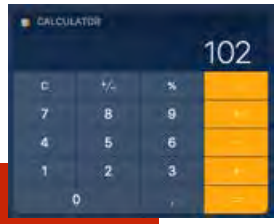
```cpp
while [redacted] {
    LidarScan scan = readLidarScan(drv);

    if [redacted] {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = [redacted]
        float [redacted]
        float [redacted]

        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;
        [redacted]

        // Create a vector that points up.
        [redacted]

        // Get a vector that is perpendicular to the nearest obstacle.
        Vector3D forward_v = [redacted]

        float vx = [redacted]
        float vy = [redacted]
        std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

        [redacted]

        vx += [redacted]
        vy += [redacted]

        [redacted]

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

Done

hello.cpp - Last Lecture

```cpp
#include <iostream>
/* Hello World - A first C++ Program
   Copyright 2021 Odest Chadwicke Jenkins at the University of Michigan
   Licensed under Michigan Honor License in the LICENSE file and
   available at to view at https://autorob.org/MichiganHonorLicense.txt
*/

int main()
{
   std::cout << "Hello World" << "\n";  // A single-line comment
   std::cout << "Chad is in Robotics 102";  // "\n" creates a new line
}
```

Now



pocketcalc.cpp - Project 0

```cpp
/*
   pocketcalc - Pocket Calculator for interactive use from a terminal interface

   An interactive infix calculator program for real numbers with variables
   that takes numbers from user input, uses functions for modularity,
   performs calculations with infinitely many consecutive operations,
   stores the entire mathematical expression in vector of structs,
   outputs this mathematical expression in infix notation as a string,
   writes the result to a file, and allows user to undo last operat

#include <iostream>  // enable C++ Input-Output streams
#include <vector>  // this enables the program to use C++
#include <string>  // this enables the program to use C++
#include <sstream> // include to enable C++ string stream
#include <fstream> // include to enable C++ streams readi

// Define new data type "operationEquation" to represent all mean
struct operationEquation {
   float operand1;
   char operation;
   float operand2;
   float result;
};

// Function defined to add two numbers and return their sum
float addTwoNumbers(float operand1, float operand2) {
   // Note: function arguments are local variables usable only in this function
   return operand1 + operand2;  // function will return a floating point number
}
```

Coming

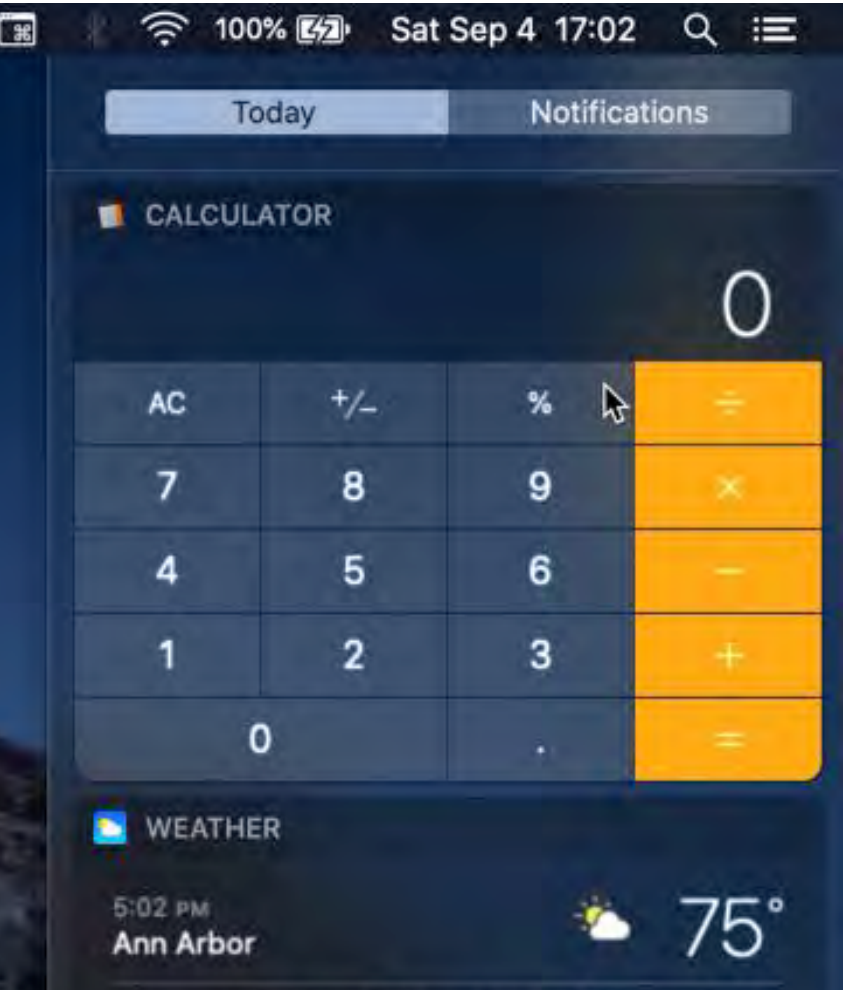
wall_follower.cpp - Project 1

```cpp
while [redacted] {
   LidarScan scan = readLidarScan(drv);

   if [redacted] {
      // Get the index of the shortest ray, and save that distance and
      // the angle of the ray.
      int min_idx = [redacted]
      float [redacted]
      float [redacted]

      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;
      [redacted]

      // Create a vector that points up.
      [redacted]

      // Get a vector that is perpendicular to the nearest obstacle.
      Vector3D forward_v = [redacted]

      float vx = [redacted]
      float vy = [redacted]
      std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

      [redacted]

      vx += [redacted]
      vy += [redacted]

      [redacted]

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

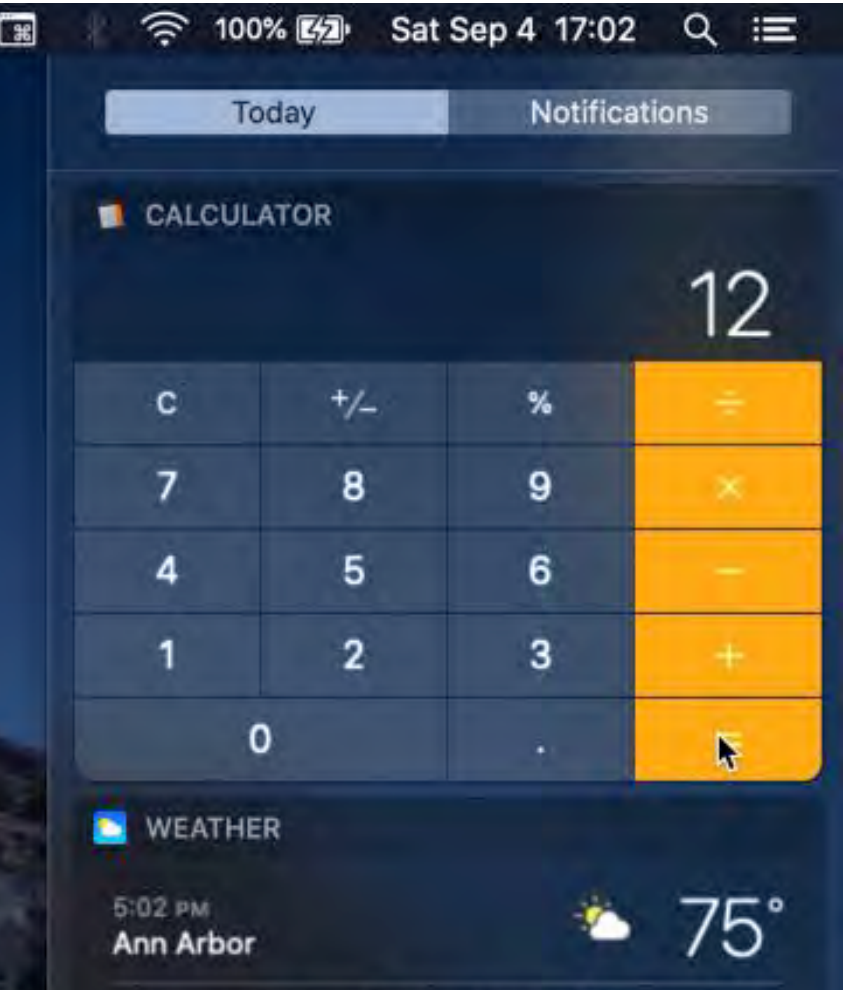
C.alculation
R.uns
E.verywhere
A.round
M.e

