This is 183

L02: Week 2 - Monday

Announcements

- P1 has been released
- Discussion sections start this week
- iClicker starts on Wednesday
- Office hours start this week
  - Monday 3-6pm, Tues-Fri 3-8pm
  - Duderstadt 3rd floor, west side
  - Sign in sheet – name, issue, location
    - All workstations are numbered, give the closest one
    - You can request a CAEN account, visit CAEN desk, 1st floor

Announcements

- Office hours start this week
  - Monday 3-6pm, Tues-Fri 3-8pm
  - Duderstadt 3rd floor, west side
  - Sign in sheet – name, issue, location
  - All workstations are numbered, give the closest one
  - You can request a CAEN account, visit CAEN desk, 1st floor

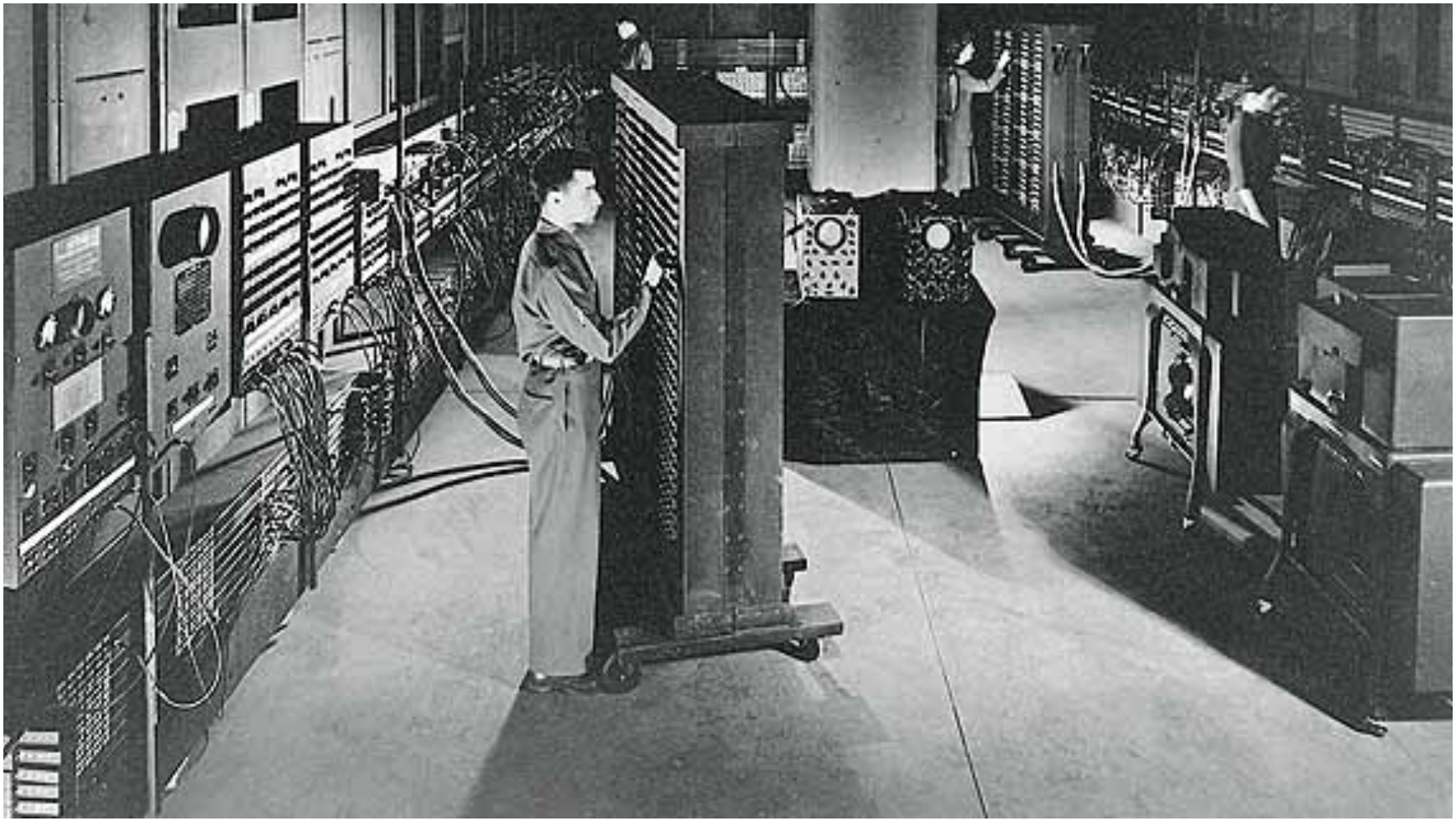
Peer Mentorship Opportunity

- **Who:** gEECS
- **What:** peer mentorship and tutoring groups
- **Where:** TBD
- **When:** throughout the semester

- **Note:** This is open to all students to sign up
- Offered to EECS 183, EECS 280, ENGR 101

- **Apply by Jan. 16** (application link in email announcement)

Eniac – 1st general purpose digital computer - 1946

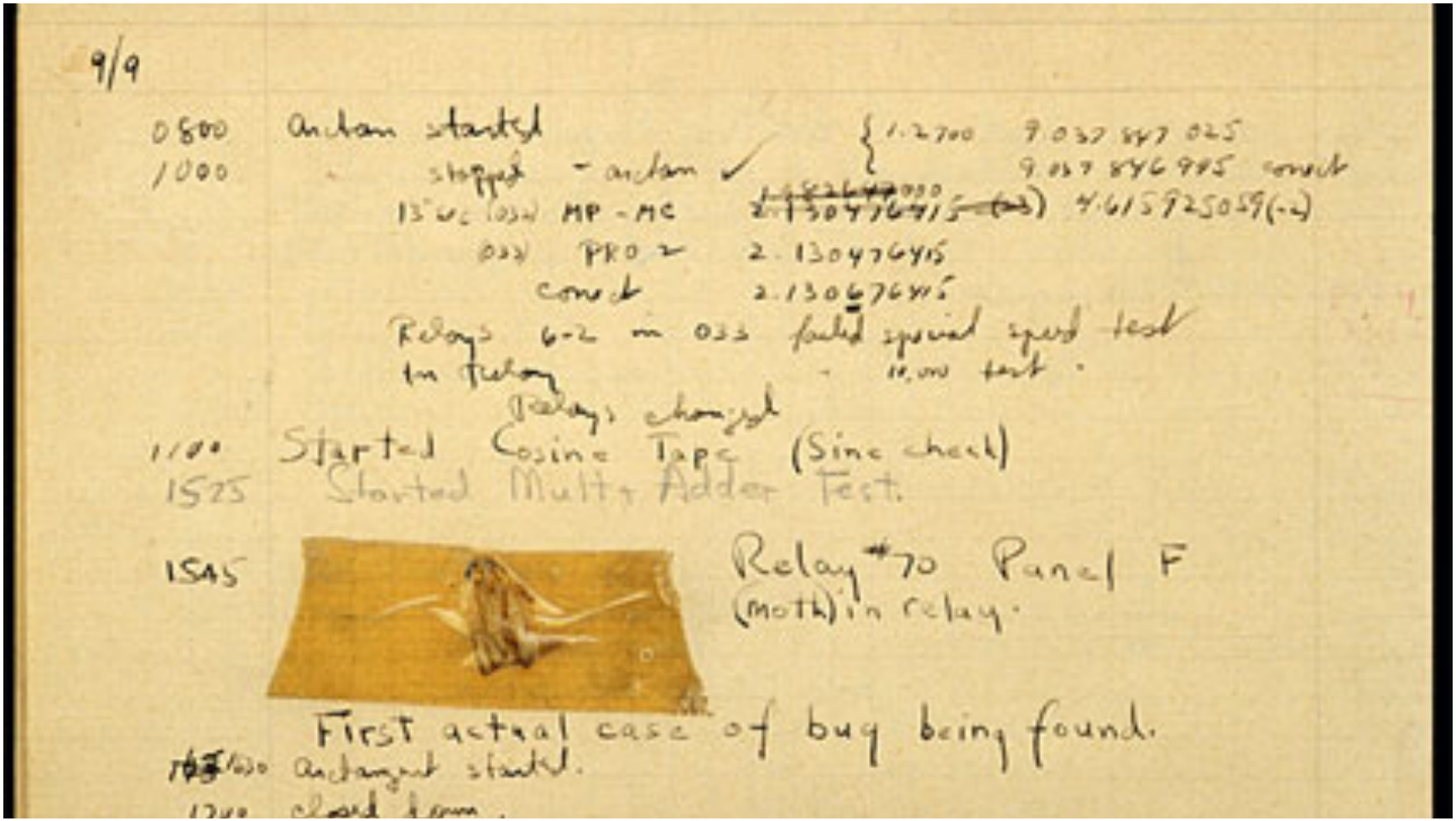


http://www.youtube.com/watch?v=HJUo8t220Rk

Parts of the Eniac

http://www.maximumpc.com/12-things-you-didnt-know-about-eniac/

1947: A "bug"



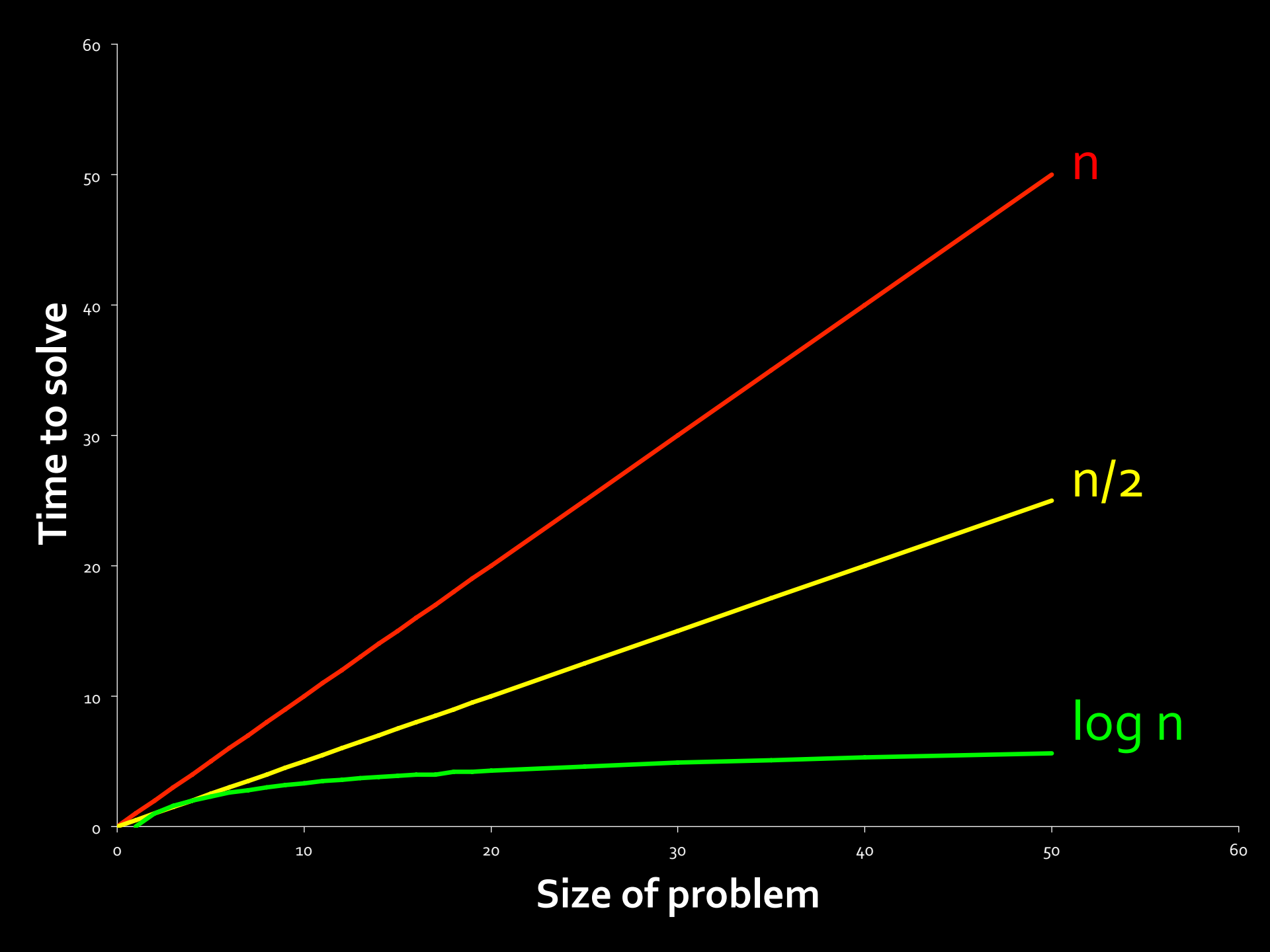
9/9

0800 Arctan started
1000 " stopped - arctan ✓
13" wc (033) MP - MC
033 PRO 2
correct
Relays 6-2 in 033 failed special speed test
In relay
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1700 Arctangent started.
closed down.

