CS 225

Data Structures

January 21 – C++ Review

G Carl Evans

Encapsulation

Cube.h

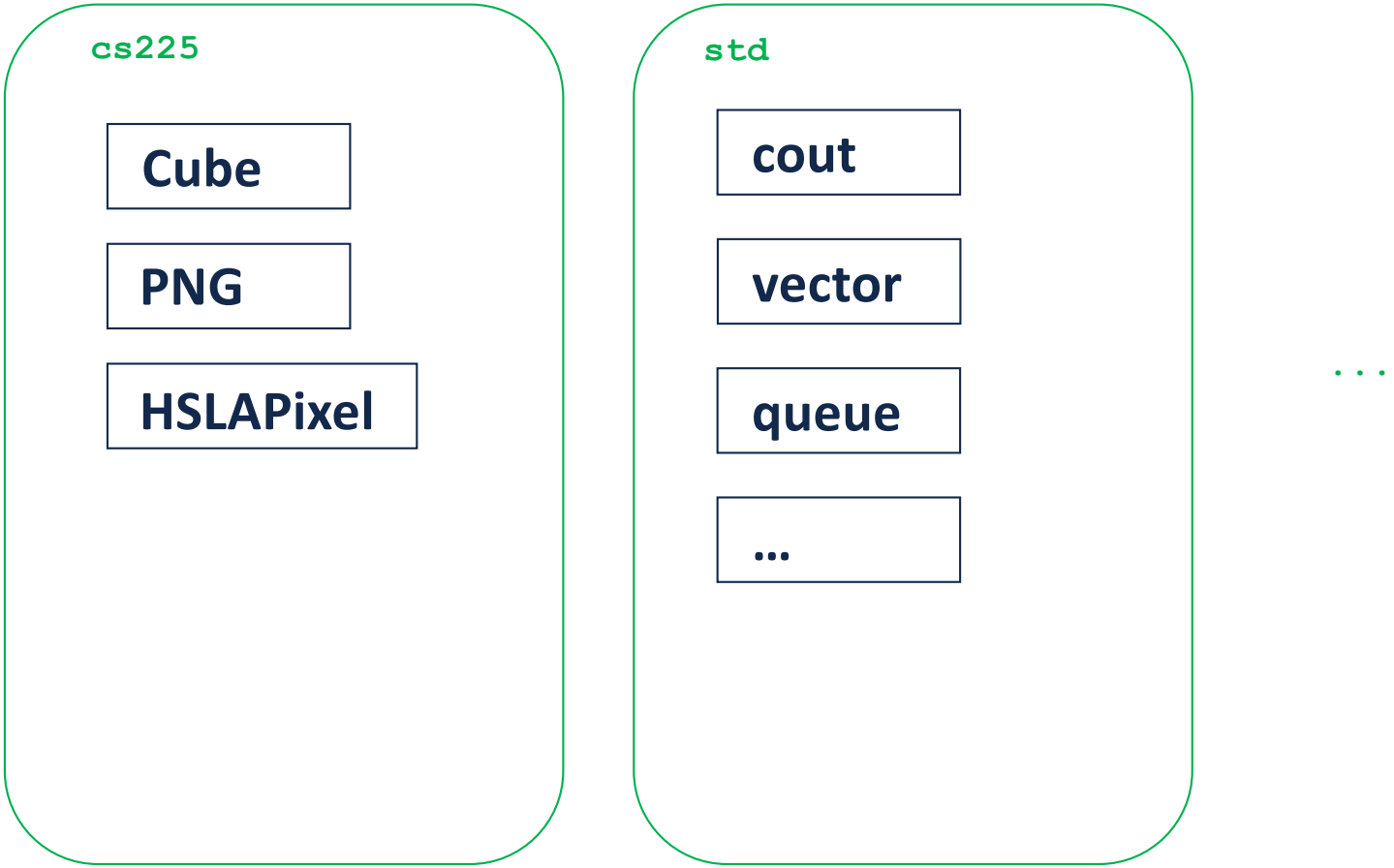
```
 1 #pragma once
 2
 3 class Cube {
 4   public:
 5     double getVolume();
 6
 7
 8
 9
10
11   private:
12
13
14 };
15
16
17
18
19
20
```

Cube.cpp

```
 1 #include "Cube.h"
 2
 3 double Cube::getVolume() {
 4
 5
 6 }
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
```

Namespaces

Namespaces



Cube.h

```
 1 #pragma once
 2
 3 namespace cs225 {
 4   class Cube {
 5     public:
 6       double getVolume();
 7       double getSurfaceArea();
 8
 9
10
11
12     private:
13       double length_;
14
15   };
16 }
17
18
19
20
```

