Creating New Types
In data structures, we will be learning and creating new types of structures to store data. We will start simply – by the end, we will have types we built being the building blocks for new types!

Big Idea: Encapsulation

Encapsulation principles:

<table>
<thead>
<tr>
<th>Cube.h</th>
<th>Cube.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Our First Class – Cube:

<table>
<thead>
<tr>
<th>Cube.h</th>
<th>Cube.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>main.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

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

...run this yourself: run make and ./main in the lecture source code.
Our First Program:

```
main.cpp
1 #include "Cube.h"
2 #include <iostream>
3 4 int main() {
5   cs225::Cube c;
6   std::cout << "Volume: " << c.getVolume() << std::endl;
7   return 0;
8 }
```

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

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:

```
Cube.h
4 class Cube {
5   public:
6   Cube();
7   /* ... */
Cube.cpp
3 Cube::Cube() {
4 }
```

Custom, Non-Default Constructors:
We can provide also create constructors that require parameters when initializing the variable:

```
Cube.h
4 class Cube {
5   public:
6   Cube(double length);
7   /* ... */
Cube.cpp
3 Cube::Cube(double length) { 
4 }
```

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

```
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!**

CS 225 – Things To Be Doing:

1. Attend lab and complete lab_intro; due Sep. 5th
2. MP1 released Tomorrow; due Monday, Sep. 6th
3. Visit Piazza and the course website often!
4. Join us on Discord