Last Time

Algorithms & Pseudocode

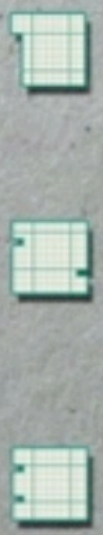
- An algorithm is a specific approach to solve a problem.
  - Counting people in the room
  - Finding a certain page in a book
- Different algorithms vary in efficiency and speed
  - By halving the problem each time, we vastly improved the search through the book


Time to solve
Size of problem
n
n/2
log n
0
10
20
30
40
50
60

Pseudo code

(RESEMBLING) (PROGRAMMING LANGUAGE)

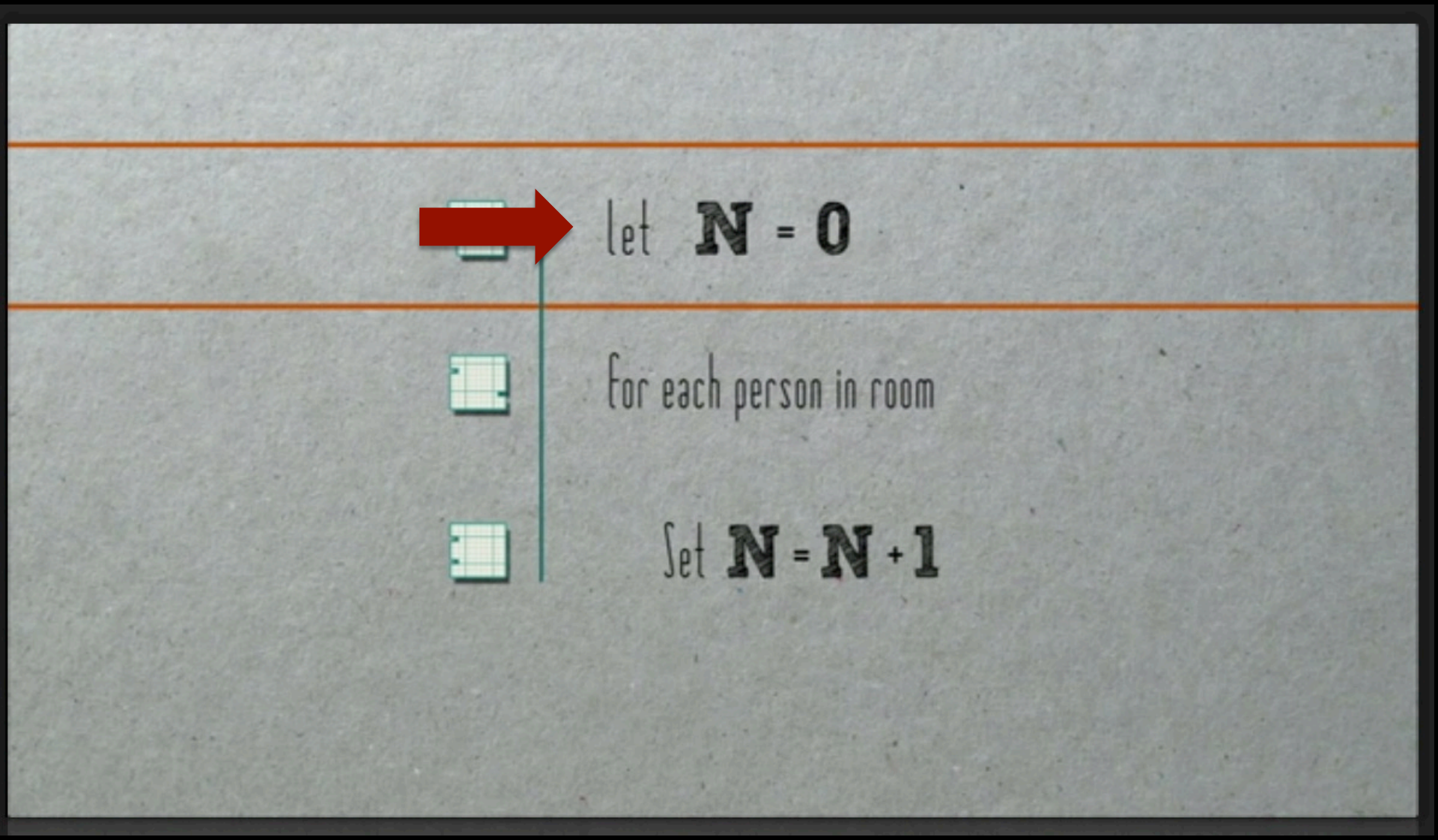
