CS105C: C++ Programming

Why C++?

Why Python?

```python
print("Hello World!")

evens = [ x for x in range(1,10) if x % 2 == 0]
```

Why C++?

```
template <class T, class Allocator BOOST_CONTAINER_DOCONLY(= new_allocator<T>),
          class Options BOOST_CONTAINER_DOCONLY(= void) >
class vector
{
   #ifndef BOOST_CONTAINER_DOXYGEN_INVOKED

   typedef typename boost::container::allocator_traits<Allocator>::size_type  alloc_size_type;
   typedef typename get_vector_opt<Options, alloc_size_type>::type            options_type;
   typedef typename options_type::growth_factor_type                          growth_factor_type;
   typedef typename options_type::stored_size_type                            stored_size_type;
   typedef value_less<T>                                                      value_less_t;

   //If provided the stored_size option must specify a type that is equal or a type that is smaller.
   BOOST_STATIC_ASSERT( (sizeof(stored_size_type) < sizeof(alloc_size_type) ||
                        dtl::is_same<stored_size_type, alloc_size_type>::value) );
```

...eek.

Why C++?

```cpp
int add(int* a){
    return a[0] + a[1] * a[4];
}
```

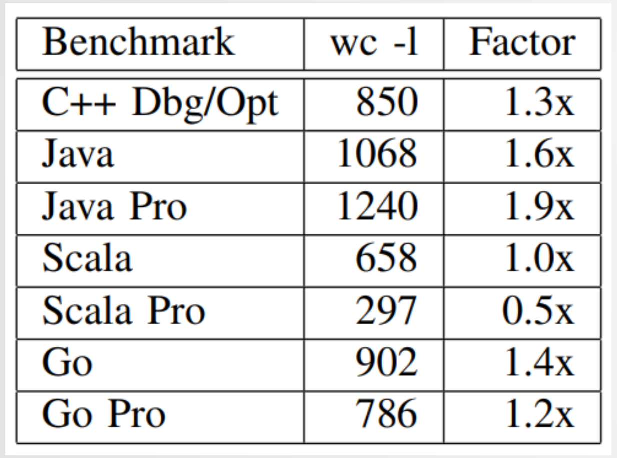
```asm
        movl    4(%rdi), %eax
        imull   16(%rdi), %eax
        addl    (%rdi), %eax
        ret
```

```cpp
template <typename T>
T add(std::vector<T>& a){
    auto x = *a.begin()
    auto y = ++a.begin()
    auto z = a[4];
    return x + *y * z;
}
```

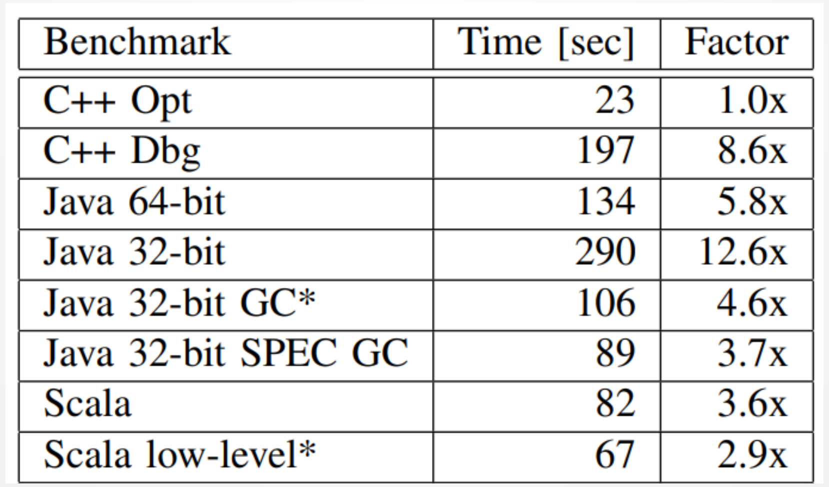
```asm
        movq    (%rdi), %rdx
        movl    4(%rdx), %eax
        imull   16(%rdx), %eax
        addl    (%rdx), %eax
        ret
```

Why C++?

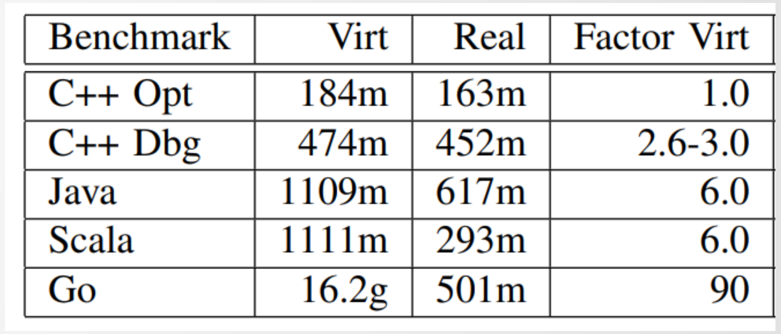
C++ is the ***only*** language used across many different disciplines that offers the user fine-grained control over how high-level abstractions map into low-level operations*



Benchmark	wc -l	Factor
C++ Dbg/Opt	850	1.3x
Java	1068	1.6x
Java Pro	1240	1.9x
Scala	658	1.0x
Scala Pro	297	0.5x
Go	902	1.4x
Go Pro	786	1.2x



Benchmark	Time [sec]	Factor
C++ Opt	23	1.0x
C++ Dbg	197	8.6x
Java 64-bit	134	5.8x
Java 32-bit	290	12.6x
Java 32-bit GC*	106	4.6x
Java 32-bit SPEC GC	89	3.7x
Scala	82	3.6x
Scala low-level*	67	2.9x



Benchmark	Virt	Real	Factor Virt
C++ Opt	184m	163m	1.0
C++ Dbg	474m	452m	2.6-3.0
Java	1109m	617m	6.0
Scala	1111m	293m	6.0
Go	16.2g	501m	90

Median line count program...

...between 3x and 13x faster than competitors...

...while using less than 1/3rd the memory

*There are enough asterisks on this statement to fill a star chart--don't take this literally.

Overview

A 1-unit **accelerated** introduction to the C++ language.

Topics covered:

- Syntax
- IOStreams
- Objects, Inheritance, Polymorphism
- C++ Templates
- Memory management
- Compilation Model
- Undefined Behavior 😱
- STL containers/algorithms
- Namespaces
- RAII
- ....

Overview

A 1-unit **accelerated** introduction to the C++ language.

Topics covered:

- Syntax
- IOStreams
- Objects, Inheritance, Polymorphism $\rightarrow$ CS 371p
- C++ Templates $\rightarrow$ CS 371g
- Memory management
- Compilation Model
- Undefined Behavior 😱 $\leftarrow$ Has been known to confuse the Linux kernel developers
- STL containers/algorithms $\leftarrow$ Another course?
- Namespaces
- RAII $\leftarrow$ An entirely different language...
- ....

So why are we here?