Let's walk through a calculation example

*3 * 4 = 12*



Let's walk through a calculation example



*3 * 4 = 12*

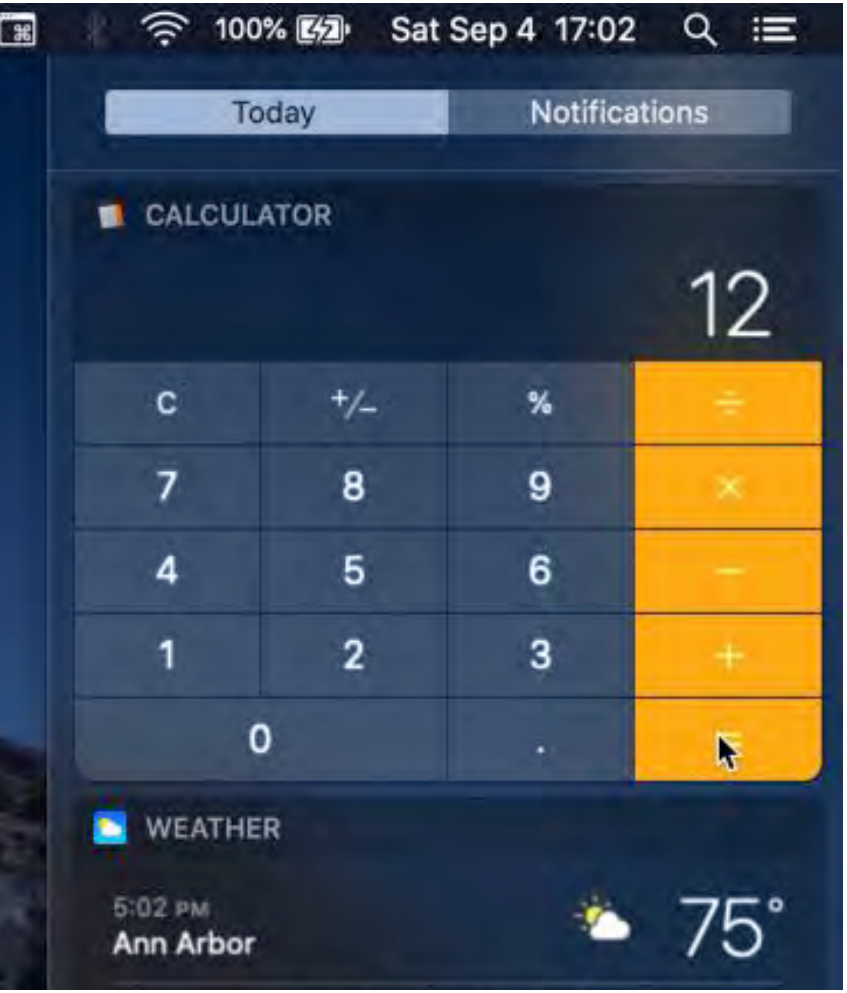
Infix notation

Operand	*Operator*	*Operand*	=	*Result*
3	*	*4*	=	*12*

Prefix notation: ** 3 4 = 12*

Postfix notation: *3 4 * = 12*

Let's walk through a calculation example



Infix notation

Operand	*Operator*	*Operand*	=	*Result*
3	*	*4*	=	*12*

Operators perform basic arithmetic operations

Addition	**+**
Subtraction	**-**
Multiplication	*****
Division	**/**

Let's do some arithmetic in C++

calculator.cpp (Version 00)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   std::cout << "What is 100 plus 2?" << "\n";
}
```

Quick Tangent: Coding Setup

filename.cpp

```
Source code
```

```
Compiler Messages
```

```
Program Output
```

calculator.cpp (Version 00)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   std::cout << "What is 100 plus 2?" << "\n";
}
```

Compile

```
[No errors]
```

```
Program Output
```

calculator.cpp (Version 00)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   std::cout << "What is 100 plus 2?" << "\n";
}
```

Compile

```
[No errors]
```

Execute

```
What is 100 plus 2?
```

calculator.cpp (Version 01)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   std::cout << "What is 100 plus 2?" << "\n";
   std::cout <<
}
```

What should go here to produce this program output ?

```
What is 100 plus 2?
102
```

calculator.cpp (Version 01)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   std::cout << "What is 100 plus 2?" << “\n";
   std::cout << 100 + 2 << "\n";  // + is a plus operator to add two numbers
}
```

```
[No errors]
```

```
What is 100 plus 2?
102
```

calculator.cpp (Version 02)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   std::cout << "What is 100 plus 2?" << 100 + 2 << “\n";
}
```

We only need one line

calculator.cpp (Version 03)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform addition and output result to screen
   std::cout << "What is 100 plus 2?" << 100 + 2 << “\n";
}
```

Add informative comment about what this code does

calculator.cpp (Version 03)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform addition and output result to screen
   std::cout << "What is 100 plus 2?" << 100 + 2 << “\n";
}
```

```
[No errors]
```

calculator.cpp (Version 03)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform addition and output result to screen
   std::cout << "What is 100 plus 2?" << 100 + 2 << “\n";
}
```

```
[No errors]
```

```
What is 100 plus 2?
102
```

Program output still correct

calculator.cpp (Version 03)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform addition and output result to screen
   std::cout << "What is 100 plus 2?" << 100 + 2 << “\n";
}
```

```
[No errors]
```

Let’s assume our code is error free for now

```
What is 100 plus 2?
102
```

calculator.cpp (Version 04)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform all arithmetic operations and output results to screen
   std::cout << "What is 100 plus 2? " << 100 + 2 << "\n";
   std::cout << "What is 100 minus 2? "
   std::cout << "What is 100 times 2? "
   std::cout << "What is 100 divided by 2? "
}
```

What should go on these lines ?

