#2: Classes and Reference Variables

January 16, 2019 · *Zilles, Fagen-Ulmschneider*

Our First Class – Cube:

Cube.h

```
#pragma once

class Cube {
  public:
   double getVolume();





  private:




};
```

Cube.cpp

```
#include "Cube.h"

double Cube::getVolume() {


}
```

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

```
#pragma once

namespace cs225 {
  class Cube {
  public:
    double getVolume();

...
```

Cube.cpp

```
#include "Cube.h"

namespace cs225 {
  double
    Cube::getVolume() {
      return length_ *
             length_ * length_;
  }
...
```

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`

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:

Cube.h

```
…
4  class Cube {
5   public:
6    Cube();
…    /* ... */
```

Cube.cpp

```
…
3  Cube::Cube() {
4
5
6  }
…
```

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

Cube.h

```
…
class Cube {
  public:
   Cube(double length);
   /* ... */
```

Cube.cpp

```
…
Cube::Cube(double length) {


}
…
```

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

puzzle.cpp w/ above custom constructor

```
…
  cs225::Cube c;
  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!

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 power, but extremely dangerous.
- Pointers are **<u>level of indirection</u>** via memory to our data.

As a level of indirection via memory to the data:

1. ______________________________

2. ______________________________

Often, we will have direct access to our object:

```
Cube c1;      // A variable of type Cube
```

Occasionally, we have a reference or pointer to our data:

```
Cube & s1;  // A reference variable of type Cube
Cube * s1;  // A pointer that points to a Cube
```

Reference Variable
A reference variable is an <u>alias</u> 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:

main-ref.cpp

```
int main() {
  int i = 7;
  int & j = i;   // j is an alias of i

  j = 4;                          // j and i are both 4.
  std::cout << i << " " << j << std::endl;

  i = 2;                          // j and i are both 2.
  std::cout << i << " " << j << std::endl;
  return 0;
}
```

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

Three things to note about reference variables:

1. ______________________________

2. ______________________________

3. ______________________________

CS 225 – Things To Be Doing:

1. Sign up for "Exam 0" (exam starts Thursday, Jan. 24th)
2. Attend lab and complete lab_intro; due Sunday, Jan. 20th
3. MP1 released Friday; due Monday, Jan. 28th
4. Visit Piazza and the course website often!