CS 225

Data Structures

January 28 - Lifecycle

Wade Fagen-Ulmschneider, Craig Zilles

Copy Constructor

Copy Constructor

Automatic Copy Constructor

Custom Copy Constructor

Cube.h

```cpp
#pragma once

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


      double getVolume() const;
      double getSurfaceArea() const;

    private:
      double length_;
  };
}
```

Cube.cpp

```cpp
namespace cs225 {
  Cube::Cube() {
    length_ = 1;
    cout << "Default ctor"
         << endl;
  }

  Cube::Cube(double length) {
    length_ = length;
    cout << "1-arg ctor"
         << endl;
  }









  // ...
```

joinCubes-byValue.cpp

```cpp
/*
 * Creates a new Cube that contains the exact volume
 * of the volume of the two input Cubes.
 */
Cube joinCubes(Cube c1, Cube c2) {
  double totalVolume = c1.getVolume() + c2.getVolume();

  double newLength = std::pow( totalVolume, 1.0/3.0 );

  Cube result(newLength);
  return result;
}
```

```cpp
int main() {
  Cube *c1 = new Cube(4);
  Cube *c2 = new Cube(5);

  Cube c3 = joinCubes(*c1, *c2);

  return 0;
}
```

Calls to constructors

	By Value `void foo(Cube a) { … }`	**By Pointer** `void foo(Cube *a) { … }`	**By Reference** `void foo(Cube &a) { … }`
`Cube::Cube()`			
`Cube::Cube(double)`			
`Cube::Cube(const Cube&)`			

joinCubes-byPointer.cpp

```
/*
 * Creates a new Cube that contains the exact volume
 * of the volume of the two input Cubes.
 */
Cube joinCubes(Cube * c1, Cube * c2) {
  double totalVolume = c1->getVolume() + c2->getVolume();

  double newLength = std::pow( totalVolume, 1.0/3.0 );

  Cube result(newLength);
  return result;
}
```

```
int main() {
  Cube *c1 = new Cube(4);
  Cube *c2 = new Cube(5);

  Cube c3 = joinCubes(c1, c2);

  return 0;
}
```

joinCubes-byRef.cpp

```cpp
/*
 * Creates a new Cube that contains the exact volume
 * of the volume of the two input Cubes.
 */
Cube joinCubes(Cube & c1, Cube & c2) {
  double totalVolume = c1.getVolume() + c2.getVolume();

  double newLength = std::pow( totalVolume, 1.0/3.0 );

  Cube result(newLength);
  return result;
}
```

```cpp
int main() {
  Cube *c1 = new Cube(4);
  Cube *c2 = new Cube(5);

  Cube c3 = joinCubes(*c1, *c2);

  return 0;
}
```