```
What is 100 plus 2? 102
```

What should be the output ?

calculator.cpp (Version 04)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform all arithmetic operations and output results to screen
   std::cout << "What is 100 plus 2? " << 100 + 2 << "\n";
   std::cout << "What is 100 minus 2? " << 100 - 2 << "\n";
   std::cout << "What is 100 times 2? " << 100 * 2 << "\n";
   std::cout << "What is 100 divided by 2? " << 100 / 2 << "\n";
}
```

```
What is 100 plus 2? 102
What is 100 minus 2? 98
What is 100 times 2? 200
What is 100 divided by 2? 50
```

calculator.cpp (Version 04)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform all arithmetic operations and output results to screen
   std::cout << "What is 100 plus 2? " << 100 + 2 << "\n";
   std::cout << "What is 100 minus 2? " << 100 - 2 << "\n";
   std::cout << "What is 100 times 2? " << 100 * 2 << "\n";
   std::cout << "What is 100 divided by 2? " << 100 / 2 << "\n";
}
```

We can operate on any numbers

Let's try 8 and 5

```
What is 100 plus 2? 102
What is 100 minus 2? 98
What is 100 times 2? 200
What is 100 divided by 2? 50
```

calculator.cpp (Version 05)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform all arithmetic operations and output results to screen
   std::cout << "What is 8 plus 5? " << 8 + 5 << "\n";
   std::cout << "What is 8 minus 5? " << 8 - 5 << "\n";
   std::cout << "What is 8 times 5? " << 8 * 5 << "\n";
   std::cout << "What is 8 divided by 5? " << 8 / 5 << “\n";
}
```

Something is not quite right

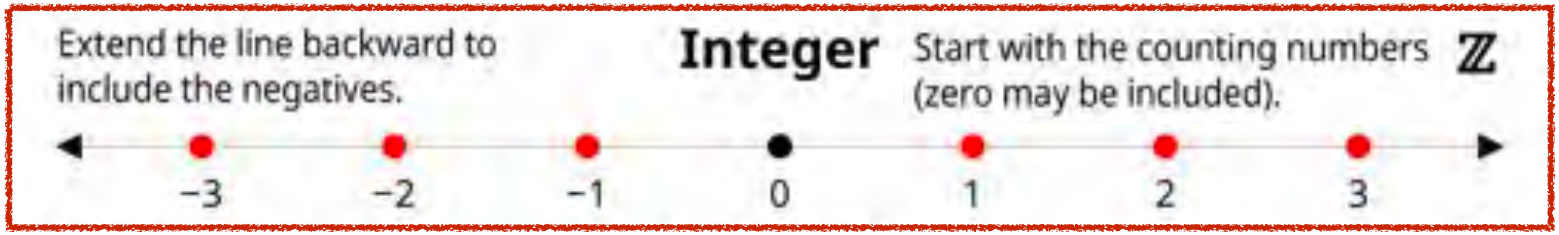
```
What is 8 plus 5? 13
What is 8 minus 5? 3
What is 8 times 5? 40
What is 8 divided by 5? 1
```

Something is not quite right

calculator.cpp (Version 05)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // Perform all arithmetic operations and output results to screen
   std::cout << "What is 8 plus 5? " << 8 + 5 << "\n";
   std::cout << "What is 8 minus 5? " << 8 - 5 << "\n";
   std::cout << "What is 8 times 5? " << 8 * 5 << "\n";
   std::cout << "What is 8 divided by 5? " << 8 / 5 << “\n";
   std::cout << "What is the remainder of 8 divided by 5? " << 8 % 5 << "\n";
}
```