Pseudocode



let **N = 0**

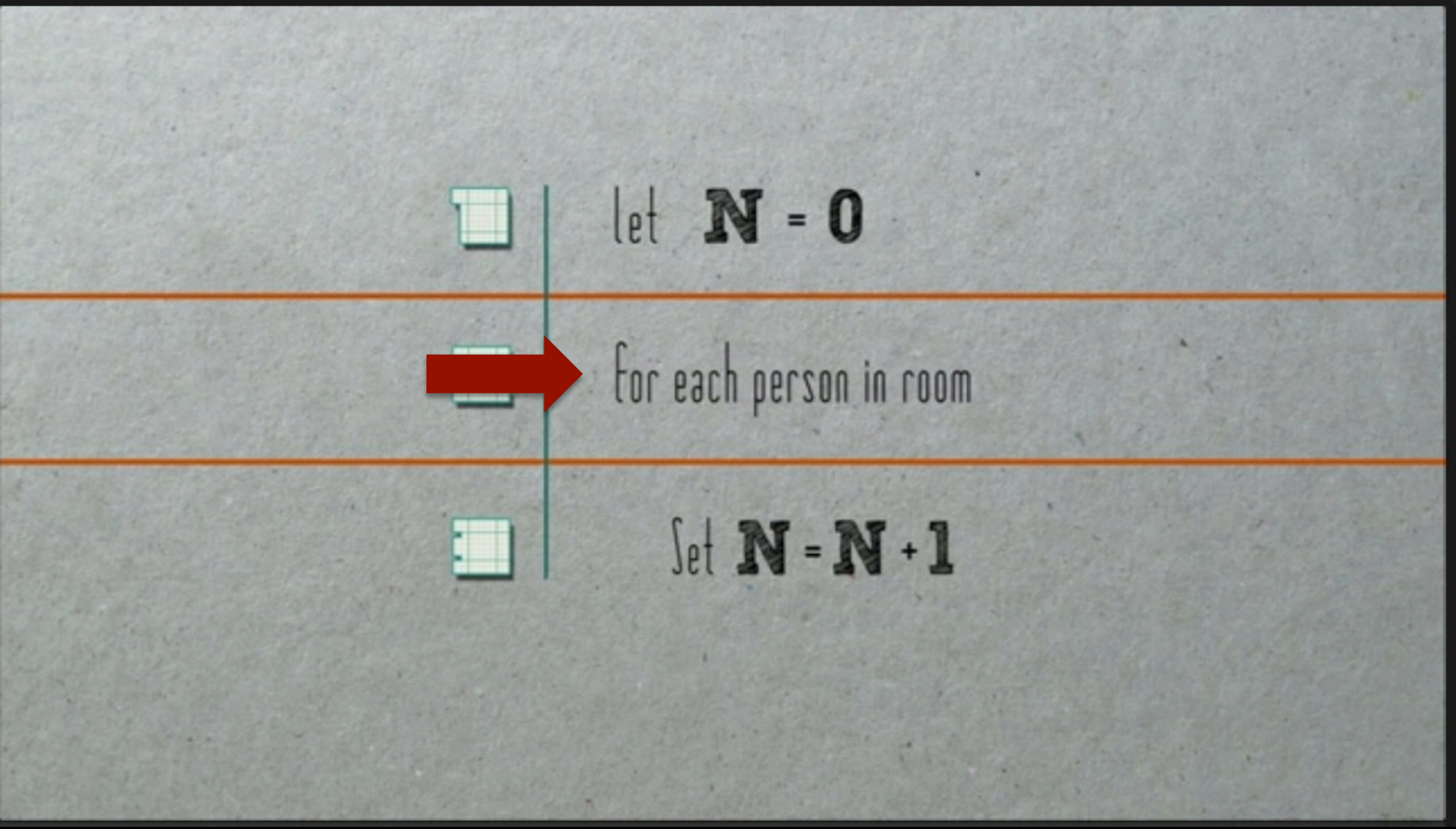
for each person in room

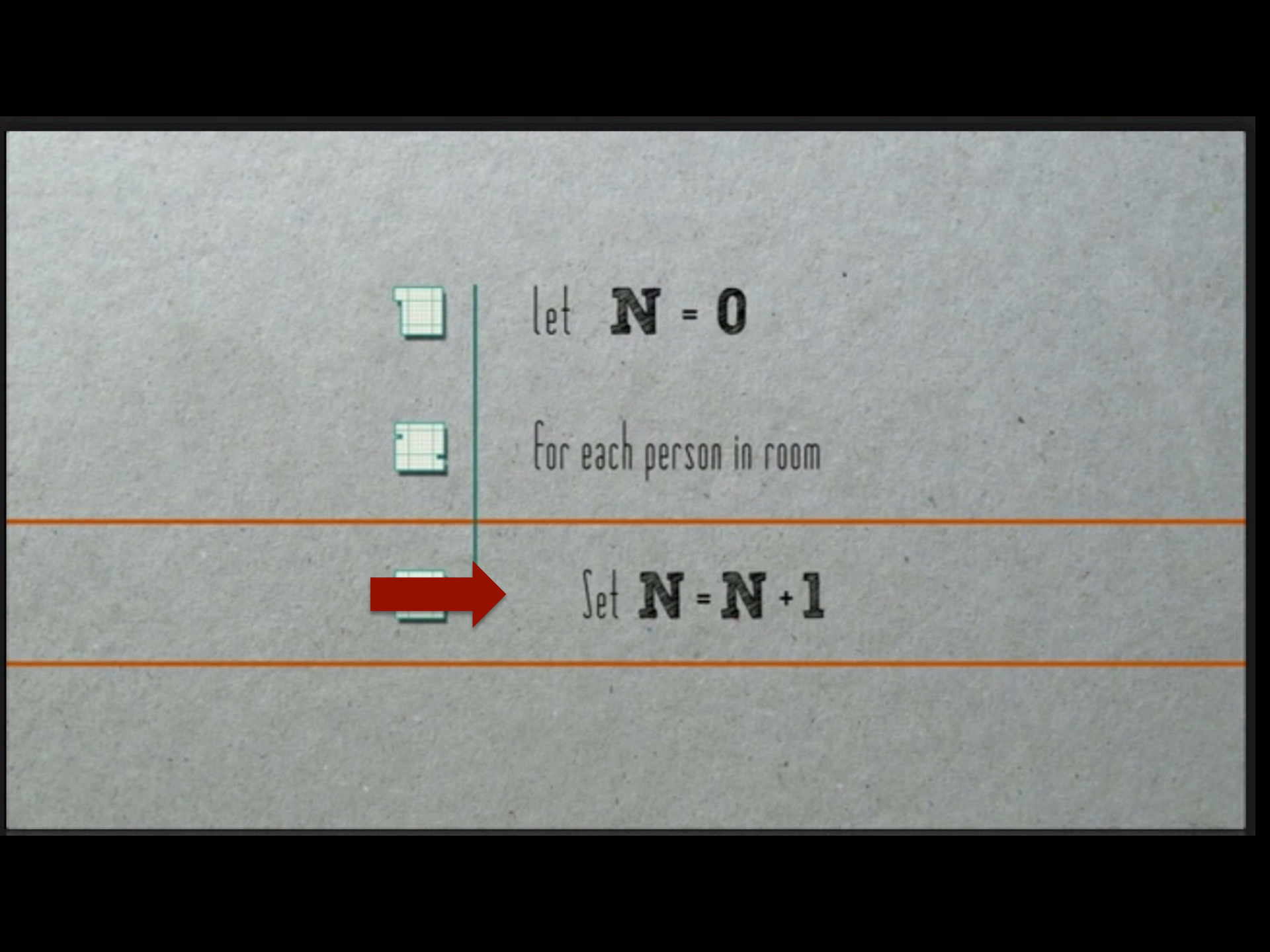
Set **N = N + 1**


let N = 0
For each person in room
Set N = N + 1

Loop

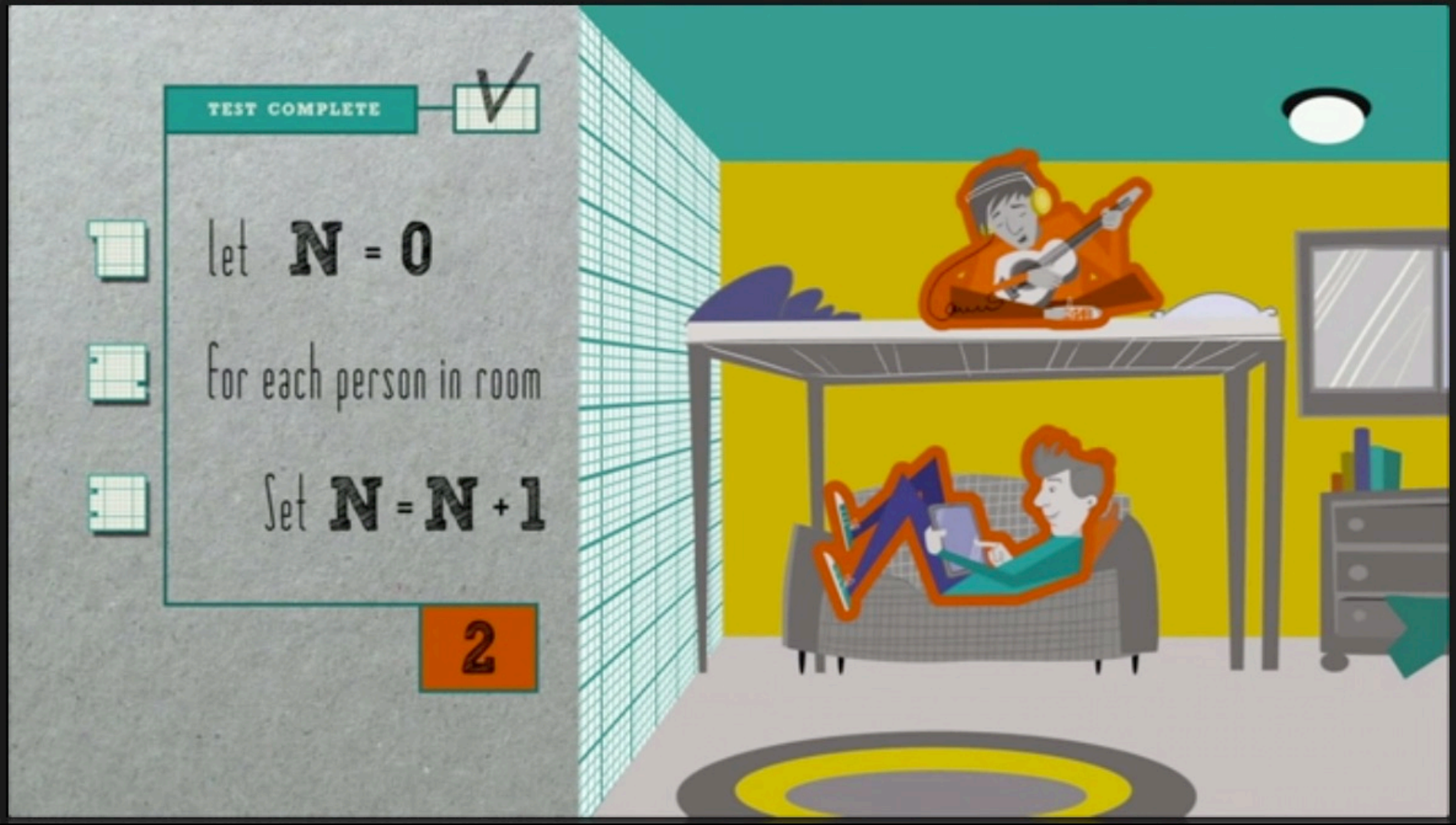
Do something repeatedly


let N = 0
For each person in room
Set N = N + 1

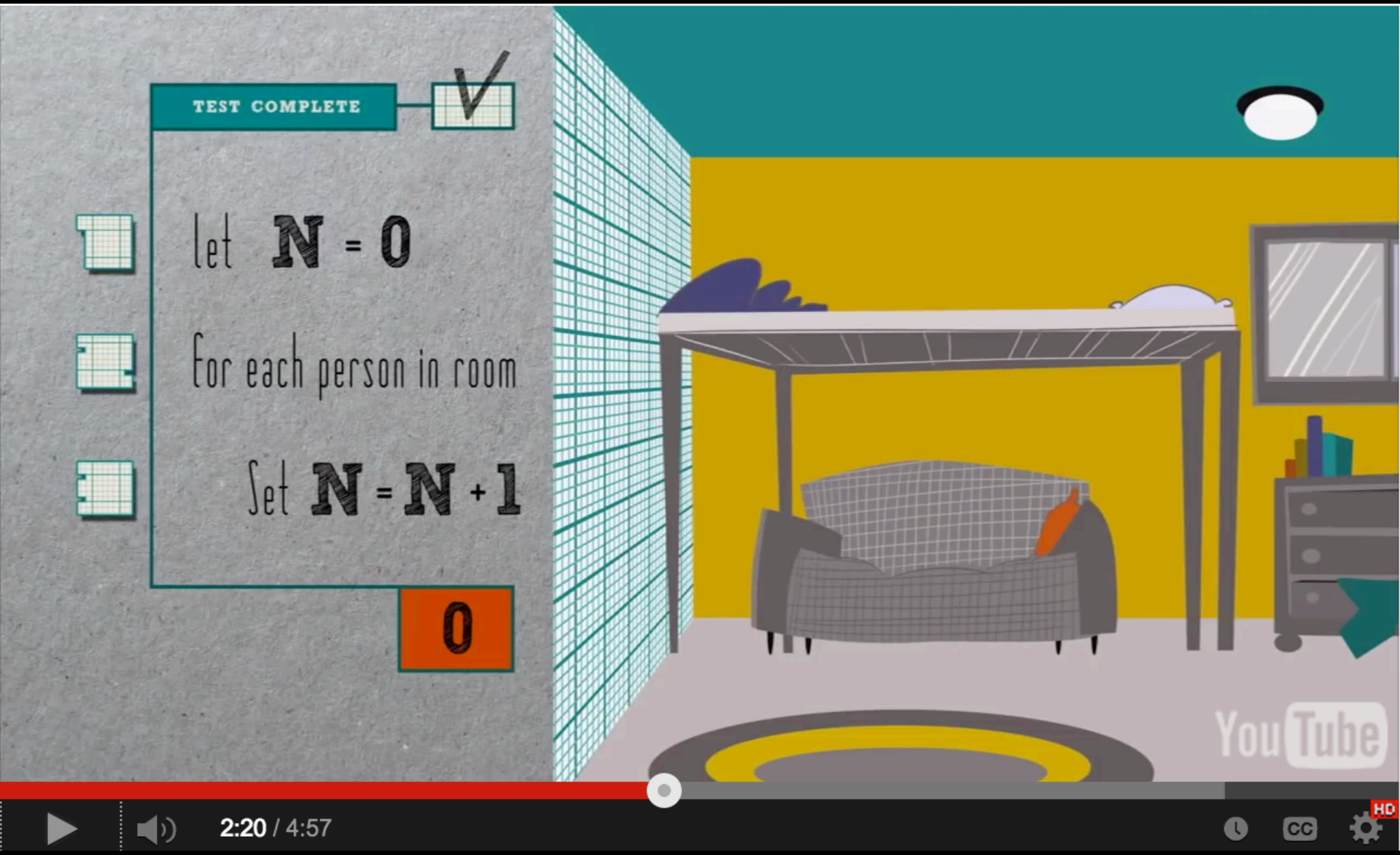

let N = 0
For each person in room
Set N = N + 1

Testing is Critical

You must **always** test your algorithms.