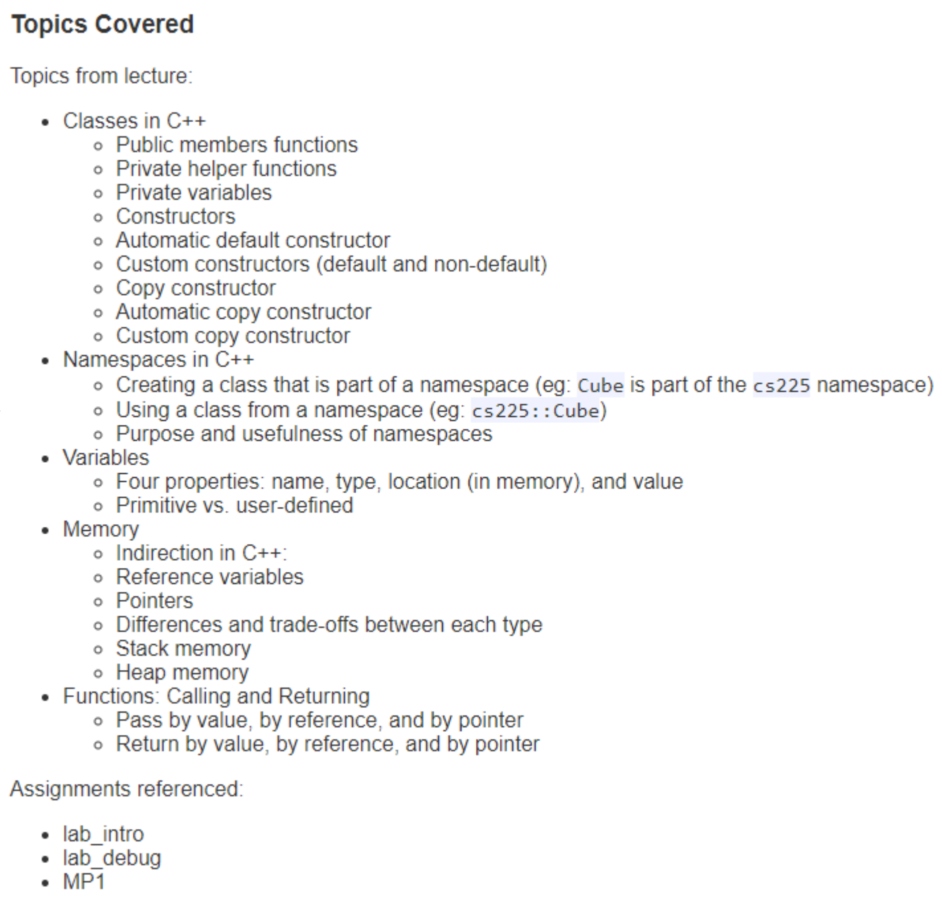
Upcoming: Theory Exam #1

Theory Exam #1

- Starts this Thursday

- 70 points
- 14 MC, 1 code-reading

- **Topic List:** posted to web page soon



Topics Covered

Topics from lecture:

- Classes in C++
  - Public members functions
  - Private helper functions
  - Private variables
  - Constructors
  - Automatic default constructor
  - Custom constructors (default and non-default)
  - Copy constructor
  - Automatic copy constructor
  - Custom copy constructor
- Namespaces in C++
  - Creating a class that is part of a namespace (eg: `Cube` is part of the `cs225` namespace)
  - Using a class from a namespace (eg: `cs225::Cube`)
  - Purpose and usefulness of namespaces
- Variables
  - Four properties: name, type, location (in memory), and value
  - Primitive vs. user-defined
- Memory
  - Indirection in C++:
  - Reference variables
  - Pointers
  - Differences and trade-offs between each type
  - Stack memory
  - Heap memory
- Functions: Calling and Returning
  - Pass by value, by reference, and by pointer
  - Return by value, by reference, and by pointer

Assignments referenced:

- lab_intro
- lab_debug
- MP1

Wade Monday

Honors Section

CS 225 offers a one-credit add on honors section!

What is data science?

Algorithms

Visualizations

Python

Data Structures

JavaScript

pandas

d3.js

Honors Section

Course Starts: Thursday, February 14, 2019

Meets: Thursdays: 5:00 – 5:50pm, 1404 Siebel Center

Taught By: Wade Fagen-Ulmschneider (CS faculty)

Open to EVERYONE – not required to be part of an honors program. Fulfills HCLA, James Scholar, etc.

CS 296, Section 25 (CRN: 31262)

MP1 Deadline

Programming is hard!

MP1 Deadline

Programming is hard!

Every MP in CS 225 will have an automatic 24-hour grace period after the due date.

- **Due:** Monday, 11:59pm
- **Grade Period until:** Tuesday, 11:59pm

MP1 Deadline

Programming is hard!

Every MP in CS 225 will have an automatic 24-hour grace period after the due date.

Due: Monday, 11:59pm

Grade Period until: Tuesday, 11:59pm

Since the MP will past-due, **there are absolutely no office/lab hours on Tuesdays**.

Registration

The last chance to register for CS 225 is today.
We will not being doing any late adds.

If you've registered late, everything so far is due this **Tuesday, January 29th @ 11:59pm**.