```
What is 8 plus 5? 13
What is 8 minus 5? 3
What is 8 times 5? 40
What is 8 divided by 5? 1
What is the remainder of 8 divided by 5? 3
```

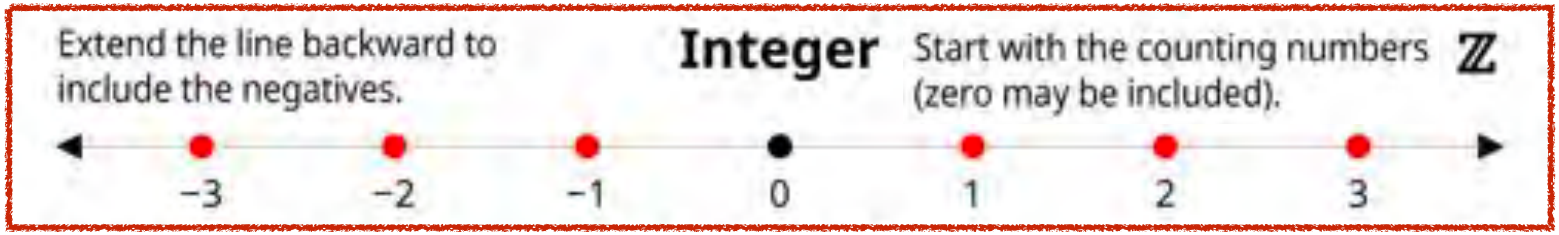


https://thinkzone.wlonk.com/Numbers/NumberSets.htm

Operators perform basic arithmetic operations

Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%

Computes remainder for Integer division



https://thinkzone.wlonk.com/Numbers/NumberSets.htm

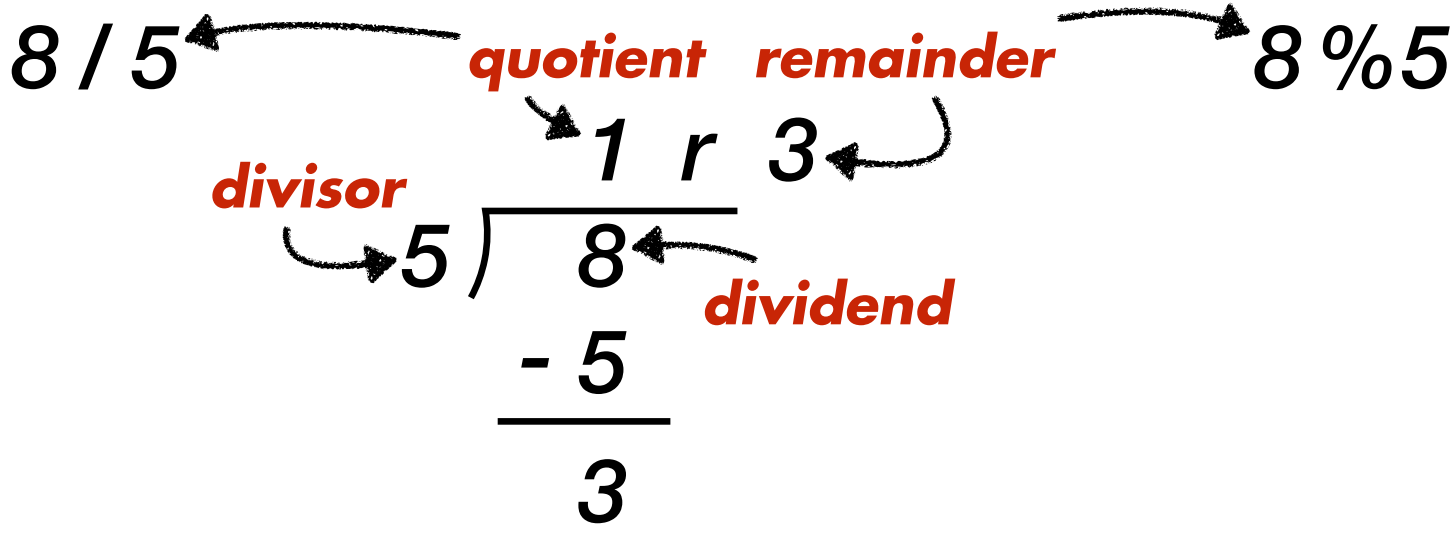
Operators perform basic arithmetic operations

Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%

Integer division

$$
\begin{array}{r}
1\ r\ 3 \\
5\overline{)\ 8} \\
-\ 5 \\
\hline
3
\end{array}
$$

Integer division

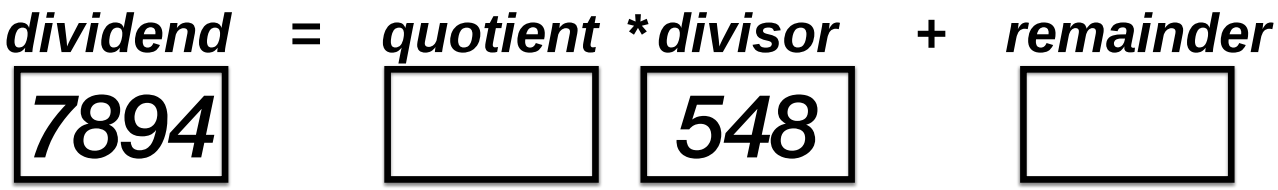


dividend	=	quotient	*	divisor	+	remainder
8		1		5		3

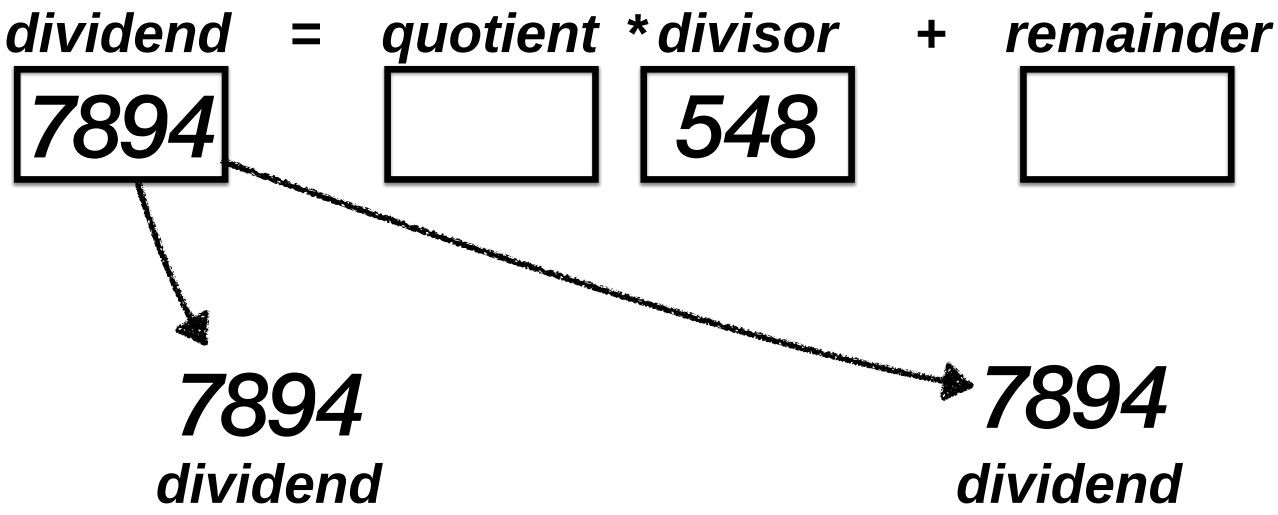
Integer division

dividend	=	*quotient*	*	*divisor*	+	*remainder*
8		1		5		3

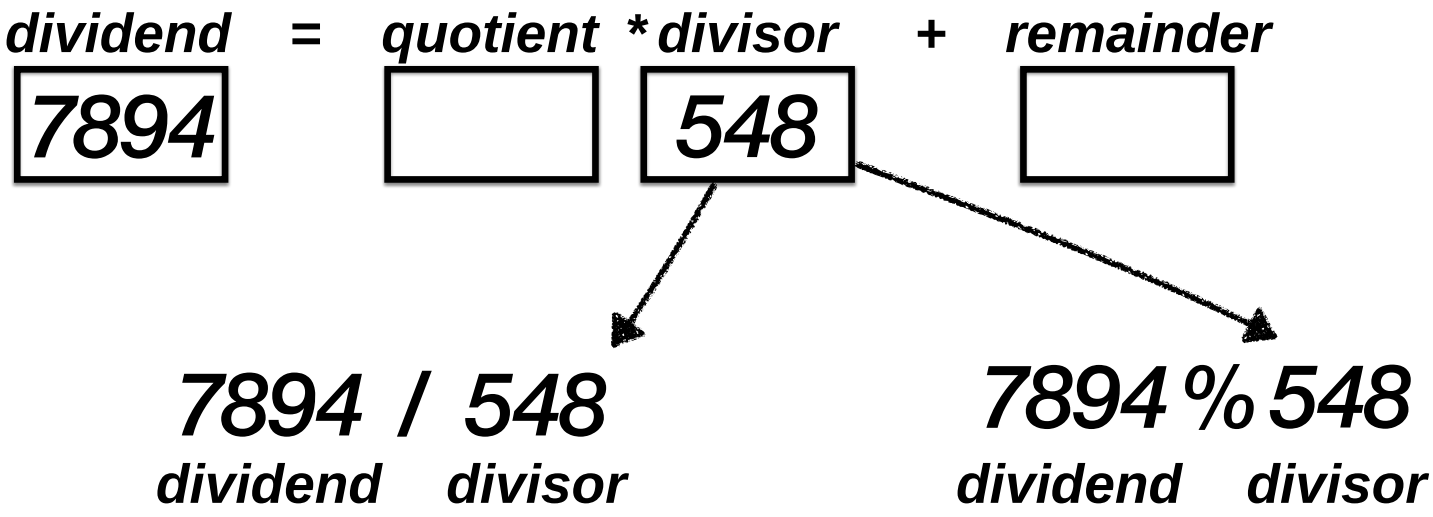
Integer division



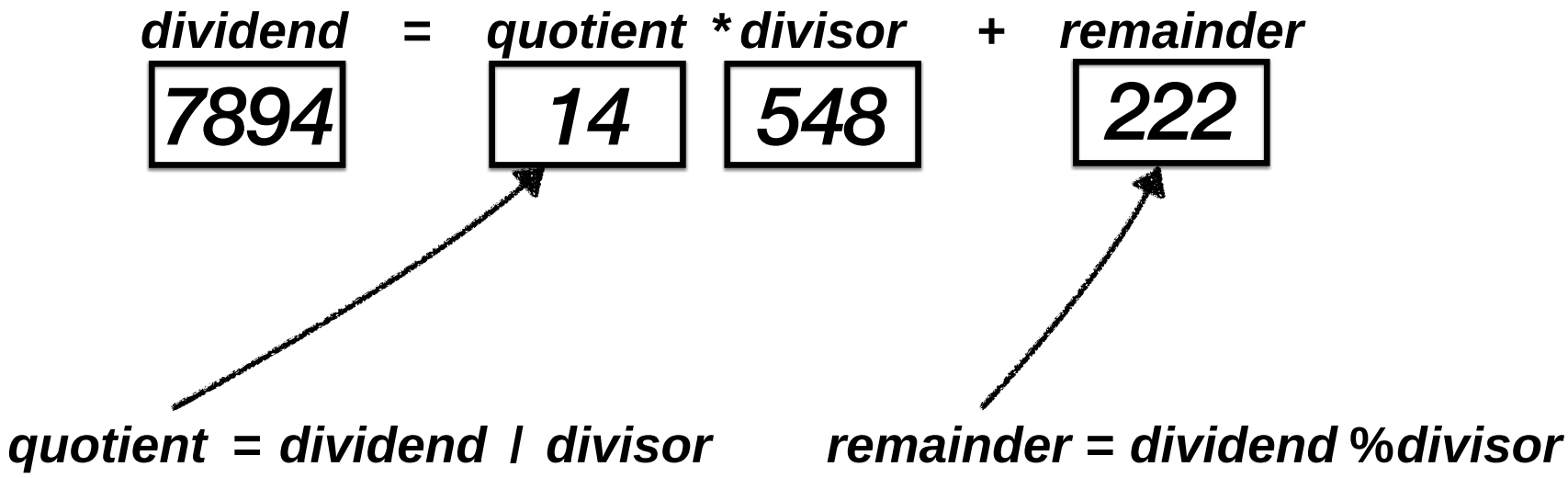
Integer division



Integer division



Integer division



Integer division with constants

14 = 7894 / 548

222 = 7894 % 548

Integer division with variables

dividend ***= 7894***

divisor ***= 548***

quotient ***= dividend / divisor***

remainder ***= dividend % divisor***

Integer division with variables

Let's turn these into C++ statements

dividend ***= 7894***

divisor ***= 548***

quotient ***= dividend / divisor***

remainder ***= dividend % divisor***

Integer division with variables

A variable is a container for a specified type of data

Container stored in computer memory