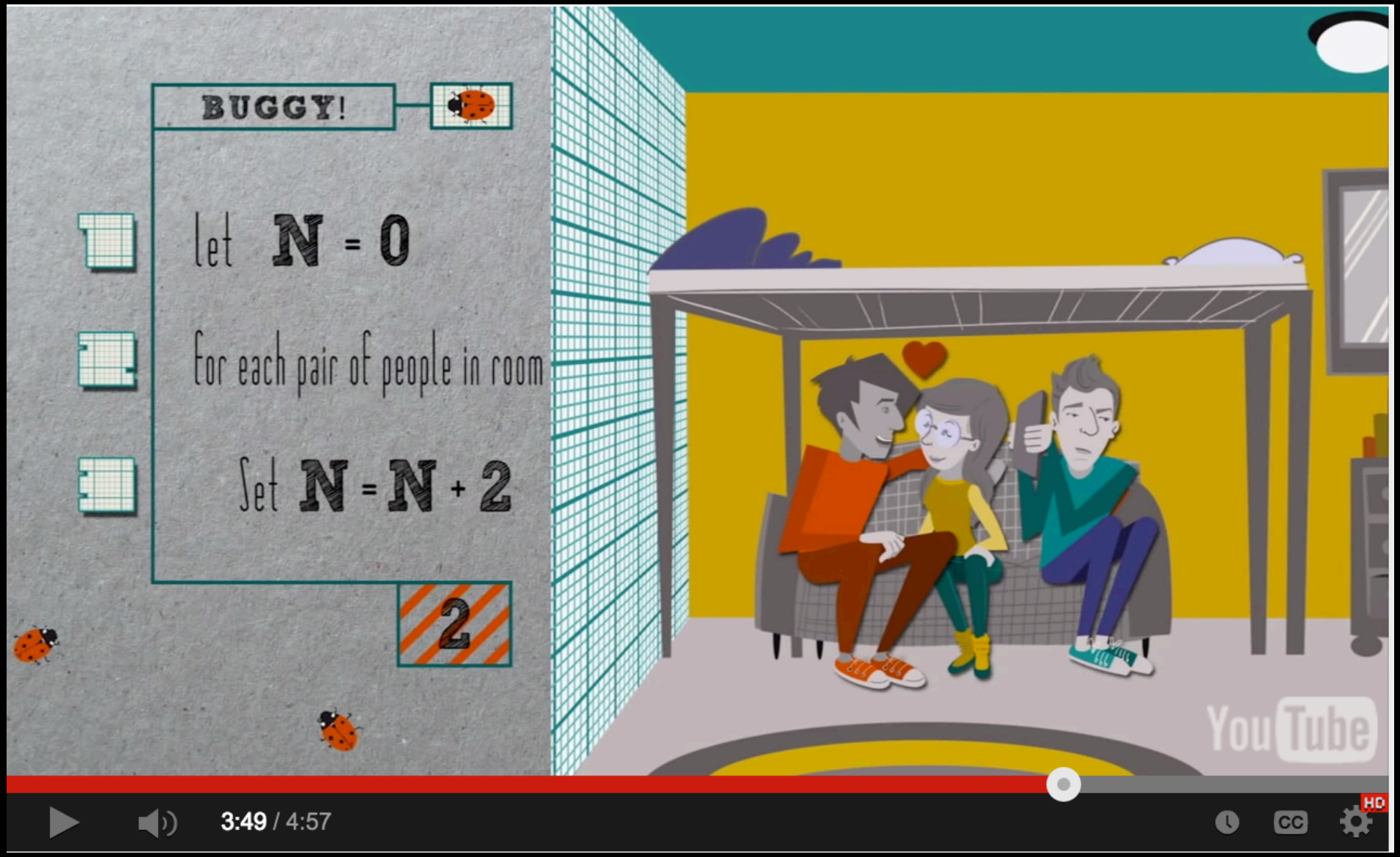


Initial algorithm for 1 at-a-time worked ok.



It even worked with 0 people in the room.

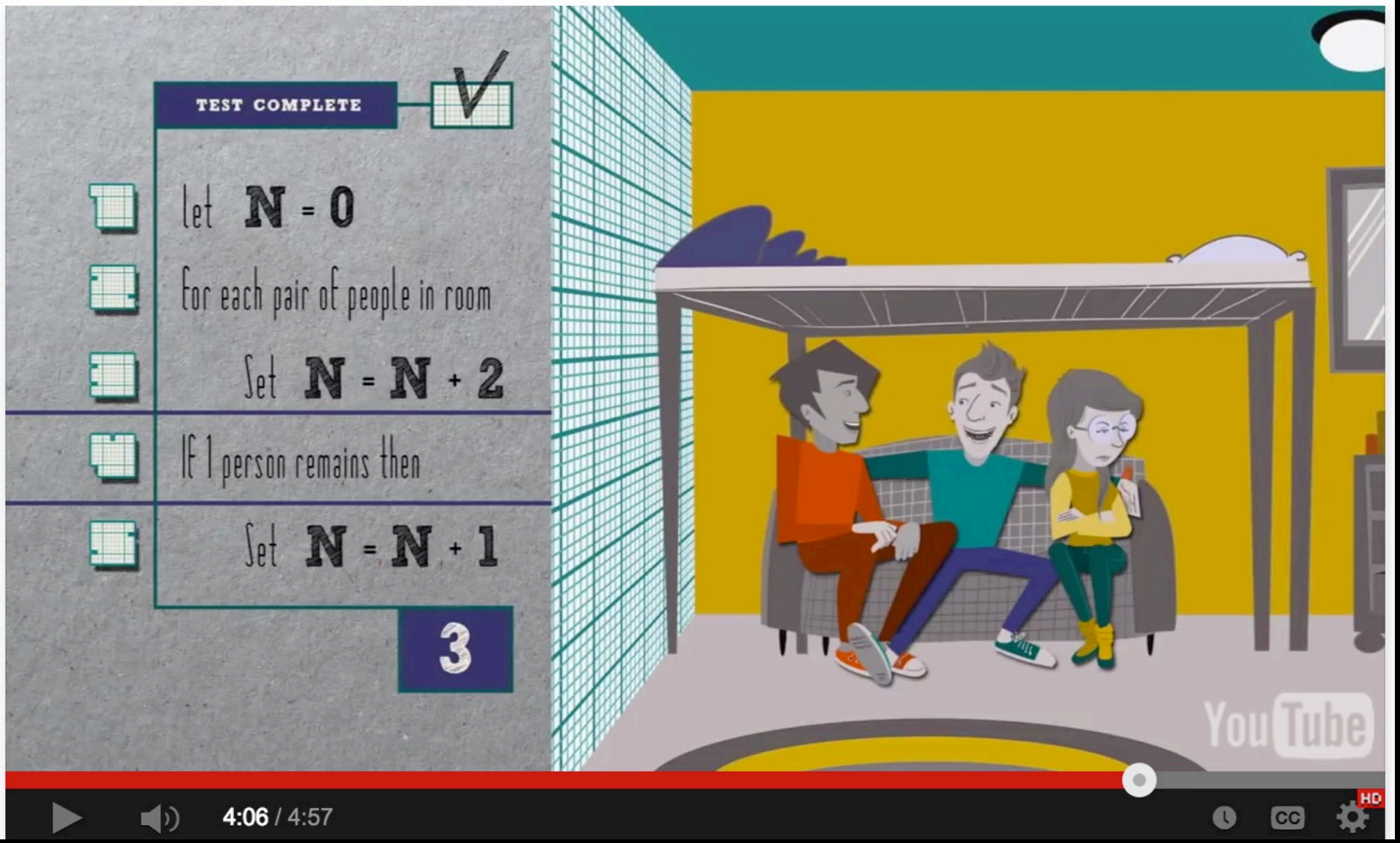
Initial algorithm for 2 at-a-time seemed ok.



But, it broke when we tried counting an odd number.

Condition

Do different things based on a condition



If one person remains…

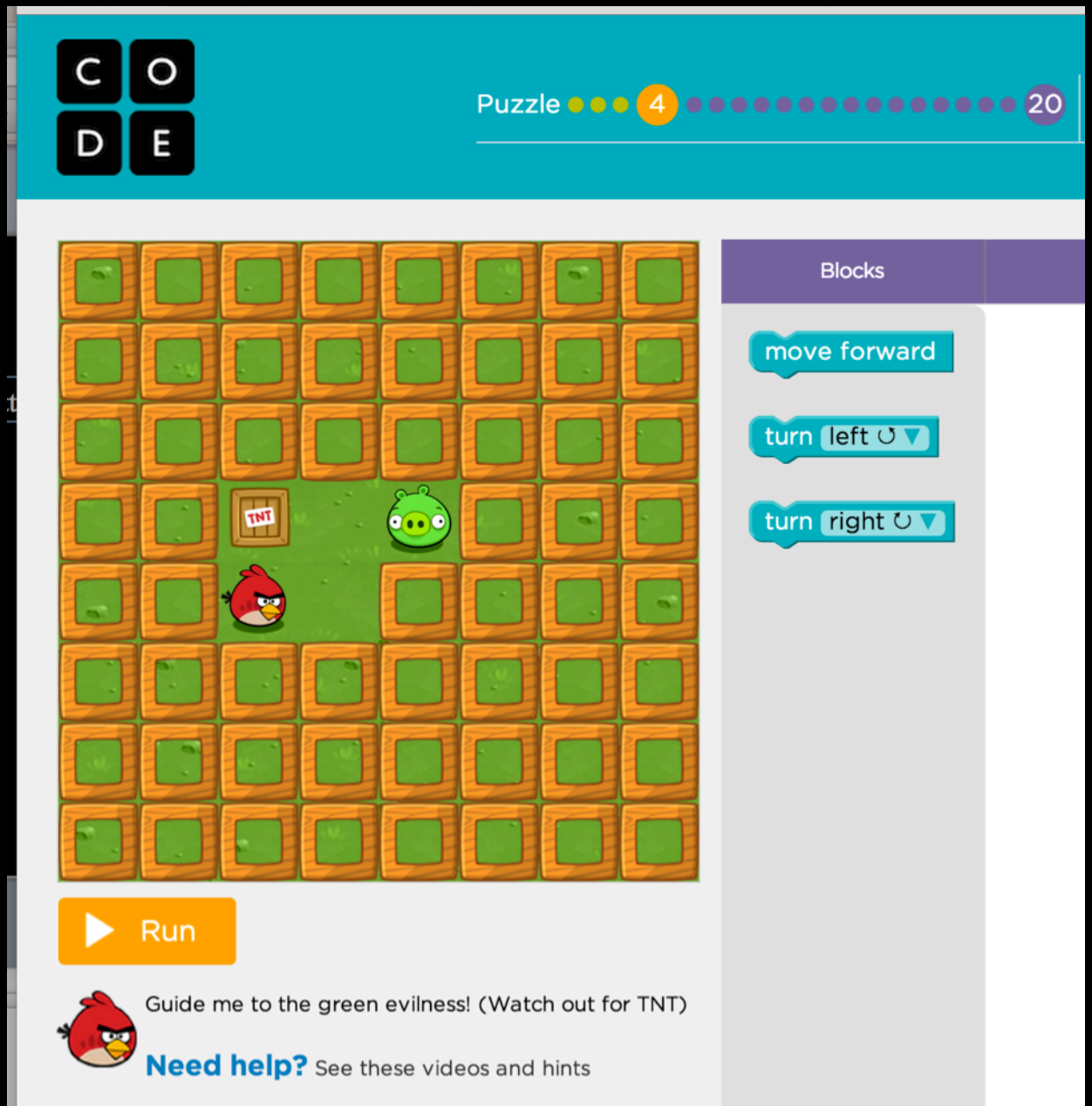
Code

Source Code

Pseudocode: An algorithm expressed in your own words

Source code: An Algorithm expressed in a programming language.