- lab_intro
- lab_debug
- mp1

Tower.h

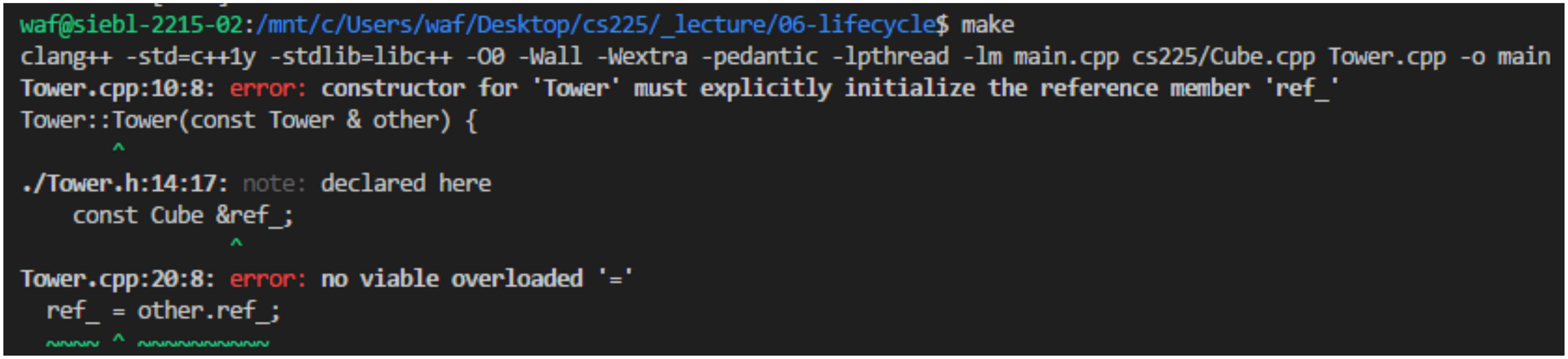
```cpp
#pragma once

#include "cs225/Cube.h"
using cs225::Cube;

class Tower {
  public:
    Tower(Cube c, Cube *ptr, const Cube &ref);
    Tower(const Tower & other);

  private:
    Cube cube_;
    Cube *ptr_;
    const Cube &ref_;
};


```

Tower.cpp

```cpp
Tower::Tower(const Tower & other) {
  cube_ = other.cube_;
  ptr_ = other.ptr_;
  ref_ = other.ref_;
}
```

Tower.cpp

```
10 Tower::Tower(const Tower & other) {
11   cube_ = other.cube_;
12   ptr_ = other.ptr_;
13   ref_ = other.ref_;
14 }
```



```
waf@siebl-2215-02:/mnt/c/Users/waf/Desktop/cs225/_lecture/06-lifecycle$ make
clang++ -std=c++1y -stdlib=libc++ -O0 -Wall -Wextra -pedantic -lpthread -lm main.cpp cs225/Cube.cpp Tower.cpp -o main
Tower.cpp:10:8: error: constructor for 'Tower' must explicitly initialize the reference member 'ref_'
Tower::Tower(const Tower & other) {
       ^
./Tower.h:14:17: note: declared here
    const Cube &ref_;
                ^
Tower.cpp:20:8: error: no viable overloaded '='
  ref_ = other.ref_;
  ~~~~ ^ ~~~~~~~~~~
```

Tower.cpp

```cpp
Tower::Tower(const Tower & other) {
  cube_ = other.cube_;
  ptr_ = other.ptr_;
  ref_ = other.ref_;
}
```

Tower.cpp

```cpp
Tower::Tower(const Tower & other) : cube_(other.cube_),
   ptr_(other.ptr_), ref_(other.ref_) { }
```

Constructor Initializer List

Tower.cpp

```
Tower::Tower(const Tower & other) {
  // Deep copy cube_:






  // Deep copy ptr_:






  // Deep copy ref_:





}
```

Destructor

[Purpose]:

Destructor

[Purpose]: **Free any resources maintained by the class.**

Automatic Destructor:

1. Exists only when no custom destructor is defined.

2. [Functionality]:

[Invoked]:

Cube.h

```cpp
#pragma once

namespace cs225 {
  class Cube {
    public:
      Cube();
      Cube(double length);
      Cube(const Cube & other);
      ~Cube();

      double getVolume() const;
      double getSurfaceArea() const;

    private:
      double length_;
  };
}
```

Cube.cpp

```cpp
namespace cs225 {
  Cube::Cube() {
    length_ = 1;
    cout << "Default ctor"
         << endl;
  }

  Cube::Cube(double length) {
    length_ = length;
    cout << "1-arg ctor"
         << endl;
  }










  // ...
```