Goals

- Give you a starting point for continued study of C++
- Understand what features the language gives you
- Have an idea of *why* certain language elements have the designs that they do
- Know how to use these language features to model problems

Warning: this class will be *packed*

Course Info: Resources

- Lecture: Monday, 4:00-5:00, GDC 5.302
- Office Hours: Monday, 5:00 - 6:30, GDC 5.416 (+other)
- Website: cs.utexas.edu/~theksong/teaching/cs105c/f2019/
- Piazza: piazza.com/class/jwk5azq3ef2nk
- Canvas: canvas.utexas.edu/
- Instructor email: theksong@cs.utexas.edu

Course Info: Grading

See syllabus for further details

Projects	**70%**
Quizzes	**20%**
Participation	**10%**

Projects

- Four projects to practice working on larger codebases.
- ***Exponentially*** escalating difficulty!!
- Three slip days distributed across projects: 25% per-day penalty assessed after slip days are gone.
- Specific rules must be followed--these will be posted on the website and linked to by the project handouts.
- Bonus sections can be completed for extra credit. Talk to me if you want to try them but don't know how!

Quizzes

- In-class, 5 minutes
- Issued on Canvas, access code will be given out in lecture.
- Recommended: bring a device with a keyboard.
- Every other week(?) or so

Participation

- When entering lecture, grab an index card
- Turn it in at the end of lecture with the following:
  - One thing you understand/know now that you didn't before.
  - One question you have about the lecture material.
  - Your name + EID

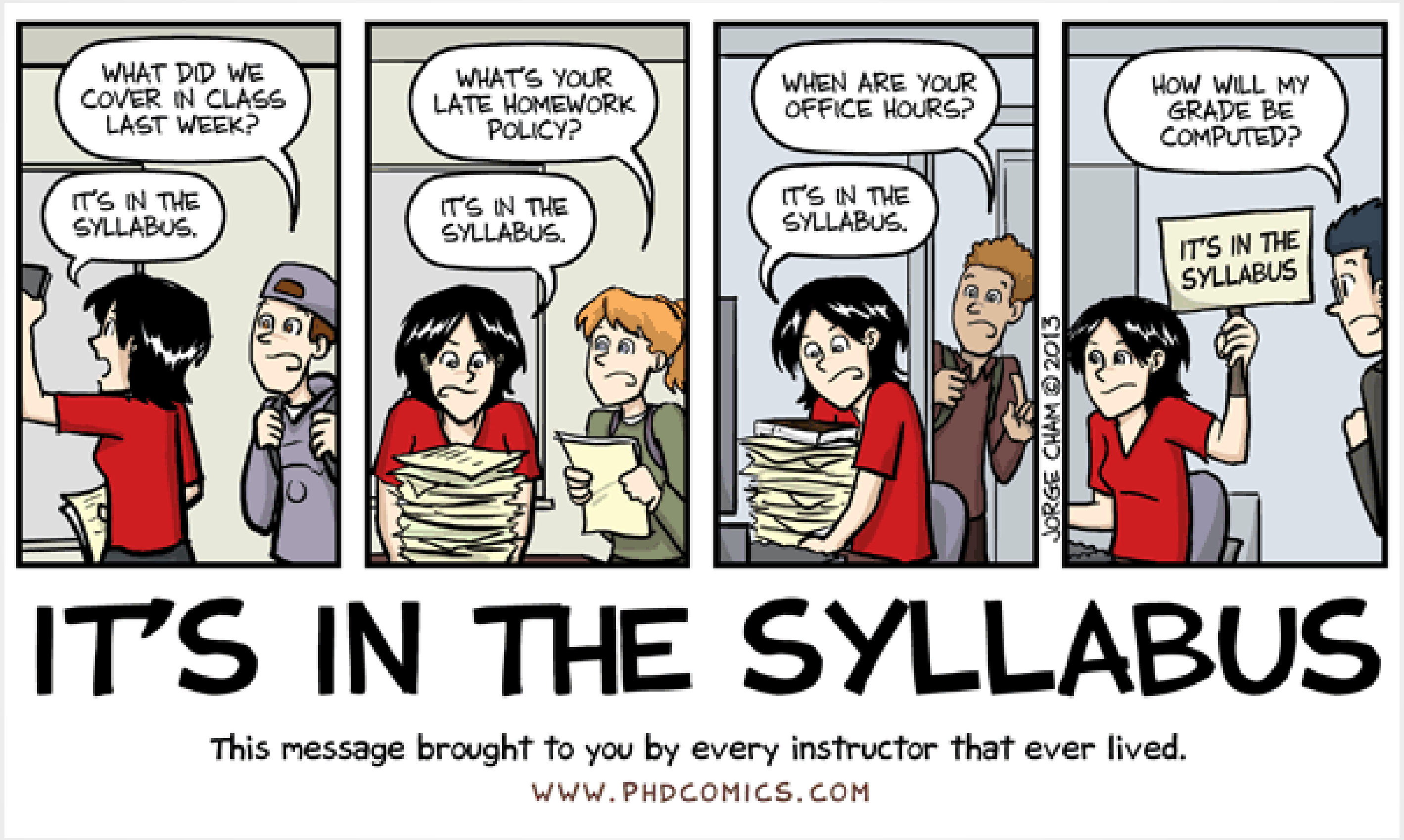
Course Info: Cheating

- Projects:
  - You may talk to other students to get high level ideas
  - Do not take notes or view code when discussing these ideas
  - Afterwards, spend at least 30 minutes doing something completely different before programming
  - Do not reuse ***any*** code without permission
- If you have any questions, *ask me*.

Minimum dishonesty penalty:

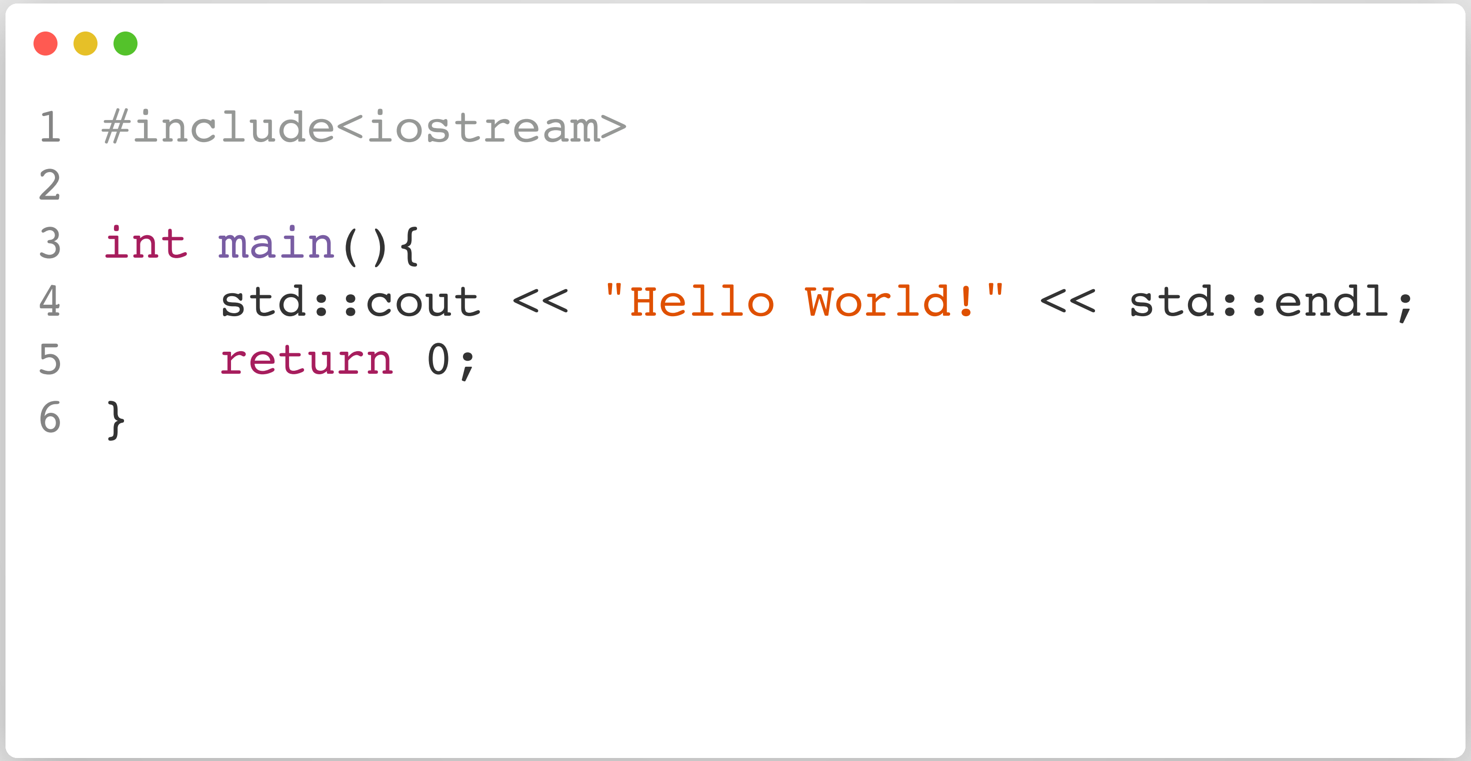
- 0 on the assignment
- Additional reduction in final grade
- Referral to Dean of Students

Course Info: Other


WHAT DID WE COVER IN CLASS LAST WEEK?
IT'S IN THE SYLLABUS.
WHAT'S YOUR LATE HOMEWORK POLICY?
IT'S IN THE SYLLABUS.
WHEN ARE YOUR OFFICE HOURS?
IT'S IN THE SYLLABUS.
JORGE CHAM © 2013
HOW WILL MY GRADE BE COMPUTED?
IT'S IN THE SYLLABUS
IT'S IN THE SYLLABUS
This message brought to you by every instructor that ever lived.
WWW.PHDCOMICS.COM

*If you're sure it's not, email me!