http://learn.code.org/hoc/8



http://studio.code.org/hoc/12

Source Code in C++

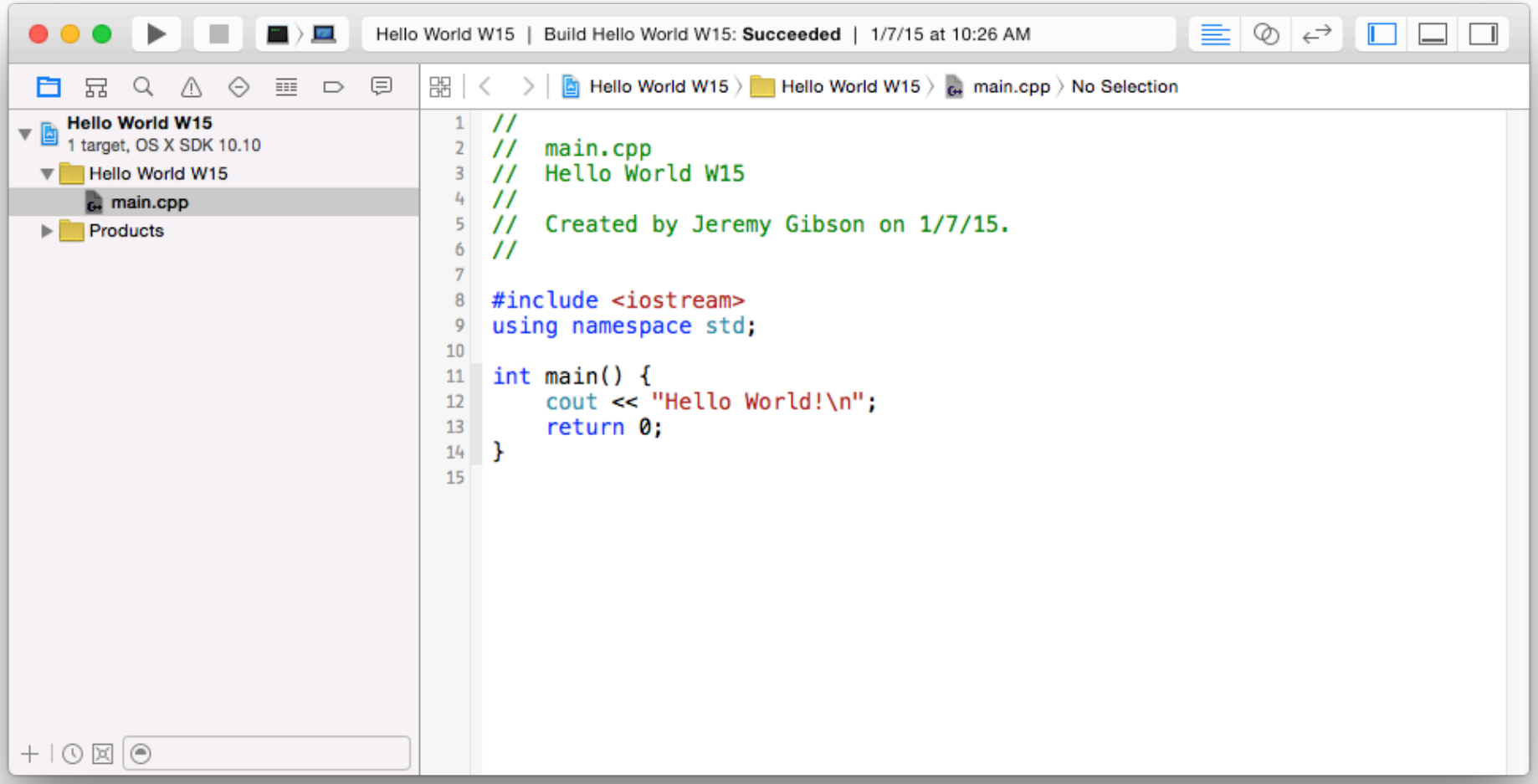
Sunday tutorial was about
Meet the Staff
Xcode / Visual Studio

If you missed it:
The video should be online

Getting Started With an IDE

- Course Website: eecs183.org
- Look in: Assignments > Project 1> Creating a Project
  - Getting Started with Visual Studio (Windows)
  - Getting Started with Xcode (Mac)

Code Editor

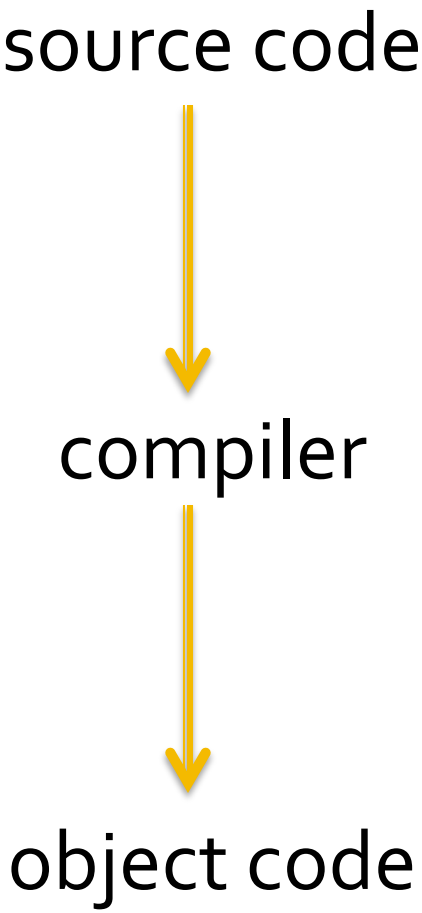

Hello World W15 | Build Hello World W15: Succeeded | 1/7/15 at 10:26 AM
Hello World W15 〉 Hello World W15 〉 main.cpp 〉 No Selection
Hello World W15
1 target, OS X SDK 10.10
Hello World W15
main.cpp
Products
//
// main.cpp
// Hello World W15
//
// Created by Jeremy Gibson on 1/7/15.
//
#include <iostream>
using namespace std;
int main() {
cout << "Hello World!\n";
return 0;
}

Compiler

source code

compiler

Compiler



10000011 00000001 00010001 00000000 00111101 11111100 01110100 00111101

00000000 01000000 00000000 00000000 00000000 00000000 00000000 00000000

10010000 00000000 00000000 00000000 01010000 00000000 00000111 00110000

00001011 00000001 00001011 00000011 00001010 00000000 00000000 00000000

00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000

00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

01110000 00010000 00000000 00100000 00000001 00000000 00000000 00000000

00000000 00000000 00000000 00100000 00000001 00000000 00000000 00000000

00000000 00000000 00000000 01000000 00000001 00000000 00000000 00000000

00000000 00100000 00000000 01000000 00000001 00000000 00000000 00000000

11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111

10010000 10000000 00000000 01000000 00000001 00000000 00000000 00000000

00101110 01100100 01111001 01101110 01100001 01101101 01101001 01100011

10100000 00000001 00000000 00000000 00000000 00000000 00000000 00000000

10110000 00000100 00000000 00000000 00000000 00000000 00000000 00000000

Source Code in C++

Now, let's look at that C++ source code

a "Hello World" program in C++

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

int main(void) {
    cout << "Hello World!";
    return 0;
}
```

Zyante readings - Questions

Q1:

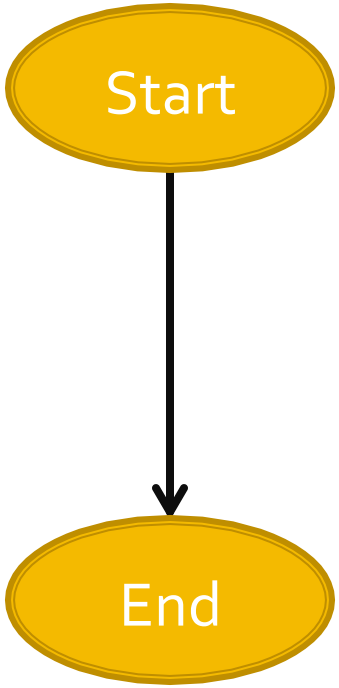
```
1 #include <iostream>
2 using namespace std;
3
4 int main(void) {
5     cout << "Hello World!";
6     return 0;
7 }
```

Where does the program begin?

A) line 1
B) line 2
C) line 4
D) line 5
E) line 6

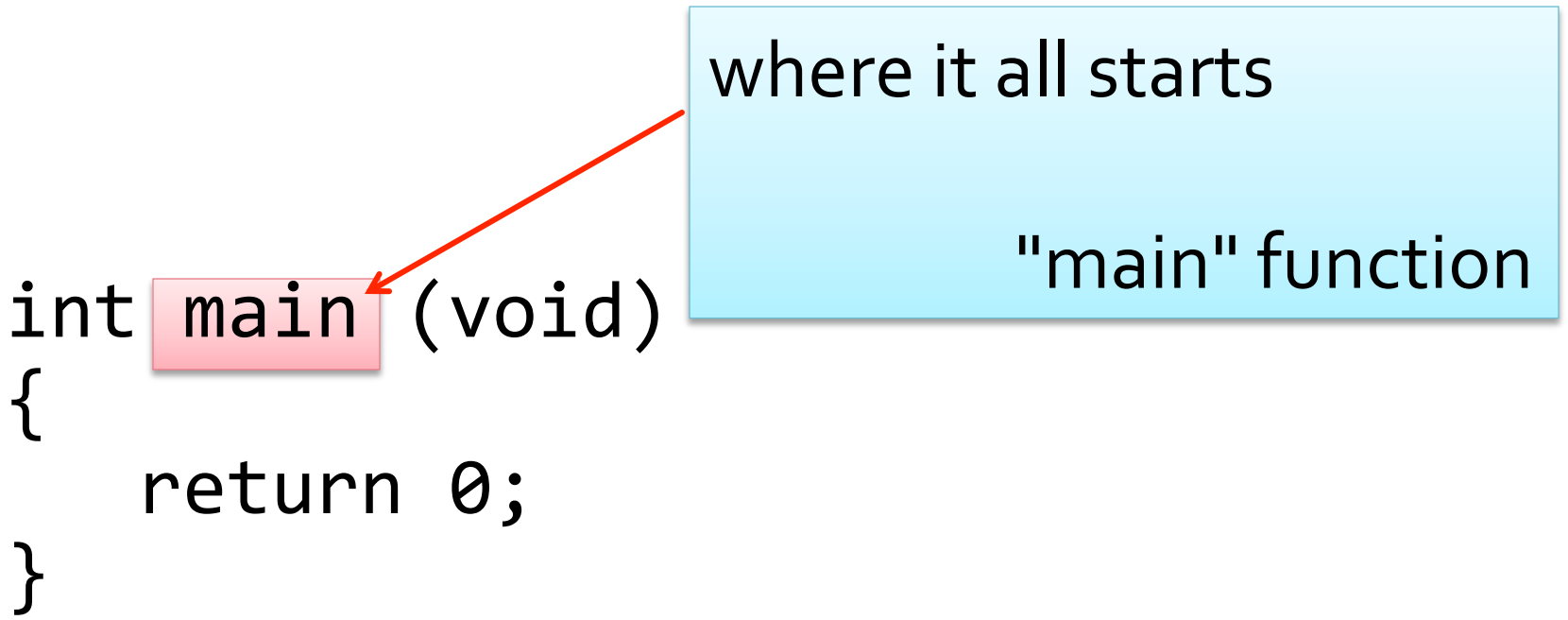
First Program

```
int main (void)
{
   return 0;
}
```



.. and go back to caller
(usually the OS)

First Program



First Program

```
int main (void)
{
    return 0;
}
```

values passed to the program (nothing in this case)

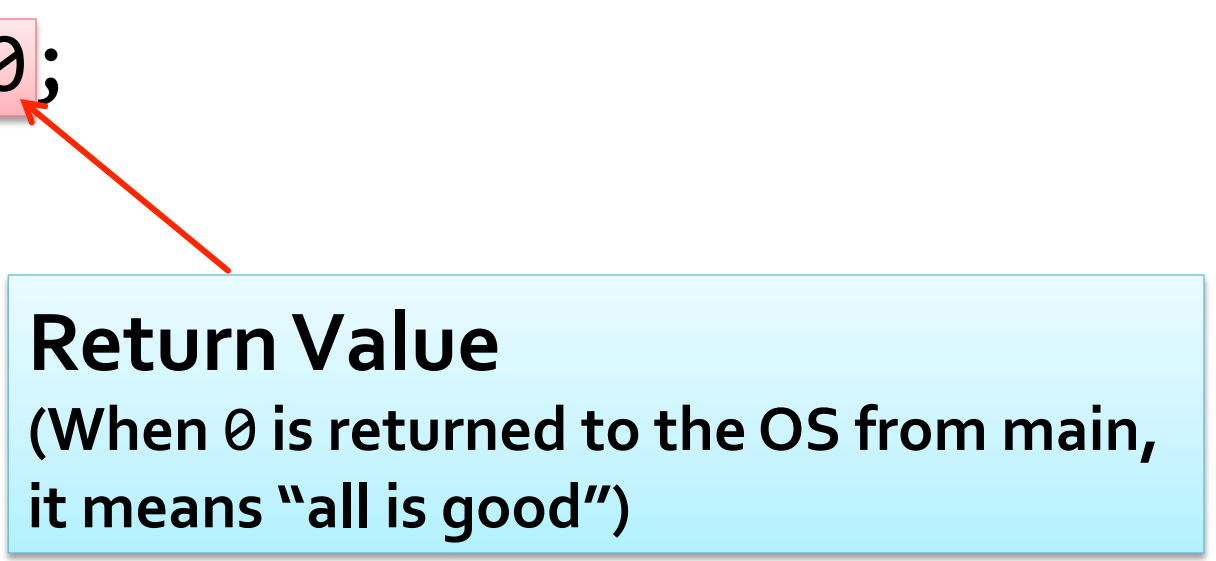
First Program

