CS 225 Spring 2019 :: TA Lecture Notes
1/16 Classes

By Wenjie

Thanks to all of you who pointed out typos in the notes, and gave formatting suggestions:)
But the google doc seems auto-reject comments from time to time - please lmk if you ran into this!

- **Public vs Private**
    - Public: similar to java, members in public classes can be accessed from outside of the class (i.e from the main function)
    - Private: members in private classes can only be used inside of its own class and not allowed to be even viewed from outside

- **Namespace**
    - Libraries in C++ are organized into namespaces (like packages in Java). We cannot have two classes with the same name in the same namespace.
    - Useful shortcut:
        - cs225::Cube
        - std :: cout
    - Discouraged: import everything from a namespace
    - Examples:
        - std - the standard namespace, including cout, vector, queue, etc.
        - CS225 - Cube, PNG, HSLAPixel etc...

cube.h

```
#pragma once
namespace cs225 {
  class Cube {
    public:
      double getVolume();
      double getSurfaceArea();
      void setLength(double length);
    private:
      double length_;
  };
}
```

cube.cpp

```
#include "Cube.h"
namespace cs225 {
  double Cube::getVolume() {
    return length_*length_*length_;
  }
  double Cube::getSurfaceArea() {
    return 6 * length_ * length_;
  }
 void Cube::setLength(double length){
    length_ = length;
  }
}
```

CS 225 Spring 2019 :: TA Lecture Notes

1/16 Classes

By Wenjie

main.cpp

```
1  #include "Cube.h"
2  #include <iostream>
3
4  int main {
5      cs225::Cube c;
6      std::cout << "Volume: " <<
7  c.getVolume() << std::endl;
8      return 0;
9  }
```

```
line 5:  Declaring an object of type Cube. Our cube is inside of the namespace cs225, so we have a scope resolution operator indicating that the class belongs to the namespace cs225.

line 6:  cout is a print statement in C++. It resides in the standard library which we included in line 2. The double less sign is called alligator brackets. Finally, endl is adding a new line (\n) to the end of the output.

line 8:  if the main function returns 0 then it is saying that the execution completed fine.
```

- **Constructor**
  - Default Constructor:
    - Automatic Default Constructor: provided automatically if no constructor is defined
    - No parameter
    - Initialize the class value to default values
  - Customized constructors
    - In order to be able to choose values for our member variables we need to define **custom constructors**. As soon as we define one custom constructor, automatic default constructor is gone. In other words, if we don't use one of the defined constructor, the program won't compile.

    - We can define multiple constructors and usually one of them is default. In the default constructor we assign values to variables, while the others we allow the user to choose values.

Cube.h

```
/* … */
class Cube{
    Public:
// default constructor
       Cube ();
       Cube (double s);

/* … */
```

Cube.cpp

```
/* … */
//define default constructor
Cube::Cube() {
     radius_ = 1;
}
Cube::Cube(double r) {
     radius_ = r;
}
```

  - In this case, if we want to define a cube object in main class without specify any int value for it:
    - 1 - use default automatic constructor
      - cs225::cube c;
    - 2 - use self defined default constructor
    - 3 - void setLength();

- **Reference Variable**
  - An alias to an existing variable.
  - Must be initialized upon creation and its reference cannot be changed
  - Does not create its own memory

main.cpp

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

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

     i = 2;
```

```
      std::cout << i << “ ” << j << std::endl;
// j and i are both 2
      return 0;
}
```

- Since j is alias of i and their values are bound together. Once the value of j changes, value of i will also be modified, and vice versa.