```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

A variable is a container for a specified type of data

Container stored in computer memory

3

3.141592

3.141592653589793

C++ basic data types:

`int`	*Integer number*
`float`	*Floating point number*
`double`	*Double precision floating point*
`char`	*Character*
`bool`	*Boolean*

'+'

Either true or false

Integer division with variables

A variable is a container for a specified type of data

C++ declaration of an integer variable named "dividend"


```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Variable names are call identifiers

Integer division with variables

A variable is a container for a specified type of data

C++ declaration of an integer variable named "dividend"

Assignment of a value To a variable


```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Integer division with variables

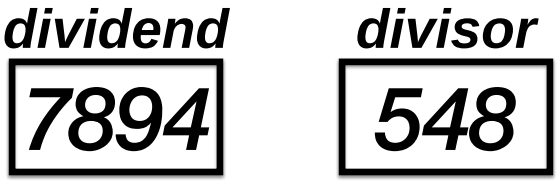
A variable is a container for a specified type of data

C++ declaration of an integer variable named "dividend"

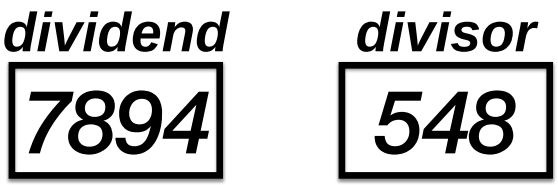
Assignment of a value To a variable

```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

A variable can be declared and assigned a value in one statement



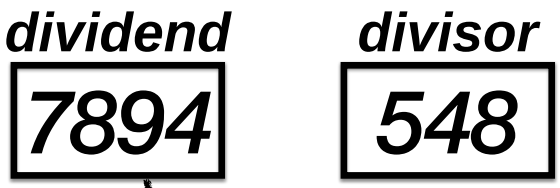
Integer division with variables




```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Arithmetic operations can be performed on values stored in variables

Integer division with variables



Retrieve value from variable

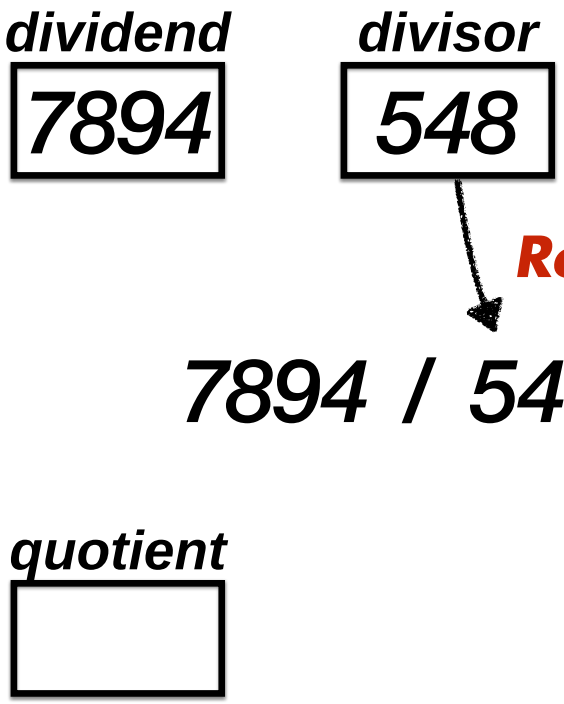
7894

quotient

```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Arithmetic operations can be performed on values stored in variables

Integer division with variables




```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Arithmetic operations can be performed on values stored in variables

Integer division with variables

dividend 7894

divisor 548

Perform operation

14 = 7894 / 548

quotient

```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Arithmetic operations can be performed on values stored in variables

Integer division with variables

dividend 7894

divisor 548

14 = 7894 / 548

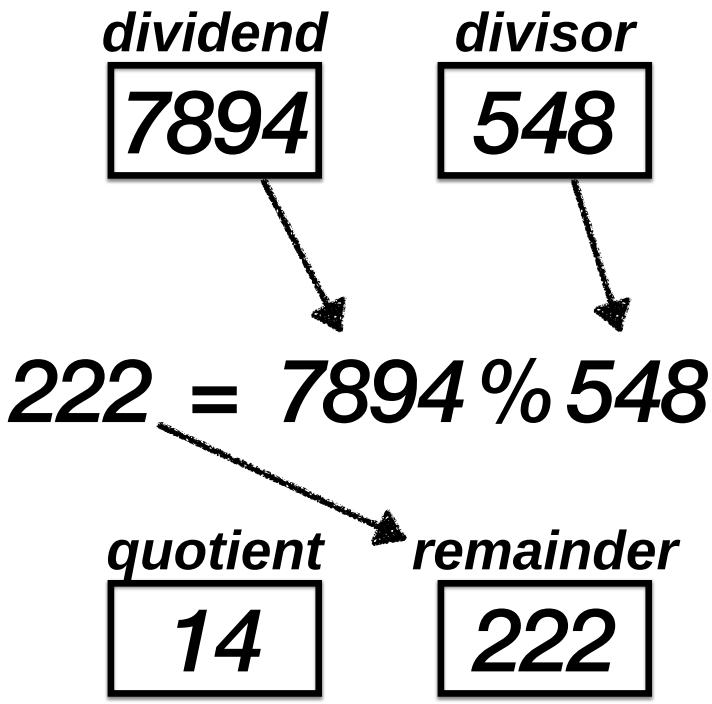
Store result to variable

quotient 14

```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Arithmetic operations can be performed on values stored in variables

Integer division with variables




```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

Arithmetic operations can be performed on values stored in variables

C++ reserved words cannot be used as variable names

and	double	not_eq	throw
and_eq	dynamic_cast	operator	true
asm	else	or	try
auto	enum	or_eq	typedef
bitand	explicit	private	typeid
bitor	extern	protected	typename
bool	false	public	union
break	float	register	unsigned
case	for	reinterpret-cast	using
catch	friend	return	virtual
char	goto	short	void
class	if	signed	volatile
compl	inline	sizeof	wchar_t
const	int	static	while
const-cast	long	static_cast	xor
continue	mutable	struct	xor_eq
default	namespace	switch	
delete	new	template	
do	not	this	

A C++ variable must be declared before its used

Assignment is NOT equality (a variable "gets" a value)

```
int dividend;
dividend = 7894;

int divisor = 548;

int quotient = dividend / divisor;
int remainder = dividend % divisor;
```

dividend	*divisor*	*quotient*	*remainder*
7894	548	14	222

Let's do division with C++ variables

calculator.cpp (Version 09)