```
int main (void)
{
    return 0;
}
```

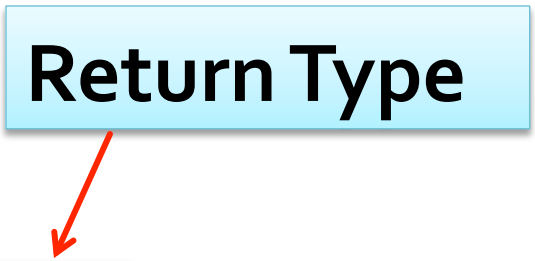
Go back to where the function was called from
(OS in case of 'main')

First Program

```
int main (void)
{
    return 0;
}
```

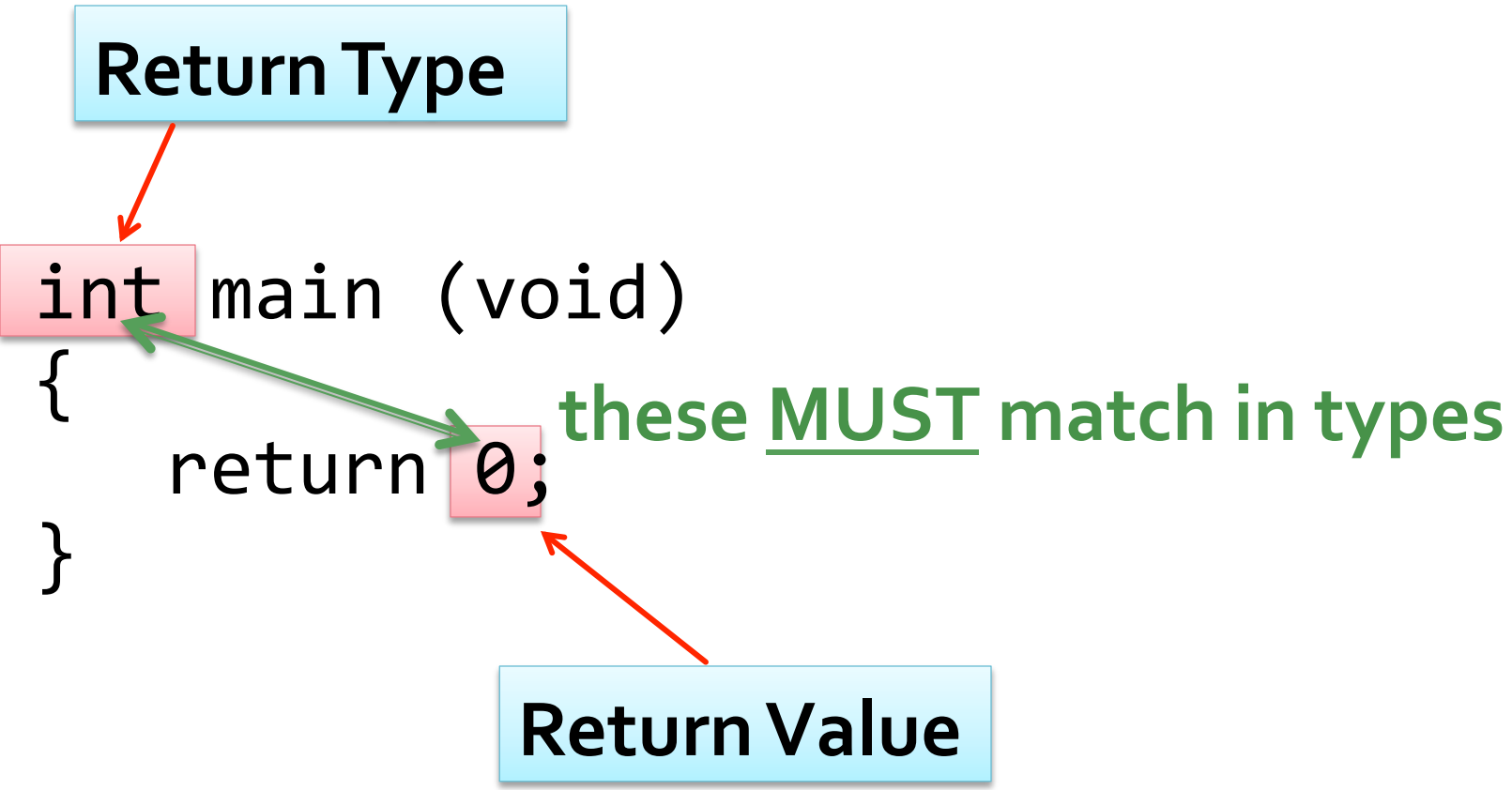


First Program



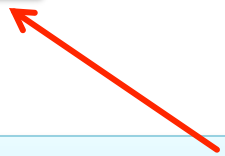

```
int main (void)
{
    return 0;
}
```

First Program



First Program

```
int main (void)
{
    return 0;
}
```



Denotes the end of a simple statement

First Program

```
int main (void)
{
    return 0;
}
```

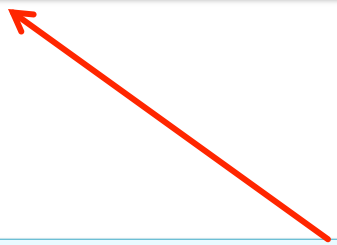
Braces denote the beginning and the end of code blocks. In this case, they define where the function starts and ends.

"Hello World" in C++

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

int main (void)
{
   cout << "Hello World!" << endl;
   return 0;
}
```

"Hello World" in C++

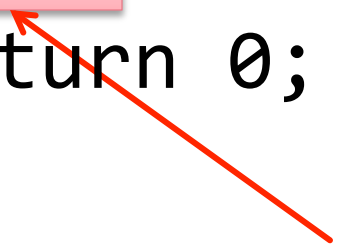
```
#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
```



A *Statement.* Prints "Hello World!" to the standard output stream

"Hello World" in C++

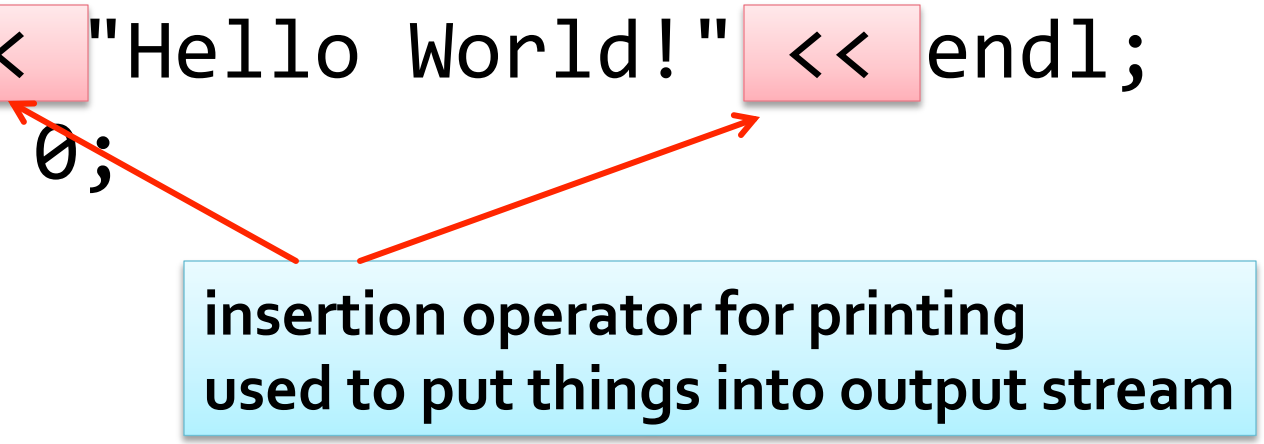
```
#include <iostream>
using namespace std;

int main (void)
{
   cout << "Hello World!" << endl;
   return 0;
}
```



object to send the output
Basically, says to print to standard output stream

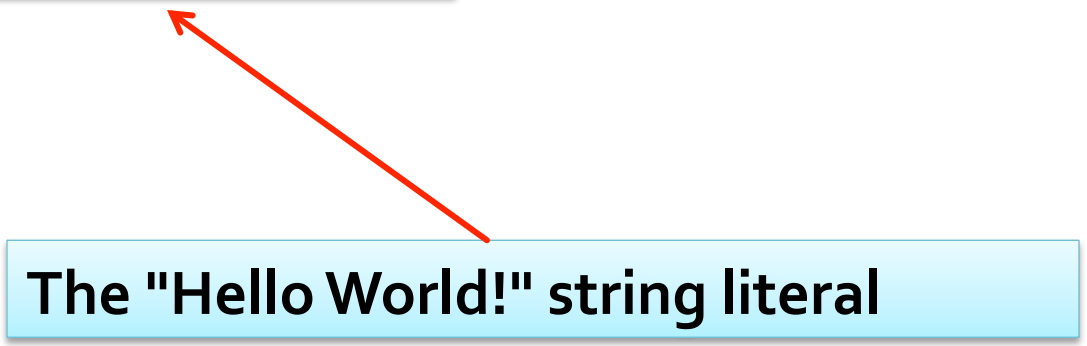
"Hello World" in C++

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

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
```



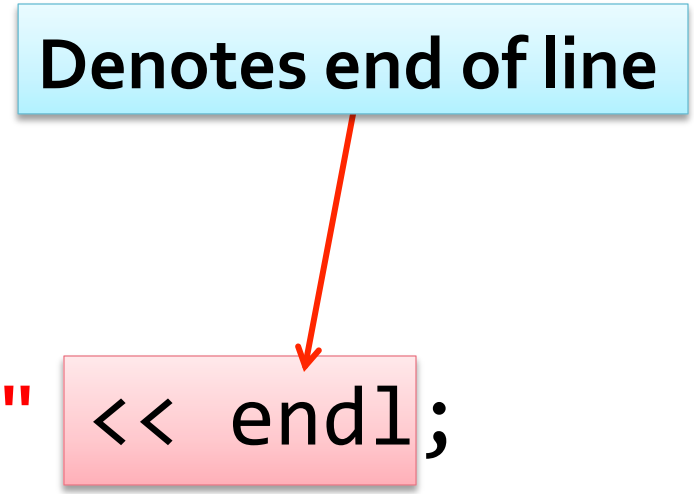
"Hello World" in C++

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

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
```



Let's add output

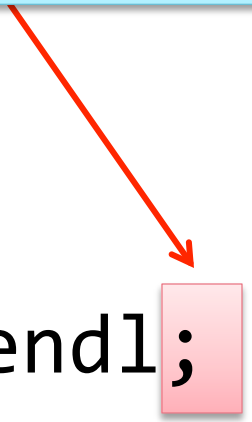
```
#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
```



Let's add output

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

int main (void)
{
   cout << "Hello World!" << endl;
   return 0;
}
```

C++
this is a STATEMENT



Let's add output

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

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
```

I/O operations require including the I/O standard file from the C++ standard library. Enables cout

Let's add output

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

int main (void)
{
   cout << "Hello World!" << endl;
   return 0;
}
```

A directive that tells the preprocessor to include code of iostream library before compilation

Let's add output

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

int main (void)
{
   cout << "Hello World!" << endl;
   return 0;
}
```

All files in the C++ standard library declare all of its entities within the std namespace.