C++

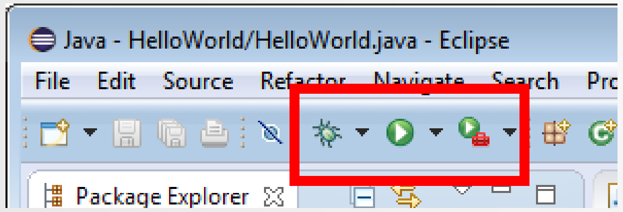


```cpp
#include<iostream>

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

How do I run it?

Python or Java IDE:
One step

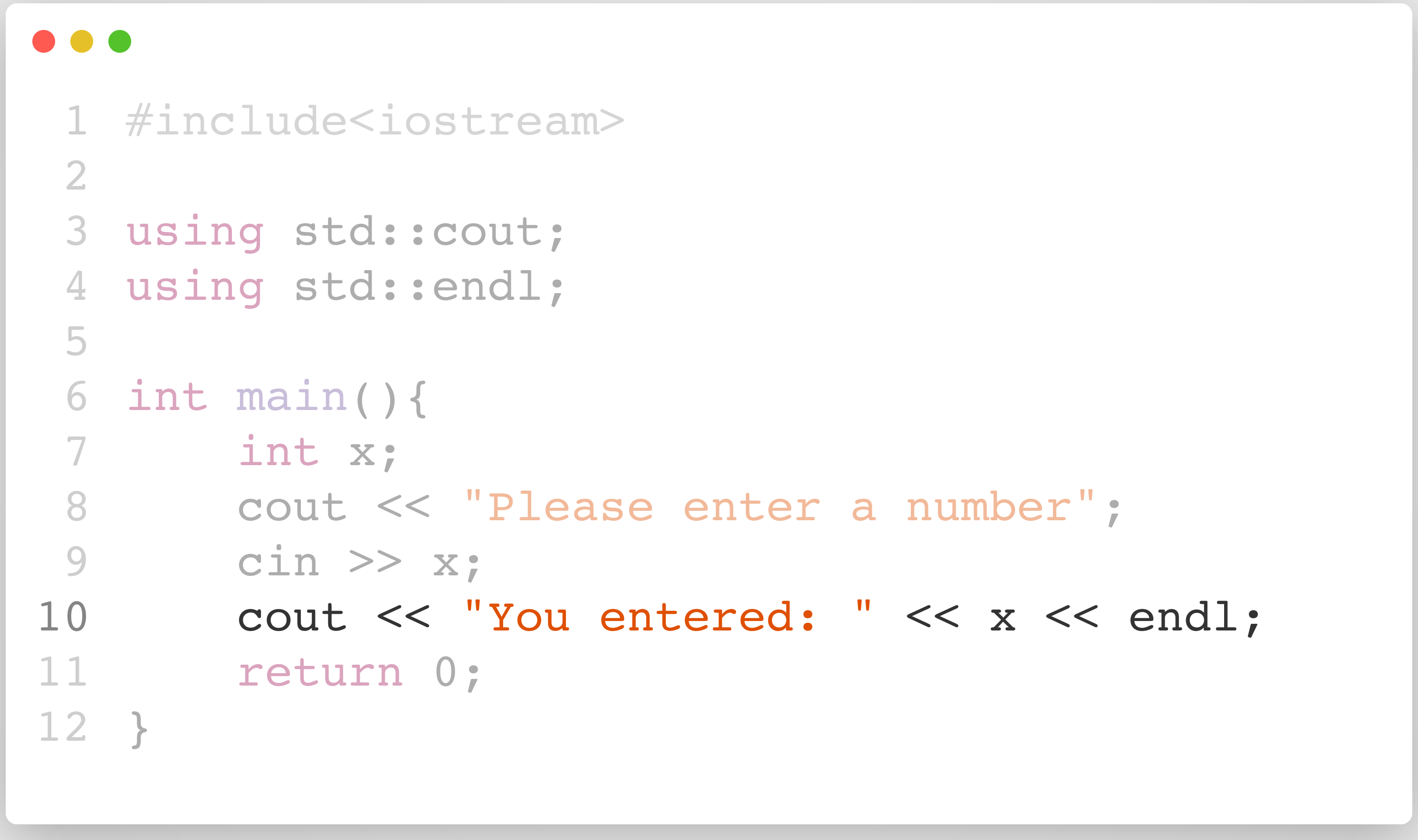
```
python helloworld.py
```


Java - HelloWorld/HelloWorld.java - Eclipse
File Edit Source Refactor Navigate Search Pro
Package Explorer

C++: Two Step

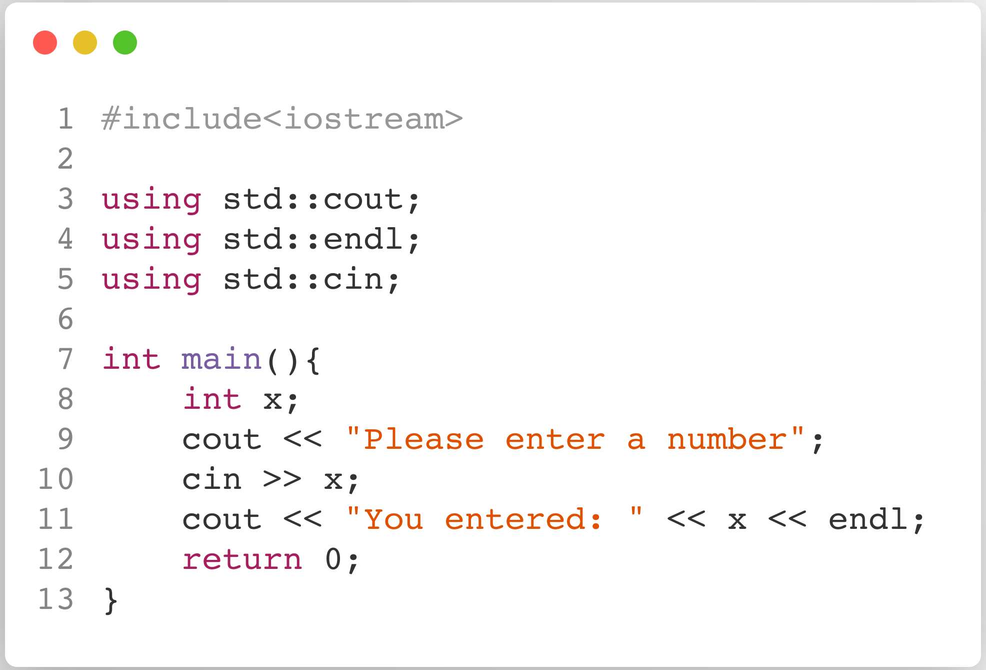
```
Instructor@CS105C
> 
```

A Minor Twist

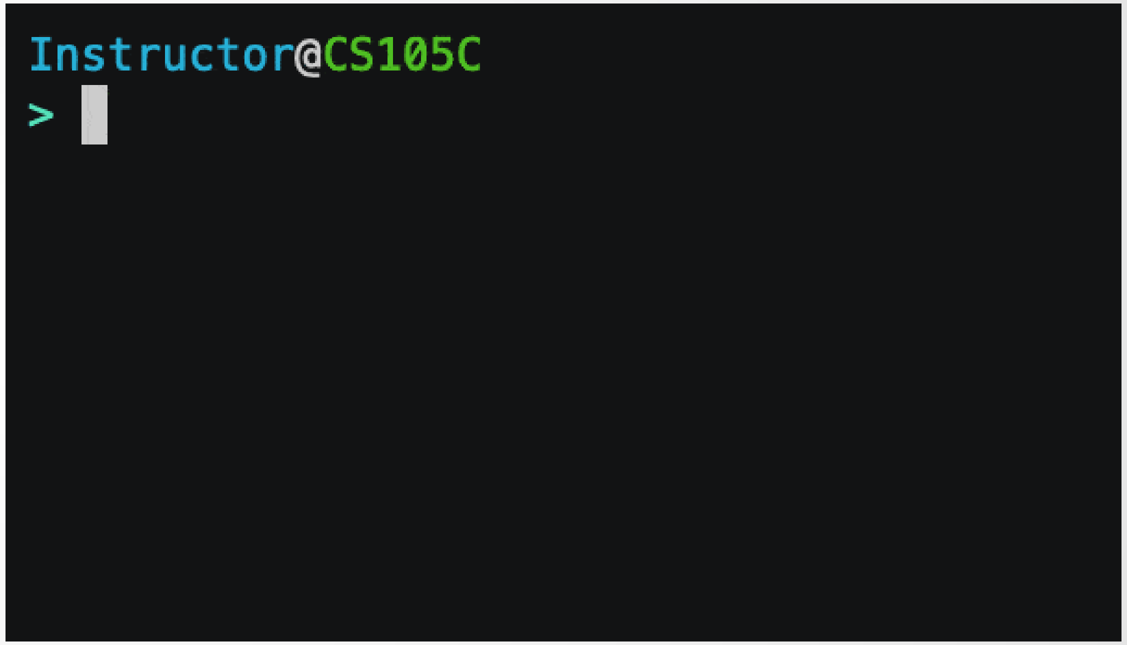


```cpp
#include<iostream>

using std::cout;
using std::endl;