```
#include <iostream>

/*   Let's write a calculator program  */

int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;

   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   // A variable can be output to the screen using its name (or identifier)
   std::cout << "What is myNumber? " << myNumber << "\n";

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is 7894 divided by 548? " << 7894 / 548 << "\n";
   std::cout << "What is the remainder of 7894 divided by 548? "
      << 7894 % 548 << "\n";
   std::cout << "Verify 7894 is equal to 14 times 548 plus 222: "
      << 14 * 548 + 222 - 7894 << "\n";
}
```

calculator.cpp (Version 09)

```
int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;

   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   // A variable can be output to the screen using its name (or identifier)
   std::cout << "What is myNumber? " << myNumber << "\n";

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is 7894 divided by 548? " << 7894 / 548 << "\n";
   std::cout << "What is the remainder of 7894 divided by 548? "
      << 7894 % 548 << "\n";
   std::cout << "Verify 7894 is equal to 14 times 548 plus 222: "
      << 14 * 548 + 222 - 7894 << "\n";
}
```

Variable declaration

Variable assignment

The current value of a variable can be printed out

calculator.cpp (Version 09)

```
int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;

   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   // A variable can be output to the screen using its name (or identifier)
   std::cout << "What is myNumber? " << myNumber << “\n";

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is 7894 divided by 548? " << 7894 / 548 << "\n";
   std::cout << "What is the remainder of 7894 divided by 548? "
      << 7894 % 548 << "\n";
   std::cout << "Verify 7894 is equal to 14 times 548 plus 222: "
      << 14 * 548 + 222 - 7894 << "\n";
}
```

What will be the output of this program ?

calculator.cpp (Version 09)

```
int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;

   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   // A variable can be output to the screen using its name (or identifier)
   std::cout << "What is myNumber? " << myNumber << “\n";

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is 7894 divided by 548? " << 7894 / 548 << "\n";
   std::cout << "What is the remainder of 7894 divided by 548? "
      << 7894 % 548 << "\n";
   std::cout << "Verify 7894 is equal to 14 times 548 plus 222: "
      << 14 * 548 + 222 - 7894 << "\n";  // Zero is the correct output
}
```

```
What is myNumber? 7894
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

calculator.cpp (Version 09)

```
int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;

   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   // A variable can be output to the screen using its name (or identifier)
   std::cout << "What is myNumber? " << myNumber << "\n";

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is 7894 divided by 548? " << 7894 / 548 << "\n";
   std::cout << "What is the remainder of 7894 divided by 548? "
      << 7894 % 548 << "\n";
   std::cout << "Verify 7894 is equal to 14 times 548 plus 222: "
      << 14 * 548 + 222 - 7894 << "\n";
}
```

"Magic numbers" are constants in programs

Remove these magic numbers. Get same correct result.

```
What is myNumber? 7894
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

calculator.cpp (Version 10)

Remove dividend constant from operations

```
int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;

   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << myNumber << " divided by 548? "
      << myNumber / 548 << "\n";
   std::cout << "What is the remainder of " << myNumber << " divided by 548? "
      << myNumber % 548 << "\n";
   std::cout << "Verify " << myNumber << " is equal to 14 times 548 plus 222: "
      << 14 * 548 + 222 - myNumber << "\n";  // Zero is the correct output
}
```

```
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

Output still correct

calculator.cpp (Version 11)

Remove divisor constant from operations

```
int main()
{
   // This statement declares a variable named "myNumber" as an integer number
   int myNumber;
   // Any integer number can be assigned to variable of type "int"
   myNumber = 7894;  // Let's use the dividend from our example below

   int myOtherNumber = 548;  // Let's use the divisor from our example

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << myNumber << " divided by " << myOtherNumber
      << " ? " << myNumber / myOtherNumber << "\n";
   std::cout << "What is the remainder of " << myNumber << " divided by "
      << myOtherNumber << " ? " << myNumber % myOtherNumber << "\n";
   std::cout << "Verify " << myNumber << " is equal to 14 times "
      << myOtherNumber << " plus 222: " << 14 * myOtherNumber + 222 - myNumber
      << "\n";  // Zero is the correct output
}
```

```
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

Output still correct

calculator.cpp (Version 12)

Let's clean up and get some space

```
int main()
{
   // Declare and assign values for our variables
   int myNumber = 7894;  // Any number of our choice
   int myOtherNumber = 548;  // Another number of our choice
   int dividend = myNumber;  // Copy value to a new variable
   int divisor = myOtherNumber;

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << dividend << " divided by " << divisor
      << " ? " << dividend / divisor << "\n";
   std::cout << "What is the remainder of " << dividend << " divided by "
      << divisor << " ? " << dividend % divisor << "\n";
   std::cout << "Verify " << dividend << " is equal to 14 times "
      << divisor << " plus 222: " << 14 * divisor + 222 - dividend
      << "\n";  // Zero is the correct output
}
```

calculator.cpp (Version 12)

```
int main()
{
   // Declare and assign values for our variables
   int myNumber = 7894;  // Any number of our choice
   int myOtherNumber = 548;  // Another number of our choice
   int dividend = myNumber;  // Copy value to a new variable
   int divisor = myOtherNumber;

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << dividend << " divided by " << divisor
      << " ? " << dividend / divisor << "\n";
   std::cout << "What is the remainder of " << dividend << " divided by "
      << divisor << " ? " << dividend % divisor << "\n";
   std::cout << "Verify " << dividend << " is equal to 14 times "
      << divisor << " plus 222: " << 14 * divisor + 222 - dividend
      << "\n";  // Zero is the correct output
}
```

```
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

Output still correct

calculator.cpp (Version 14)

Remove quotient and remainder constants

```
int main()
{
   // Declare and assign values for our variables
   int myNumber = 7894;  // Any number of our choice
   int myOtherNumber = 548;  // Another number of our choice
   int dividend = myNumber;  // Copy value to a new variable
   int divisor = myOtherNumber;
   int quotient = dividend / divisor;
   int remainder = dividend % divisor;

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << dividend << " divided by " << divisor
      << " ? " << quotient  << "\n";
   std::cout << "What is the remainder of " << dividend << " divided by "
      << divisor << " ? " << remainder << "\n";
   std::cout << "Verify " << dividend << " is equal to "
      << quotient << " times " << divisor << " plus " << remainder
      << ": " << quotient * divisor + remainder - dividend
      << "\n";  // Zero is the correct output
}
```

calculator.cpp (Version 14)

```
int main()
{
   // Declare and assign values for our variables
   int myNumber = 7894;  // Any number of our choice
   int myOtherNumber = 548;  // Another number of our choice
   int dividend = myNumber;  // Copy value to a new variable
   int divisor = myOtherNumber;
   int quotient = dividend / divisor;
   int remainder = dividend % divisor;

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << dividend << " divided by " << divisor
      << " ? " << quotient  << "\n";
   std::cout << "What is the remainder of " << dividend << " divided by "
      << divisor << " ? " << remainder << "\n";
   std::cout << "Verify " << dividend << " is equal to "
      << quotient << " times " << divisor << " plus " << remainder
      << ": " << quotient * divisor + remainder - dividend
      << "\n";  // Zero is the correct output
```

```
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

Output still correct

calculator.cpp (Version 14)

We still have two magic numbers

Let's ask the user to provide