Cube.cpp

```
 1 #include "Cube.h"
 2
 3 namespace cs225 {
 4   double Cube::getVolume() {
 5     return length_ * length_ *
           length_;
 6   }
 7
 8   double
     Cube::getSurfaceArea() {
 9     return 6 * length_ *
           length_;
10   }
11 }
12
13
14
15
16
17
```

Cube.h

```
1  #pragma once
2
3  namespace cs225 {
4    class Cube {
5      public:
6        double getVolume();
7        double getSurfaceArea();
8
9
10
11
12     private:
13
14
15
16 }
17
18
19
20
```

Cube.cpp

```
1  #include "Cube.h"
2
3  namespace cs225 {
4    double Cube::getVolume() {
5      return length_ * length_ *
               length_;
6    }
7
8    double
     Cube::getSurfaceArea() {
9      return 6 * length_ *
```

```
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  }
```

Constructor

Cube.h

```
 1 #pragma once
 2
 3 namespace cs225 {
 4   class Cube {
 5     public:
 6       Cube();
 7       double getVolume();
 8       double getSurfaceArea();
 9
10
11
12
13     private:
14       double length_;
15
16   };
17 }
18
19
20
```

Cube.cpp

```
 1 #include "Cube.h"
 2 namespace cs225 {
 3   Cube::Cube() {
 4
 5
 6   }
 7
 8   double Cube::getVolume() {
 9     return length_ * length_ *
10            length_;
11   }
12
13   double
14   Cube::getSurfaceArea() {
15     return 6 * length_ *
16            length_;
17   }
18 }
19
20
```

Cube.h

```
 1  #pragma once
 2
 3  namespace cs225 {
 4    class Cube {
 5      public:
 6        Cube(double length);
 7        double getVolume();
 8        double getSurfaceArea();
 9
10
11
12
13      private:
14        double length_;
15
16    };
17  }
18
19
20
```

Cube.cpp

```
 1  #include "Cube.h"
 2  namespace cs225 {
 3    Cube::Cube(double length) {
 4
 5
 6    }
 7
 8    double Cube::getVolume() {
 9      return length_ * length_ *
10             length_;
11    }
12
13    double
14    Cube::getSurfaceArea() {
15      return 6 * length_ *
16             length_;
17    }
18  }
19
20
```

Cube.h

```
1  #pragma once
2
3  namespace cs225 {
4    class Cube {
5      public:
6        Cube(double length);
7        double getVolume();
8        double getSurfaceArea();
9
10
11
12
13
14
15
16
17 }
18
19
20
```

Cube.cpp

```
1  #include "Cube.h"
2  namespace cs225 {
3    Cube::Cube(double length) {
4      length_ = length;
5
6    }
7
8    double Cube::getVolume() {
```

puzzle.cpp

```
1  #include "Cube.h"
2  using cs225::Cube;
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int main() {
8    Cube c;
9    cout << "Volume: " << c.getVolume() << endl;
10   return 0;
11 }
```

Hate Typing cout:: and cs225::?

Useful Shortcut:

```
using std::cout;   // Imports cout into global scope
using std::endl;   // Imports endl into global scope
using cs225::Cube; // Imports Cube into global scope
```

Strongly Discouraged Shortcut

```
using namespace std;  // Imports all of std:: into
                      // global scope!
                      // ...THOUSANDS of things!
```

Cube.h

```
1  #pragma once
2
3  namespace cs225 {
4    class Cube {
5      public:
6
7        Cube(double length);
8        double getVolume();
9        double getSurfaceArea();
10
11
12
13
14
15
16
17
18 }
19
20
```

Cube.cpp

```
1  #include "Cube.h"
2  namespace cs225 {
3
4
5
6
7    Cube::Cube(double length) {
8      length_ = length;
9    }
10
11   double Cube::getVolume() {
```

puzzle.cpp

```
7  int main() {
8    Cube c;
9    cout << "Volume: " << c.getVolume() << endl;
10   return 0;
11 }
```

Cube.h

```
1  #pragma once
2
3  namespace cs225 {
4    class Cube {
5      public:
6
7        Cube(double length);
8        double getVolume();
9        double getSurfaceArea();
10
11
12
13
14
15
16
17
18 }
19
20
```

Cube.cpp

```
1  #include "Cube.h"
2  namespace cs225 {
3
4
5
6
7    Cube::Cube(double length) {
8      length_ = length;
9    }
10
11   double Cube::getVolume() {
```

puzzle.cpp

```
7  int main() {
8    Cube c;
9    cout << "Volume: " << c.getVolume() << endl;
10   return 0;
11 }
```

Pointers and References

A variable containing an instance of an object:

```
1 Cube s1;
```

A reference variable of a Cube object:

```
1 Cube & r1 = s1;
```

A variable containing a pointer to a Cube object:

```
1 Cube * p1;
```

Pointers

Three key ideas:

1.

2.

3.

Indirection Operators

Given any variable v:

&v

***v**

v->