Q2:

Which statement prints: Welcome!

```
A) cout << Welcome!;
B) cout >> "Welcome!";
C) cout << "Welcome!";
```

Q3:

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

int main() {
    int wage = 20;
    cout << "Salary is ";
    cout << wage * 40 * 50;
    cout << endl;

    return 0;
}
```

What is wage?

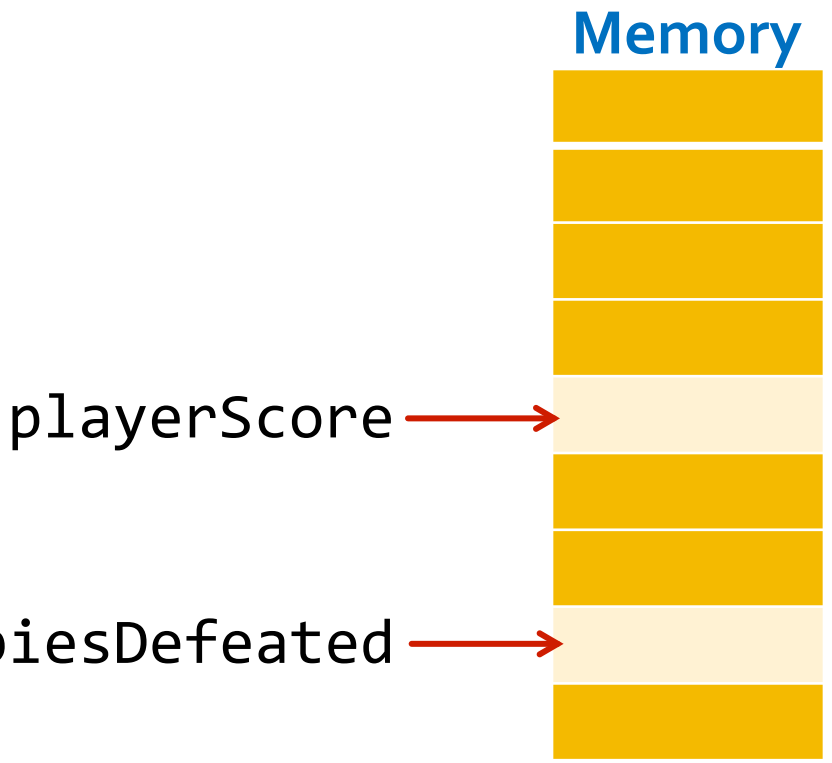
A) is a variable
B) represents a particular memory location
C) holds the value of 20
D) all of the above

Variables

- A variable is a name for something
- Sets up a location in memory
- Stores a value into that location

Variable Declaration

- Give it a name that describes its purpose
- Specify the type of info it will hold
- Examples:
  - `int playerScore;`
  - `int numZombiesDefeated;`



Variable / Identifier Rules

1) Start with a letter or underscore ('_')
2) After the first character, any number of letters, underscores, or digits
3) Can't be a *reserved* word (also known as *key words*)

- Max size 31 chars (most systems)
- C++ is case-sensitive (vName != Vname)
- System identifiers usually start with '_'

Reserved Words

- Words reserved by C++ language
- Have special meanings
- Can't be used as identifiers
  (e.g. can't be used as function names)

Reserved Words

There are 5 reserved words in this code:

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

int main (void)
{
    return 0;
}
```

Reserved Words

There are 5 reserved words in this code:

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

int main (void)
{
    return 0;
}
```

using, namespace, int, void, return

Reserved Words

alignas	class	enum	namespace	return	try
alignof	compl	explicit	new	short	typedef
and	const	export	noexcept	signed	typeid
and_eq	constexpr	extern	not	sizeof	typename
asm	const_cast	false	not_eq	static	union
auto(1)	continue	float	nullptr	static_assert	unsigned
bitand	decltype	for	operator	static_cast	using
bitor	default	friend	or	struct	virtual
bool	delete	goto	or_eq	switch	void
break	do	if	private	template	volatile
case	double	inline	protected	this	wchar_t
catch	dynamic_cast	int	public	thread_local	while
char	else	long	register	throw	xor
char16_t		mutable	reinterpret_cast	true	xor_eq
char32_t					

Reserved Words

alignas	**class**	enum	**namespace**	**return**	try
alignof	compl	explicit	new	short	typedef
and	**const**	export	noexcept	signed	typeid
and_eq	constexpr	extern	not	sizeof	typename
asm	const_cast	**false**	not_eq	static	union
auto(1)	continue	**float**	nullptr	static_assert	**unsigned**
bitand	decltype	**for**	**operator**	**static_cast**	**using**
bitor	**default**	friend	or	**struct**	virtual
bool	delete	goto	or_eq	**switch**	**void**
break	**do**	**if**	**private**	template	volatile
case	**double**	inline	protected	this	wchar_t
catch	dynamic_cast	**int**	**public**	thread_local	**while**
char	**else**	long	register	throw	xor
char16_t		mutable	reinterpret_cast	**true**	xor_eq
char32_t					

All Variables have a *Type*

data types

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

int main() {
    int wage = 20;
    cout << "Salary is ";
    cout << wage * 40 * 50;
    cout << endl;

    return 0;
}
```

Data Types

- `bool`
- `int`
- `double`
- `char`
- `string`

Data Types - boolean

- booleans only have two values
  - `true`
  - `false`

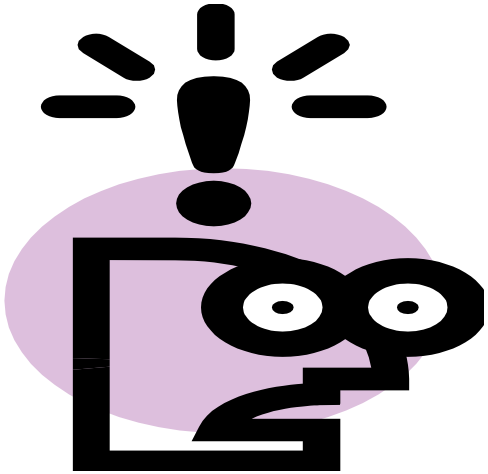
Data Types - int

- Examples: `5, –1, 323, 1000`
- NOT
  - `042` `1,000`

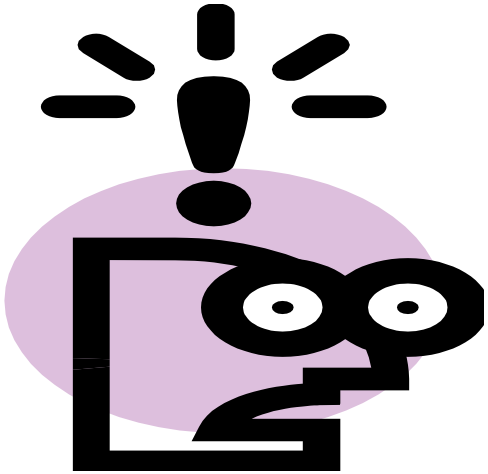
Data Types - `int`

- Modern `int`s are 32-bit
- $2^{32} = 4{,}294{,}967{,}296$
- Range of a C++ `int`:
  - **Signed:** -2,147,483,468 to 2,147,483,467
  - **Unsigned:** 0 to 4,294,967,295

#@/&%
I can't remember those!!!



#@/&%
I can't remember those!!!



How about:

Remember that it's about – to + 2 billion

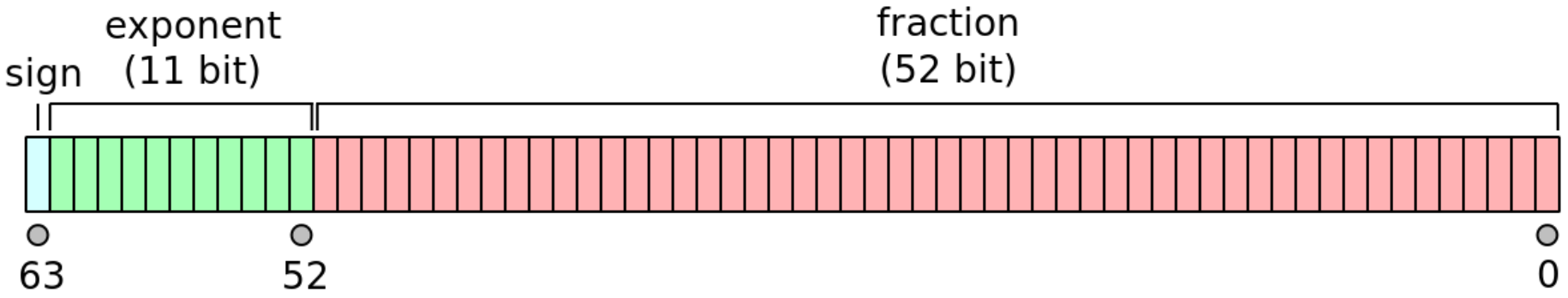
Data Types - double

- Examples:

```
1.0
3.141592
-42.15
```

```
1.32e2
3.0E4
5E-3
```

- Numbers with decimal values



Data Types - double

double

- Size: 8 bytes (64 bits)
- Range: ±2.22507e-308 to

  ±1.79769e+308
- "default" data type for Real numbers
  - Also called "floating point" numbers

Data Types - char

- Examples: 'A' '$' '5' 'e'
- A single character (i.e., letter)

Data Types - `string`

- Examples: "Hello World!"
  "10 of Spades"

- A collection of multiple chars
  - Note the double quotes " to define it

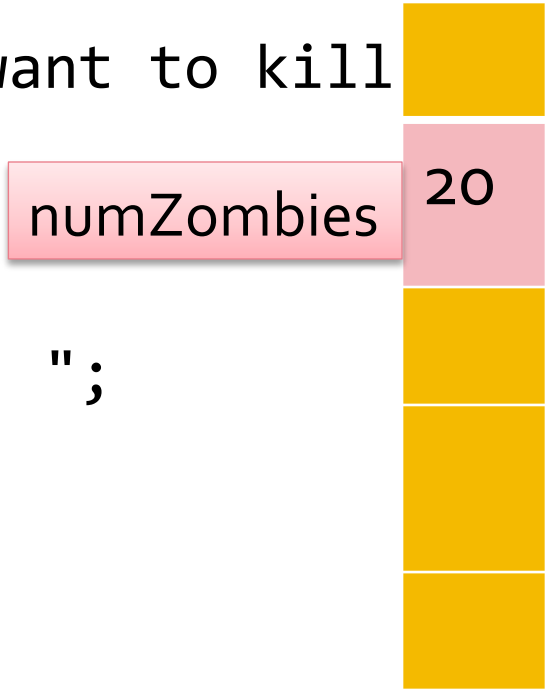
Operators

+ - * / %

```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill? ";
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies – numKilled;

    return 0;
}
```

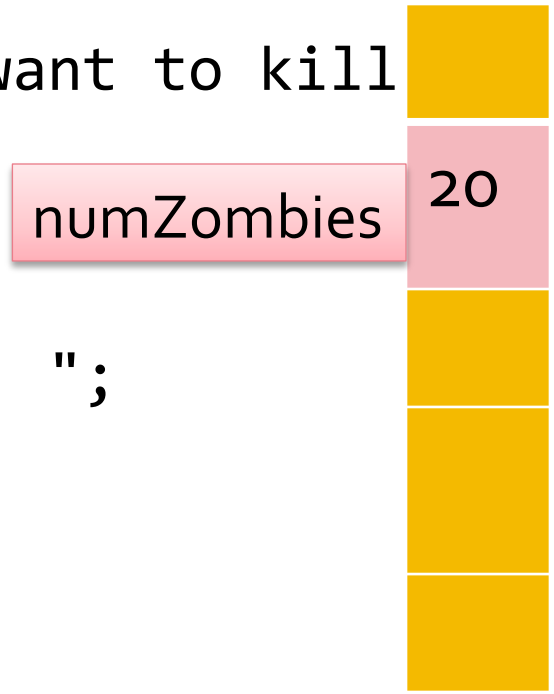
```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies - numKilled;

    return 0;
}
```