```
int main()
{
   // Declare and assign values for our variables
   int myNumber = 7894;  // Any number of our choice
   int myOtherNumber = 548;  // Another number of our choice
   int dividend = myNumber;  // Copy value to a new variable
   int divisor = myOtherNumber;
   int quotient = dividend / divisor;
   int remainder = dividend % divisor;

   // Verify that dividend equals quotient times divisor plus remainder
   std::cout << "What is " << dividend << " divided by " << divisor
      << " ? " << quotient  << "\n";
   std::cout << "What is the remainder of " << dividend << " divided by "
      << divisor << " ? " << remainder << "\n";
   std::cout << "Verify " << dividend << " is equal to "
      << quotient << " times " << divisor << " plus " << remainder
      << ": " << quotient * divisor + remainder - dividend
      << "\n";  // Zero is the correct output
}
```

calculator.cpp (Version 18)

Let's ask the user to provide

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable “myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
   << divisor << " ? " << remainder << "\n";
std::cout << "Verify " << dividend << " is equal to "
   << quotient << " times " << divisor << " plus " << remainder
   << ": " << quotient * divisor + remainder - dividend
   << "\n";  // Zero is the correct output
```

std::cin assigns value given by user in input stream to a variable

calculator.cpp (Version 18)

Current point in Program execution

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable “myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable “myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter:
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

Once executed, this statement will wait for user input

```
Please type a number and press enter:
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

Once executed, this statement will wait for user input

```
Please type a number and press enter: 7894
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

After user input, program runs to completion

```
Please type a number and press enter: 7894
```

calculator.cpp (Version 18)

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable "myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
```

```
Please type a number and press enter: 7894
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

Output still correct

calculator.cpp (Version 18)

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable "myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
```

Let's run the same executable again

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter:
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter: 5481
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter: 5481
What is 5481 divided by 548 ? 10
What is the remainder of 5481 divided by 548 ? 1
Verify 5481 is equal to 10 times 548 plus 1: 0
```

Program output correct

calculator.cpp (Version 18)

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable "myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
```

Let's run the same executable again

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter:
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter: 299792448
```

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter: 299792448
What is 299792448 divided by 548 ? 547066
What is the remainder of 299792448 divided by 548 ? 280
Verify 299792448 is equal to 547066 times 548 plus 280: 0
```

Program output correct

calculator.cpp (Version 18)

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable "myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
```

Let's run the same executable one more time

calculator.cpp (Version 18)

```
  // Ask the user to give us a number for our first operand
  std::cout << "Please type a number and press enter: ";
  // Wait for the user to enter a number and assign it variable "myNumber"
  int myNumber;
  std::cin >> myNumber;

  int myOtherNumber = 548;  // Another number of our choice
  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
```

Current point in Program execution

```
Please type a number and press enter: 287257019000024
```

calculator.cpp (Version 18)

```
// Ask the user to give us a number for our first operand
std::cout << "Please type a number and press enter: ";
// Wait for the user to enter a number and assign it variable "myNumber"
int myNumber;
std::cin >> myNumber;

int myOtherNumber = 548;  // Another number of our choice
int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
```

```
Please type a number and press enter: 28725701900024
What is -2147483648 divided by 548 ? -3918765
What is the remainder of -2147483648 divided by 548 ? -428
Verify -2147483648 is equal to -3918765 times 548 plus -428: 0
```

Program output not correct

Largest integer C++ can store: 2,147,483,647 (or INT_MAX)

calculator.cpp (Version 19)

Let's ask the user for both operands

```
// Ask the user to give us two numbers for our operands
int 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;

int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
   << divisor << " ? " << remainder << "\n";
std::cout << "Verify " << dividend << " is equal to "
   << quotient << " times " << divisor << " plus " << remainder
   << ": " << quotient * divisor + remainder - dividend
   << "\n";  // Zero is the correct output
```

Multiple variables can be declared in a single statement

calculator.cpp (Version 19)

No more magic numbers!

```
// Ask the user to give us two numbers for our operands
int 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;

int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
std::cout << "What is the remainder of " << dividend << " divided by "
   << divisor << " ? " << remainder << "\n";
std::cout << "Verify " << dividend << " is equal to "
   << quotient << " times " << divisor << " plus " << remainder
   << ": " << quotient * divisor + remainder - dividend
   << "\n";  // Zero is the correct output
```

calculator.cpp (Version 19)

```
  // Ask the user to give us two numbers for our operands
  int 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;

  int dividend = myNumber;  // Copy value to a new variable
  int divisor = myOtherNumber;
  int quotient = dividend / divisor;
  int remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
     << " ? " << quotient  << "\n";
```

```
Please type a number and press enter: 7894
Please type another number and press enter: 548
What is 7894 divided by 548? 14
What is the remainder of 7894 divided by 548? 222
Verify 7894 is equal to 14 times 548 plus 222: 0
```

Program output correct

calculator.cpp (Version 19)

```
// Ask the user to give us two numbers for our operands
int 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;

int dividend = myNumber;  // Copy value to a new variable
int divisor = myOtherNumber;
int quotient = dividend / divisor;
int remainder = dividend % divisor;

// Verify that dividend equals quotient times divisor plus remainder
std::cout << "What is " << dividend << " divided by " << divisor
   << " ? " << quotient  << "\n";
```

Let's run the same executable again

Pi

From Wikipedia, the free encyclopedia

This article is about the mathematical constant. For the Greek letter, see Pi (letter). For other uses, see Pi (disambiguation).

The number π (/paɪ/; spelled out as "pi") is a mathematical constant, approximately equal to 3.14159. It is defined in Euclidean geometry[a] as the ratio of a circle's circumference to its diameter, and also has various equivalent definitions. The number appears in many formulas in all areas of mathematics and physics. The earliest known use of the Greek letter π to represent the ratio of a circle's circumference to its diameter was by Welsh mathemati
constant.[2][3][4]

3.14286 is a floating point approximation of π

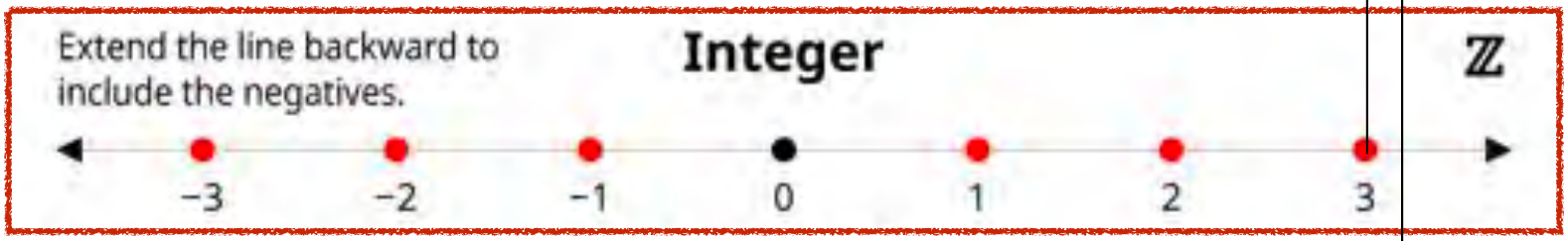
Being an irrational number, π cannot be expressed as a common fraction, although fractions such as $\frac{22}{7}$ are commonly used to approximate it. Equivalently, its decimal representation never ends and never settles into a permanently repeating pattern. Its decimal (or other base) digits appear to be randomly distributed, and are conjectured to satisfy a specific kind of statistical randomness.