int main(){
    int x;
    cout << "Please enter a number";
    cin >> x;
    cout << "You entered: " << x << endl;
    return 0;
}
```

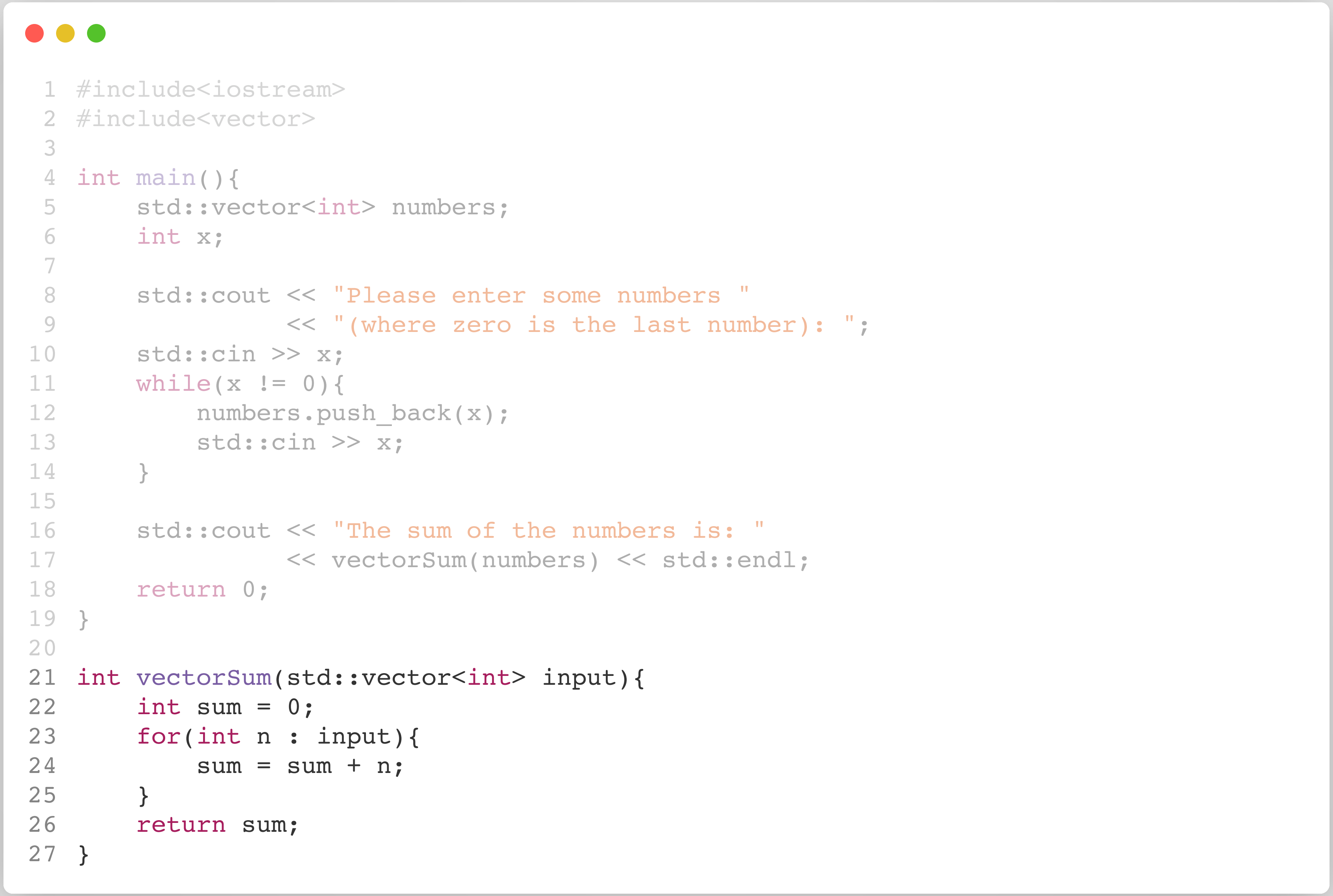
A Minor Twist



```cpp
#include<iostream>

using std::cout;
using std::endl;
using std::cin;

int main(){
    int x;
    cout << "Please enter a number";
    cin >> x;
    cout << "You entered: " << x << endl;
    return 0;
}
```



```
Instructor@CS105C
>
```

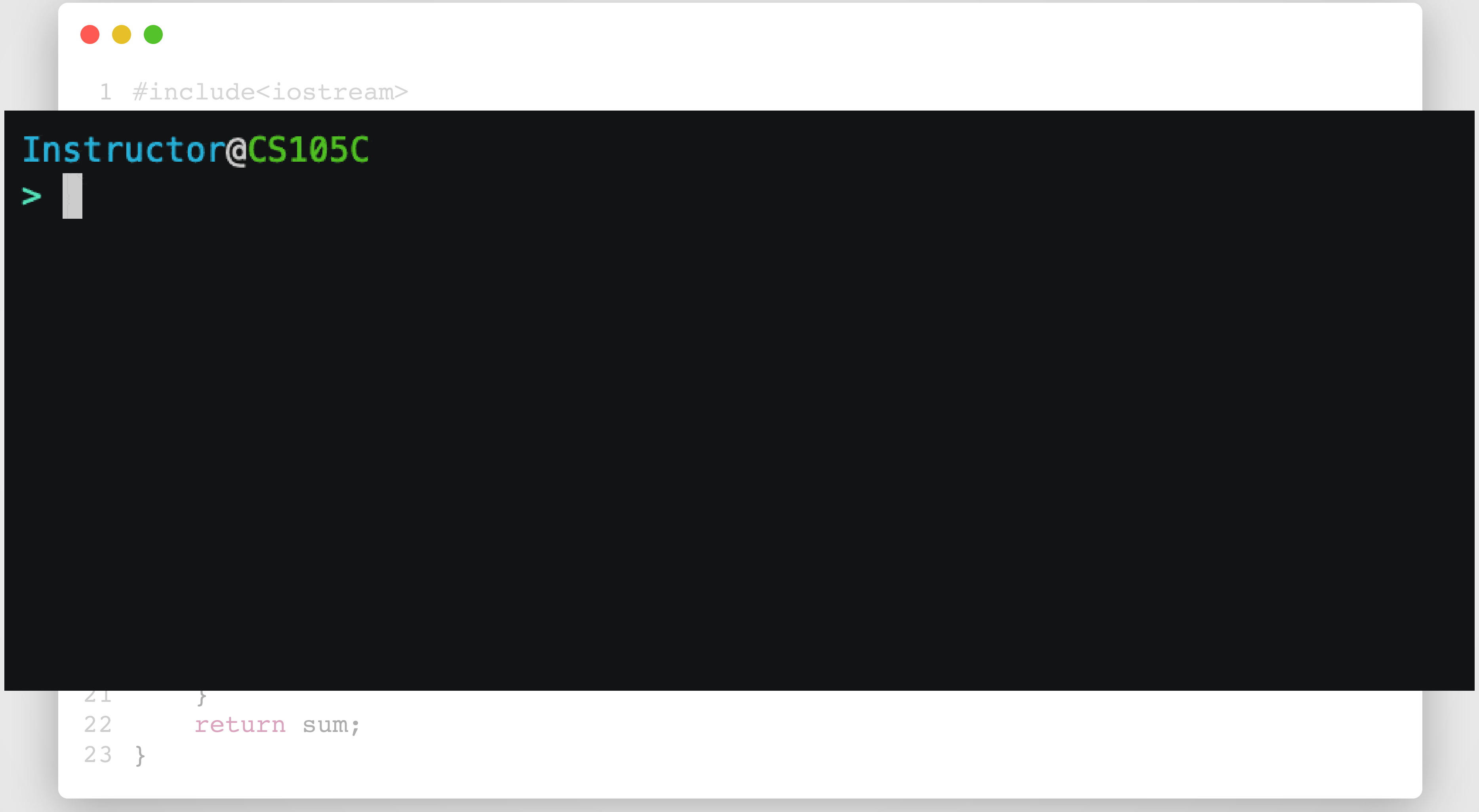


```cpp
#include<iostream>
#include<vector>

int main(){
    std::vector<int> numbers;
    int x;

    std::cout << "Please enter some numbers "
              << "(where zero is the last number): ";
    std::cin >> x;
    while(x != 0){
        numbers.push_back(x);
        std::cin >> x;
    }

    std::cout << "The sum of the numbers is: "
              << vectorSum(numbers) << std::endl;
    return 0;
}

int vectorSum(std::vector<int> input){
    int sum = 0;
    for(int n : input){
        sum = sum + n;
    }
    return sum;
}
```

Great, let's compile it!

