Our First Class – Cube:

**Cube.h**

```cpp
#pragma once

class Cube {
public:
    double getVolume();
private:
};
```

**Cube.cpp**

```cpp
#include "Cube.h"

double Cube::getVolume() {
    return length_ * length_ * length_;  
}
```

Public vs. Private:

<table>
<thead>
<tr>
<th>Situation</th>
<th>Protection Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cube functionality provided to client code</td>
<td></td>
</tr>
<tr>
<td>Variable containing data about the Cube</td>
<td></td>
</tr>
<tr>
<td>Helper function used in Cube</td>
<td></td>
</tr>
</tbody>
</table>

Hierarchy in C++:

There `Cube` class we’re building might not be the only `Cube` class. Large libraries in C++ are organized into ____________________________.

Our First Program:

**main.cpp**

```cpp
#include "Cube.h"
#include <iostream>

int main() {
    cs225::Cube c;
    std::cout << "Volume: " << c.getVolume() << std::endl;
    return 0;
}
```

...run this yourself: run `make` and `./main` in the lecture source code.

Several things about C++ are revealed by our first program:

1. ____________________________
   main.cpp:4

2. ____________________________
   main.cpp:5, main.cpp:1

3. ____________________________
   main.cpp:6, main.cpp:2

4. However, our program is unreliable. **Why?**

Default Constructor:

Every class in C++ has a constructor – even if you didn’t define one!

- Automatic/Implicit Default Constructor:

- Custom Default Constructor:

```cpp
#include "Cube.h"

class Cube {
public:
    Cube();
    Cube(double);
};
```

```cpp
class Cube {
public:
    Cube() { 
        length_ = 1;
    }
};
```
Custom, Non-Default Constructors:
We can provide also create constructors that require parameters when initializing the variable:

<table>
<thead>
<tr>
<th>Cube.h</th>
<th>Cube.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 class Cube {</td>
<td>3 Cube::Cube(double length) {</td>
</tr>
<tr>
<td>5 public:</td>
<td>4</td>
</tr>
<tr>
<td>6 Cube(double length);</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>6 }</td>
</tr>
</tbody>
</table>

Puzzle #1: How do we fix our first program?

```cpp
puzzle.cpp w/ above custom constructor

8 cs225::Cube c;
9 cout << "Volume: " << c.getVolume() << endl;
```

...run this yourself: run make puzzle and ./puzzle in the lecture source code.

Solution #1:

Solution #2:

*The beauty of programming is both solutions work! There's no one right answer; both have advantages and disadvantages!*

Reference Variable

A reference variable is an alias to an existing variable. Modifying the reference variable modifies the variable being aliased. Internally, a reference variable maps to the same memory as the variable being aliased:

```cpp
main-ref.cpp

3 int main() {
4     int i = 7;
5     int & j = i;  // j is an alias of i
6     j = 4;       // j and i are both 4.
7     std::cout << i << " " << j << std::endl;
8     i = 2;       // j and i are both 2.
9     std::cout << i << " " << j << std::endl;
10    return 0;
11 }
```

...run this yourself: run make and ./main-ref in the lecture source code.

Three things to note about reference variables:

1. _______________________
2. _______________________
3. _______________________

Pointers and References – Introduction

A major component of C++ that will be used throughout all of CS 225 is the use of references and pointers. References and pointers both:

- Are extremely powerful, but extremely dangerous.
- Pointers are *level of indirection* via memory to our data.

As a level of indirection via memory to the data:

1. _______________________
2. _______________________

CS 225 – Things To Be Doing:

1. Sign up for “Exam 0” (exam starts Thursday, Sept. 6th)
2. Attend lab and complete lab_intro; due Sunday, Sept. 2nd
3. MP1 released Friday; due Monday, Sept. 10th
4. Visit Piazza and the course website often!