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:

Cube.h	Cube.cpp
1	1
2	2
3	3
4	4
5	5

Our First Class – Cube:

Cube.h	Cube.cpp
1 `#pragma once`	1 `#include "Cube.h"`
2	2
3 `class Cube {`	3 `double Cube::getVolume() {`
4 `  public:`	4
5 `    double getVolume();`	5
6	6 `}`
7	7
8	8
9	9
10	10
11 `  private:`	11
12	12
13	13
14	14
15	15
16 `};`	16

Public vs. Private:

Situation	Protection Level
Cube functionality provided to **client code**	
Variable containing data about the **Cube**	
Helper function used in **Cube**	

Hierarchy in C++:

There **Cube** class we're building might not be the only **Cube** class. Large libraries in C++ are organized into _____.

Cube.h	Cube.cpp
1 `#pragma once`	1 `#include "Cube.h"`
2	2
3 `namespace cs225 {`	3 `namespace cs225 {`
4 `  class Cube {`	4 `  double` `    Cube::getVolume() {`
5 `  public:`	5 `    return length_ *`
6 `    double getVolume();`	`            length_ * length_;`
7	6 `  }`
…	…

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.

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`

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

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

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

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