```
Instructor@CS105C
> 
```

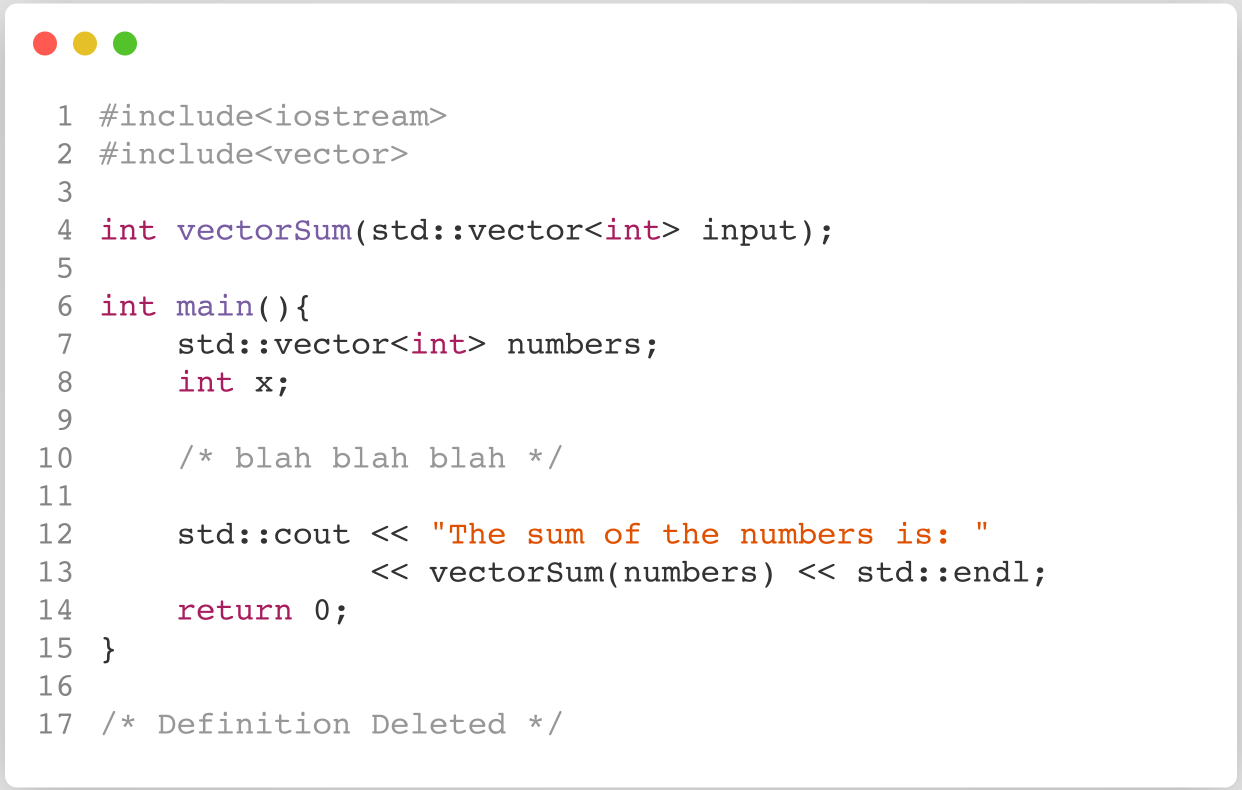
...dang.

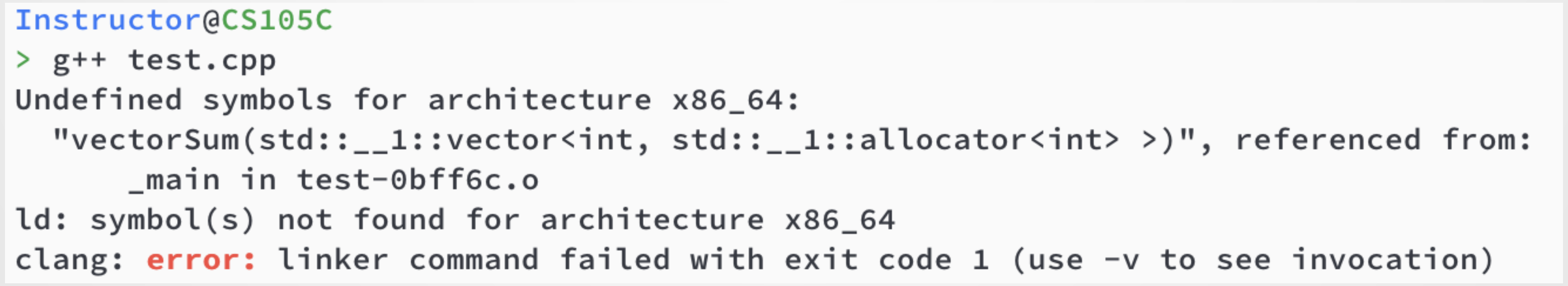


```
#include<iostream>
```

```
Instructor@CS105C
> 
```

```
    }
    return sum;
}
```


```cpp
#include<iostream>
#include<vector>

int vectorSum(std::vector<int> input);

int main(){
    std::vector<int> numbers;
    int x;

    /* blah blah blah */

    std::cout << "The sum of the numbers is: "
              << vectorSum(numbers) << std::endl;
    return 0;
}

/* Definition Deleted */
```


```
Instructor@CS105C
> g++ test.cpp
Undefined symbols for architecture x86_64:
  "vectorSum(std::__1::vector<int, std::__1::allocator<int> >)", referenced from:
      _main in test-0bff6c.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

Function Declaration

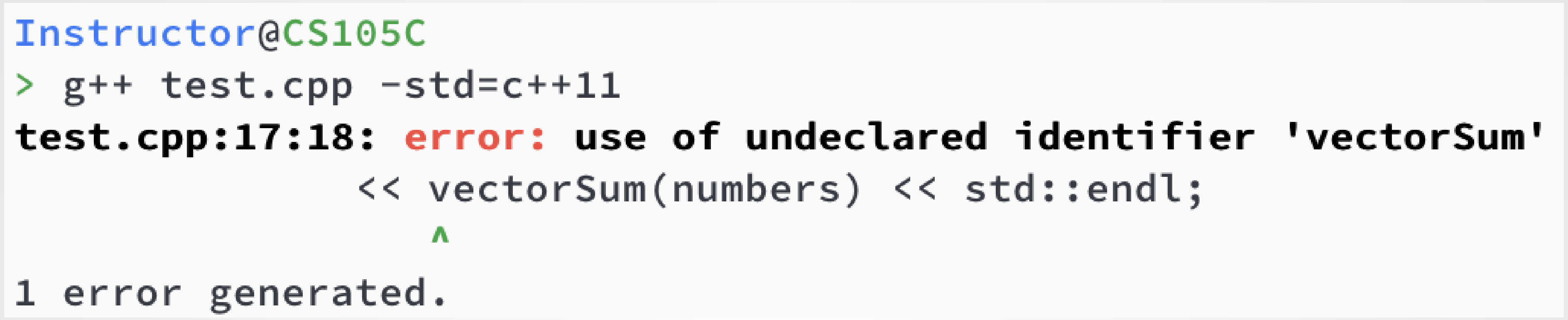
"Here is *how* you use the function."

```cpp
int vectorSum(std::vector<int> input);
```