3 is an integer approximation of π

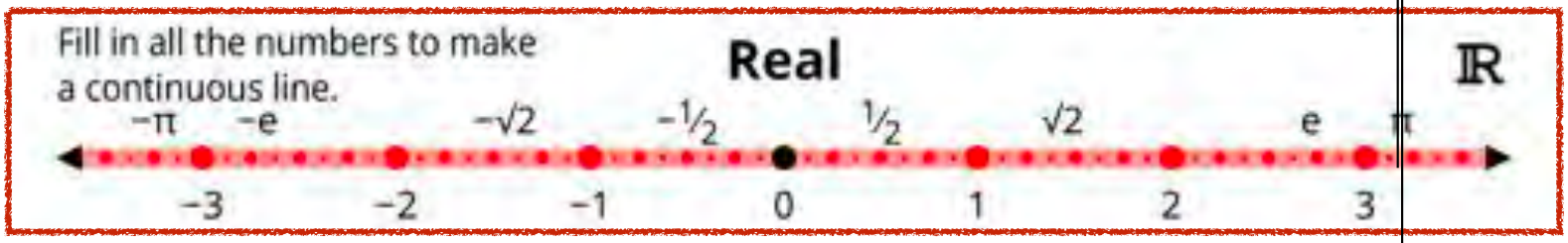
```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 divided by 7 ? 3
What is the remainder of 22 divided by 7 ? 1
Verify 22 is equal to 3 times 7 plus 1: 0
```

Program output correct but ... suppose π is needed

3 is an integer approximation of π



3.14286 is a floating point approximation of π



calculator.cpp (Version 20)

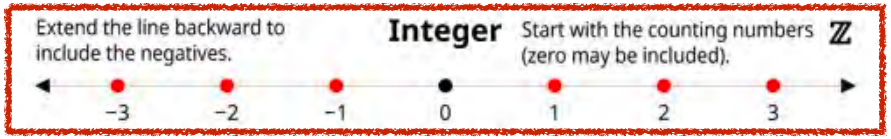
Just change all `int` to `float`

```
   // 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;

   float dividend = myNumber;  // Copy value to a new variable
   float divisor = myOtherNumber;
   float quotient = dividend / divisor;
   float remainder = dividend % divisor;

   // Verify that dividend equals quotient times divisor plus remainder
```

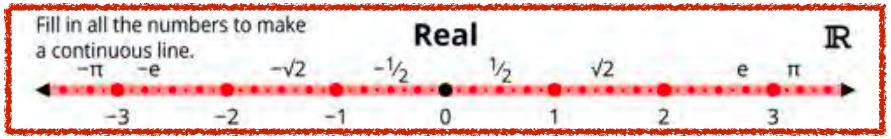
```
calculator.cpp:18:31: error: invalid operands to binary expression
      ('float' and 'float')
   float remainder = dividend % divisor;  // "%" not defined for float type
                     ~~~~~~~~ ^ ~~~~~~~
1 error generated.
```

```
      << "\n";  // Zero is the correct output
```

`int` *data type*



`float` *data type*



Operators perform basic arithmetic operations

+	Addition	+
-	Subtraction	-
*	Multiplication	*
/	Division	/
%	Modulus	

No remainder for real numbers

calculator.cpp (Version 22)

Remove code for integer division

```
  // 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;

  float dividend = myNumber;  // Copy value to a new variable
  float divisor = myOtherNumber;
  float quotient = dividend / divisor;
  float remainder = dividend % divisor;

  // Verify that dividend equals quotient times divisor plus remainder
  std::cout << "What is " << dividend << " divided by " << divisor
      << " ? " << quotient  << "\n";
  std::cout << "What is the remainder of " << dividend << " divided by "
      << divisor << " ? " << remainder << "\n";
  std::cout << "Verify " << dividend << " is equal to "
      << quotient << " times " << divisor << " plus " << remainder
      << ": " << quotient * divisor + remainder - dividend
      << "\n";  // Zero is the correct output
```

calculator.cpp (Version 22)

Add statement for floating point division

```
// 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;

// Perform division operation and output result to screen
std::cout << "What is " << myNumber << " divided by " << myOtherNumber
   << " ? " << myNumber / myOtherNumber << "\n";
```

calculator.cpp (Version 22)

```
  // 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;

  // Perform division operation and output result to screen
  std::cout << "What is " << myNumber << " divided by " << myOtherNumber
     << " ? " << myNumber / myOtherNumber << "\n";
```

I am using the cursors to denote where user input is prompted

```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 divided by 7 ?
```

What will be the output of this operation ?

calculator.cpp (Version 22)

```
  // 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;

  // Perform division operation and output result to screen
  std::cout << "What is " << myNumber << " divided by " << myOtherNumber
     << " ? " << myNumber / myOtherNumber << "\n";
```

```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 divided by 7 ? 3.14286
```

Can we get a better approximation of π ?

calculator.cpp (Version 22)

```
  // 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;

  // Perform division operation and output result to screen
  std::cout << "What is " << myNumber << " divided by " << myOtherNumber
     << " ? " << myNumber / myOtherNumber << "\n";
```

```
Please type a number and press enter: 245850922
Please type another number and press enter: 78256779
What is 2.45851e+08 divided by 7.82568e+07 ? 3.14159
```

*Scientific notation: 2.45851e+08 = 2.45851 * 10^8 ≈ 245850922*

calculator.cpp (Version 22)

```
// 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;

// Perform division operation and output result to screen
std::cout << "What is " << myNumber << " divided by " << myOtherNumber
   << " ? " << myNumber / myOtherNumber << "\n";
```

Which operation should we perform?
Let's provide them all

calculator.cpp (Version 23)

Perform all operations for the user

```
// 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;

// Perform all operations and output result to screen
std::cout << "What is " << myNumber << " plus " << myOtherNumber << "? "
   << myNumber + myOtherNumber << "\n";
std::cout << "What is " << myNumber << " minus " << myOtherNumber << " ? "
   << myNumber - myOtherNumber << "\n";
std::cout << "What is " << myNumber << " times " << myOtherNumber << " ? "
   << myNumber * myOtherNumber << "\n";
std::cout << "What is " << myNumber << " divided by " << 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
```

calculator.cpp (Version 24)

Remove some unnecessary magic text

```
// 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";
```

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
```

Next lecture: Functions

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

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
```

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

Coming

wall_follower.cpp - Project 1

```
while [redacted] {
    LidarScan scan = readLidarScan(drv);

    if [redacted] {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = [redacted]
        float [redacted]
        float [redacted]

        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;
        [redacted]

        // Create a vector that points up.
        [redacted]

        // Get a vector that is perpendicular to the nearest obstacle.
        Vector3D forward_v = [redacted]

        float vx = [redacted]
        float vy = [redacted]
        std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

        [redacted]

        vx += [redacted]
        vy += [redacted]

        [redacted]

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

Things to think about

- Why would anyone use an int when they could use a float ?
- Is 22/7 the same thing as 22.0/7.0 ?
- What should our program do if a user requests 102/0.0 ?
- Can we do operations in succession like a calculator?
- What is 8/2*(2+2)?

```
/* Robotics 102 - Fall 2021
   Introduction to AI and Programming

   C++ Operators and Variables */

shapeYCoordinate = sqrt(cos(x))*cos(300*x)
   +sqrt(abs(x))-0.7*(4-x*x)^0.01;
boundaryUpper = sqrt(6-x^2);
boundaryLower = -sqrt(6-x^2);
```