Execution



```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies - numKilled;

    return 0;
}
```

Execution

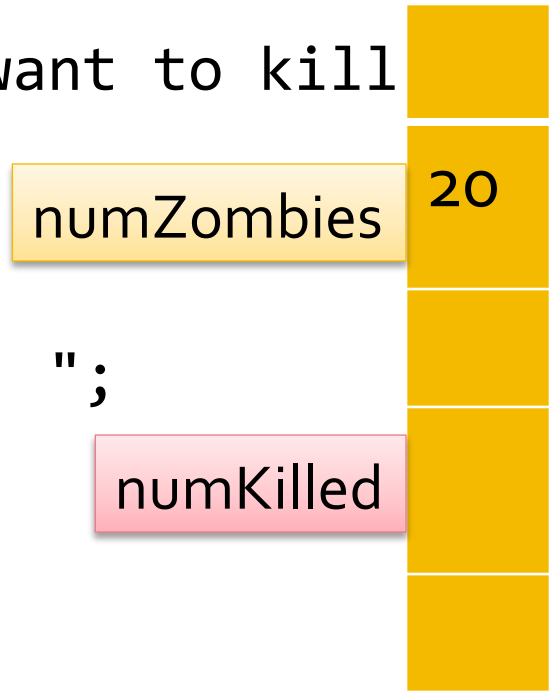


Console

```
Number of Zombies you want  to kill?<Enter>
```

```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies - numKilled;

    return 0;
}
```

Execution

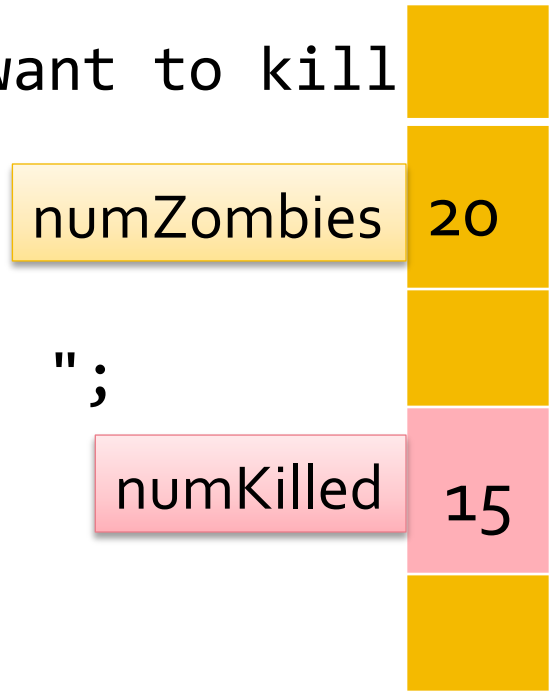


Console

```
Number of Zombies you want  to kill?<Enter>
```

```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies - numKilled;

    return 0;
}
```

Execution

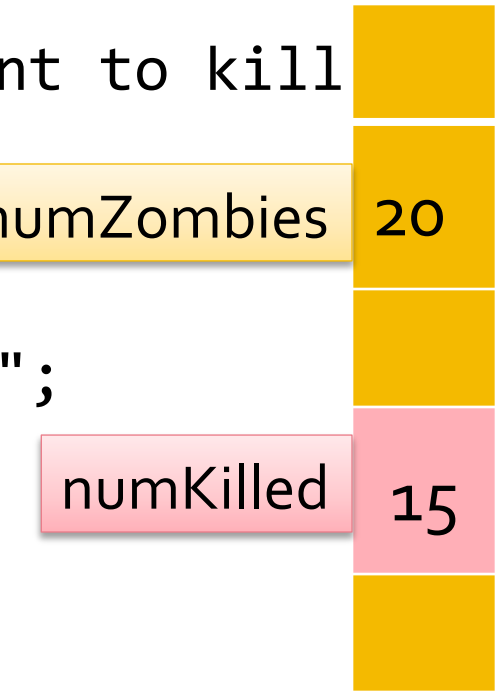


Console

```
Number of Zombies you want  to kill?<E15nter>
```

```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies - numKilled;

    return 0;
}
```

Execution



Console

```
Number of Zombies you want  to kill?<E15nte
Number of zombies left:
```

```
int main() {
    int numZombies = 20;

    cout << "Number of Zombies you want to kill
    int numKilled;
    cin >> numKilled;

    cout << "Number of zombies left: ";
    cout << numZombies - numKilled;

    return 0;
}
```

Execution

Console

```
Number of Zombies you want to kill?<E15nte
Number of zombies left:  5
```

Operators

- In order to compute things, computers need operators
- Some of the most common are mathematical
  - + – Add
  - - – Subtract
  - * – Multiply
  - / – Divide
  - % – Modulo

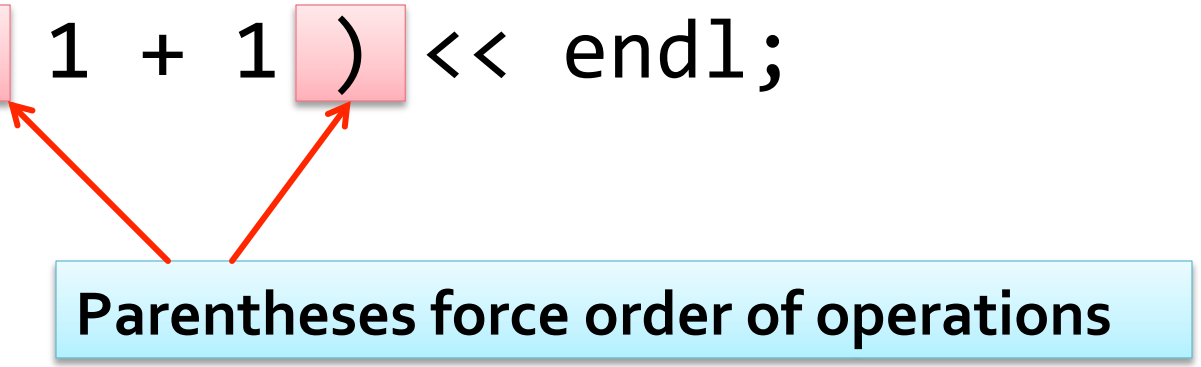
The Addition Operator

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

int main(void)
{
    cout << ( 1 + 1 ) << endl;
    return 0;
}
```

We're just replacing this statement.

The Addition Operator

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

int main(void)
{
    cout << ( 1 + 1 ) << endl;
    return 0;
}
```



The Subtraction Operator

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

int main(void)
{
    cout << ( 5 - 2 ) << endl;
    return 0;
}
```

The Multiplication Operator

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

int main(void)
{
   cout << ( 3 * 4 ) << endl;
   return 0;
}
```

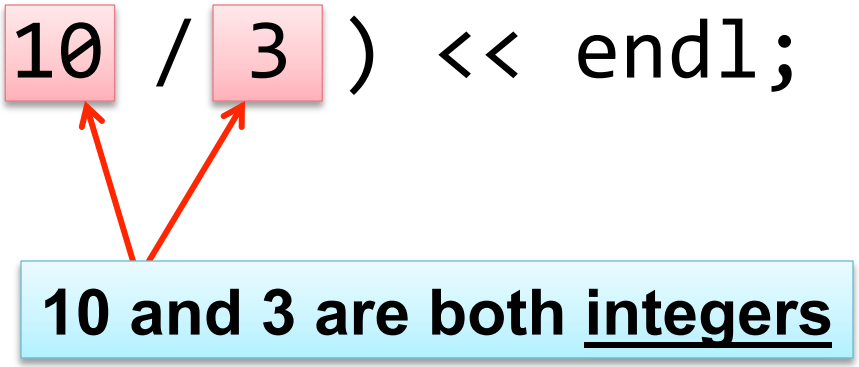
The Division Operator

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

int main(void)
{
    cout << ( 10 / 3 ) << endl;
    return 0;
}
```

Why was the output 3?

The Division Operator

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

int main(void)
{
    cout << ( 10 / 3 ) << endl;
    return 0;
}
```



int Division Always Truncates

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

int main(void)
{
    cout << ( 19 / 10 ) << endl;
    return 0;
}
```

The output is 1.

The Modulo Operator

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

int main(void)
{
    cout << ( 19 % 10 ) << endl;
    return 0;
}
```

Modulo (%) gives you the remainder.

The Modulo Operator

How many hours and minutes in

142 minutes?

The Modulo Operator

How many hours and minutes in

142 minutes?

```
// 60 minutes per hour
int hours = 142 / 60;

// how many minutes are left over
int minutes = 142 % 60;
```

More Complex Output

Combining strings with numbers

More Complex cout

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

int main() {
    cout << "142 minutes is:" << endl
         << ( 142 / 60 ) << " Hours";
    cout << " and" << endl;
    cout << (142 % 60) << " minutes"
         << endl;
    return 0;
}
```

Output:

```
142 minutes is:
2 Hours and
22 minutes
```

Gathering Input

using `cin` for input

Using cin to Get User Info

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

int main() {
    cout << "Enter your name: ";
    string userString;
    cin >> userString;
    cout << "Hello, " << userString << ".";
    cout << endl;
    return 0;
}
```

Using cin to Get User Info

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

int main() {
    cout << "Enter your name: ";
    string userString;
    cin >> userString;
    cout << "Hello, " << userString << ".";
    cout << endl;
    return 0;
}
```

Declares variable *userString*

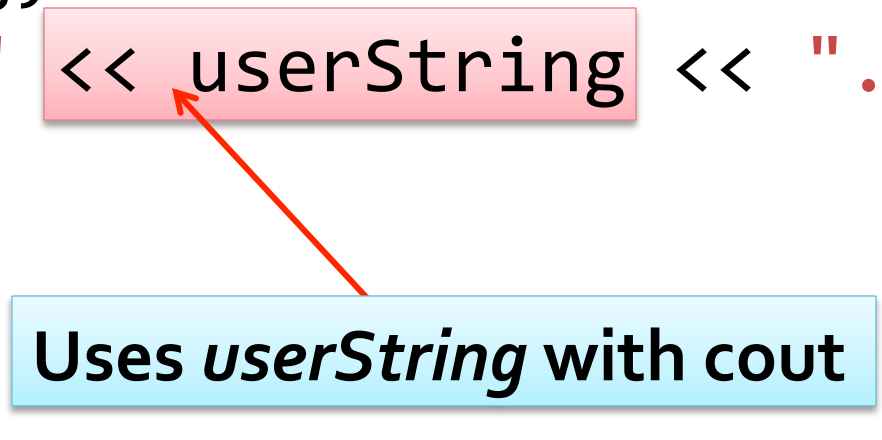
Using cin to Get User Info

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

int main() {
    cout << "Enter your name: ";
    string userString;
    cin >> userString;
    cout << "Hello, " << userString << ".";
    cout << endl;
    return 0;
}
```

Uses cin to define *userString*

Using cin to Get User Info

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

int main() {
    cout << "Enter your name: ";
    string userString;
    cin >> userString;
    cout << "Hello, " << userString << ".";
    cout << endl;
    return 0;
}
```



Next Time on EECS 183…

More details on: Variables and Data Types
Casting from One Type to Another
Imprecision, Compile and run-time errors
Libraries, Testing and Debugging
Pre-defined functions, e.g., cmath

Remember!

Bring your i>Clicker **next lecture**

CodeLab is due **this Friday**

Discussion Sections start this week!