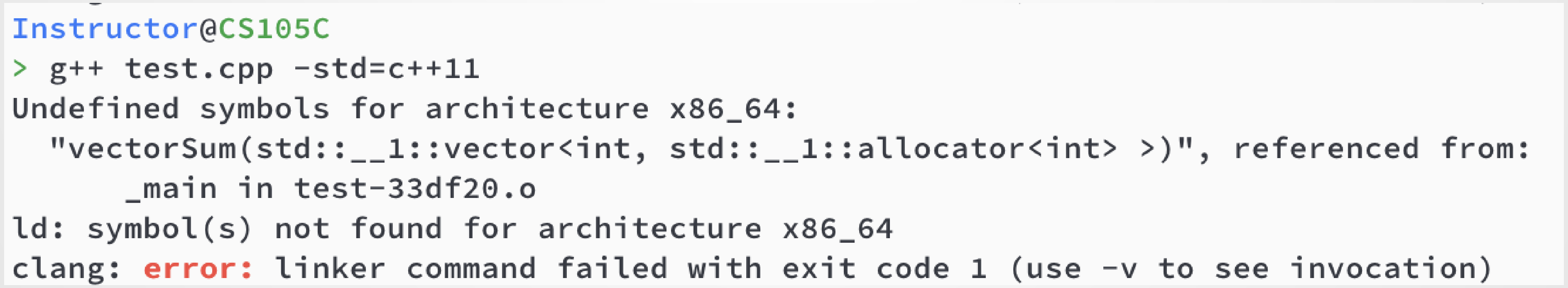
Function Definition

"This is what happens when you call the function."

```cpp
int vectorSum(std::vector<int> input){
    int sum = 0;
    for(int n : input){
        sum = sum + n;
    }
    return sum;
}
```



```
Instructor@CS105C
> g++ test.cpp -std=c++11
test.cpp:17:18: error: use of undeclared identifier 'vectorSum'
       << vectorSum(numbers) << std::endl;
          ^
1 error generated.
```



```
Instructor@CS105C
> g++ test.cpp -std=c++11
Undefined symbols for architecture x86_64:
  "vectorSum(std::__1::vector<int, std::__1::allocator<int> >)", referenced from:
      _main in test-33df20.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

Why though?

- 6' 5" tall
- 115 kW of power draw (average U.S. home is ~1kW)
- 5.5 tons
- $7.9 million ($37.6m in 2019)

- CPU: 80MHz (single core SIMD)
- 8.39 MB of main memory
- 303 MB of main storage
- 160 MFLOPS

iPhone XS: 10-50 GFLOPS

What if...?

```cpp
int vectorSum(std::vector<int> input){
    int sum;  // Oops.
    for(int n : input){
        sum = sum + n;
    }
    return sum;
}
```

Recap

A demo of how simple C++ programs work.

How to use some C++ language features:

- Namespaces (std::)
- Templates (vector<int>)
- User-defined functions

Our first distinctions between similar operations:

- Function **declaration** *vs* **definition**

Miniproject

Out on Canvas

Due next week at start of lecture.

Designed to help you understand a little about the compilation process of C++ and working with errors.

Must compile with no warnings or errors on the basement lab machines.

Ask questions on Piazza (especially for Problem 1!)

Index Cards!

- Your name + EID
- One thing you understand/know now that you didn't before.
- One question you have about the lecture material.

Appendix

Why are we here?

Think about when you watch a movie.

Protagonist is having a hard time at their job, wants to quit. They have a fight with their best friend, where their friend urges them to try harder at the job. Then their friend is killed by the Mafia.

What are reasonable outcomes?

- Protagonist goes on a revenge-fueled killing spree against Mafia
- Protagonist returns to work promising to try harder for their friend
- Protagonist quits their job to travel the world and self-introspect
- Protagonist descends into drug-fueled depression

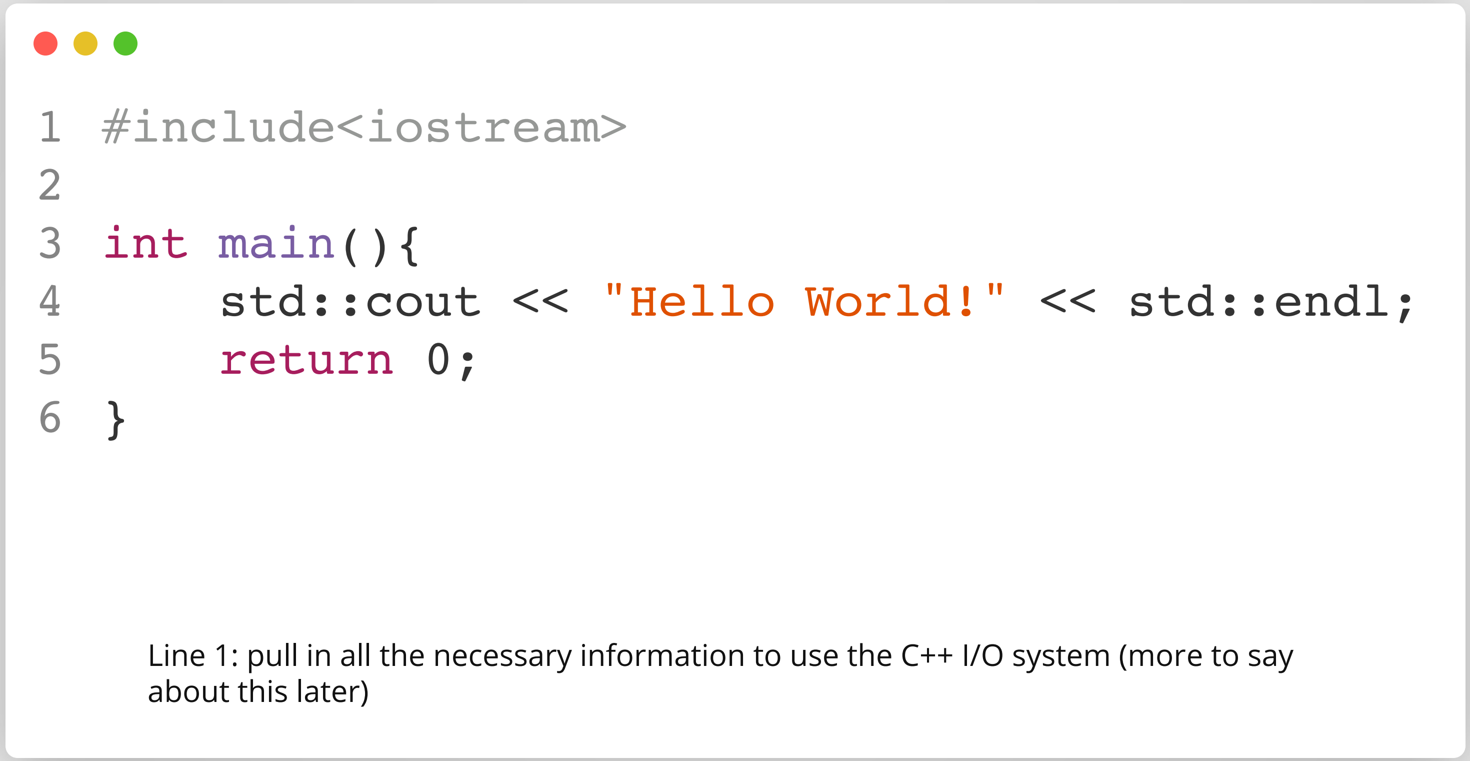
But if you saw the protagonist marry a brick wall and then take a job as a walrus at SeaWorld Moscow....you would ask what the writers were smoking.

Same deal here: we want to learn the general shape of C++: what makes sense, what doesn't and where we can look if we get confused

```java
String s1 = new String("haha");
String s2 = new String("haha");

if(s1 == s2){
    System.out.println("equal");
}
else{
    System.out.println("not equal");
}
```

What does "equal" mean? Does it mean they have the same value or that they're literally the same object? (Are two identical twins "equal" to each other?)

The designers had to pick one meaning: they went with same-object equality and used .equals() for value equality. Not a bad choice, but one we have to be aware of.

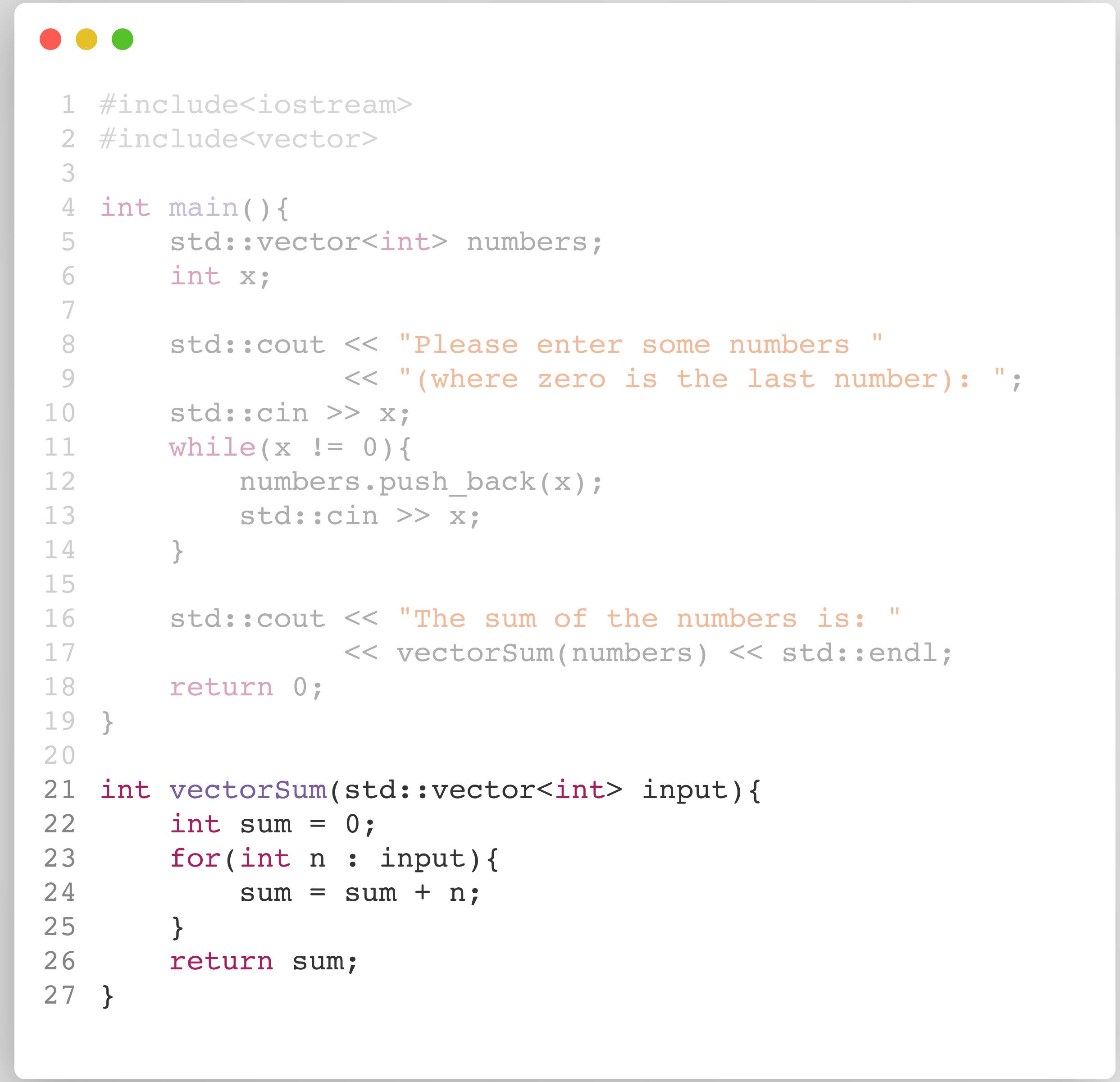


```
1 #include<iostream>
2
3 int main(){
4     std::cout << "Hello World!" << std::endl;
5     return 0;
6 }
```

Line 1: pull in all the necessary information to use the C++ I/O system (more to say about this later)

Line 3: Indicate the start of the main function. Every C++ program must have one (and only one!) main function. Program execution starts in main()

Line 4: Print "Hello World!" to std::cout (which usually represents the terminal)



```
 1 #include<iostream>
 2 #include<vector>
 3 
 4 int main(){
 5     std::vector<int> numbers;
 6     int x;
 7 
 8     std::cout << "Please enter some numbers "
 9               << "(where zero is the last number): ";
10     std::cin >> x;
11     while(x != 0){
12         numbers.push_back(x);
13         std::cin >> x;
14     }
15 
16     std::cout << "The sum of the numbers is: "
17               << vectorSum(numbers) << std::endl;
18     return 0;
19 }
20 
21 int vectorSum(std::vector<int> input){
22     int sum = 0;
23     for(int n : input){
24         sum = sum + n;
25     }
26     return sum;
27 }
```

Program that reads in a bunch of input (until we get zero), then adds all the entered numbers.

Line 6: Declare a variable. Type int, name x.

Line 8: Prompt user for input (note: no newline)

Line 10-15: read in X, until x is zero. Note that cin automatically handles argument splitting on whitespace and type conversion for us (this may be annoying later on).

Line 16: compute vectorSum and print result

Line 21: Define vectorSum function.

Line 23: Range-based for loop--similar to for..in in Python or foreach loop in Java.

NOTE: Code does not compile due to missing vectorSum declaration before main().