CS 225

Data Structures

February 5 – Overloading

G Carl Evans

Destructor

[Purpose]:

Destructor

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

Automatic Destructor:

1. Exists only when no custom destructor is defined.

2. [Invoked]:

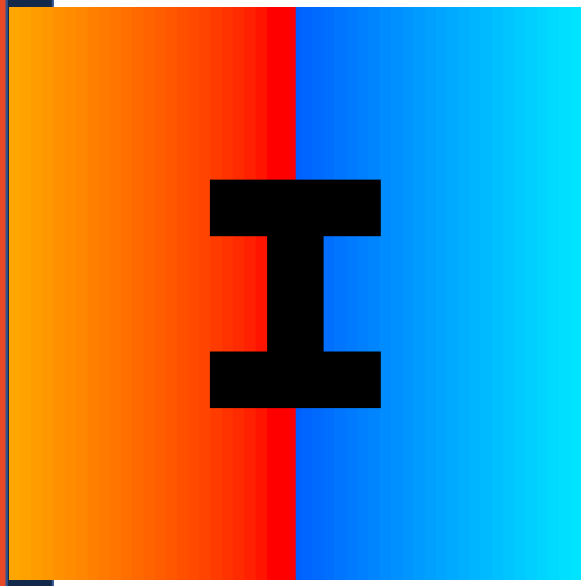
3. [Functionality]:

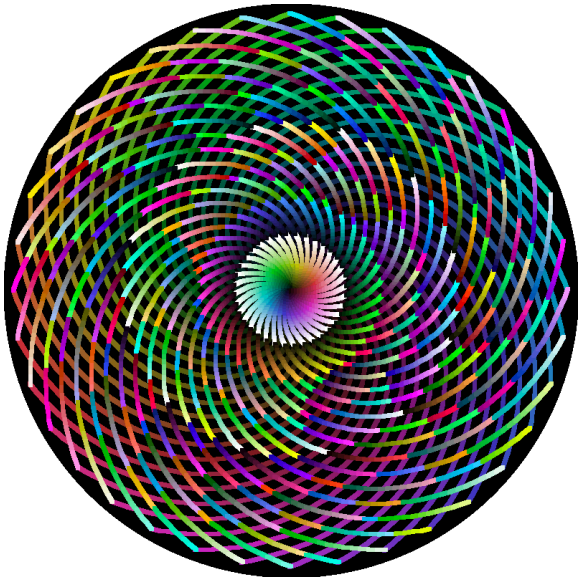
cs225/Cube.h

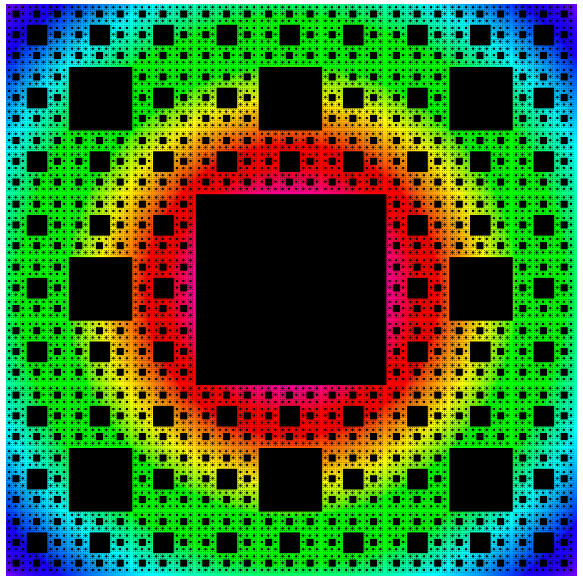
```
 1  #pragma once
 2
 3  namespace cs225 {
 4    class Cube {
 5      public:
 6        Cube();
 7        Cube(double length);
 8        Cube(const Cube & other);
 9        ~Cube();
10
11        double getVolume() const;
12        double getSurfaceArea() const;
13
14      private:
15        double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

cs225/Cube.cpp

```
 7  namespace cs225 {
 8    Cube::Cube() {
 9      length_ = 1;
10      cout << "Default ctor"
             << endl;
11    }
12
13    Cube::Cube(double length) {
14      length_ = length;
15      cout << "1-arg ctor"
             << endl;
16    }
17
18
19
20
21
22
23
24
25
…     // ...
```







MP 1 Art

Operators that can be overloaded in C++	
Arithmetic	+ - * / % ++ --
Bitwise	& | ^ ~ << >>
Assignment	=
Comparison	== != > < >= <=
Logical	! && ||
Other	[] () ->

cs225/Cube.h

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

cs225/Cube.cpp

```
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
```

One Very Special Operator

Definition Syntax (.h):

```
Cube & operator=(const Cube& s)
```

Implementation Syntax (.cpp):

```
Cube & Cube::operator=(const Cube& s)
```

Assignment Operator

Similar to Copy Constructor:

Different from Copy Constructor:

Assignment Operator

	Copies an object	Destroys an object
Copy constructor		
Copy Assignment operator		
Destructor		

MP: Extra Credit

The most successful MP is an MP done early!

Unless otherwise specified in the MP, we will award +1 extra credit point per day **for completing Part 1** before the due date *(up to +7 points)*:

Example for MP2:

+7 points: Complete by **Monday**, Sept. 16 (11:59pm)
+6 points: Complete by **Tuesday**, Sept. 17 (11:59pm)
+5 points: Complete by **Wednesday**, Sept. 18 (11:59pm)
+4 points: Complete by **Thursday**, Sept. 19 (11:59pm)
+3 points: Complete by **Friday**, Sept. 20 (11:59pm)
+2 points: Complete by **Saturday**, Sept. 21 (11:59pm)
+1 points: Complete by **Sunday**, Sept. 22 (11:59pm)
MP2 Due Date: Monday, Sept. 23

MP: Extra Credit

We will give **partial credit** and **maximize the value** of your extra credit:

You made a submission and missed a few edge cases in Part 1:

Monday: +7 * 80% = **+5.6** earned

MP: Extra Credit

We will give **partial credit** and **maximize the value** of your extra credit:

You made a submission and missed a few edge cases in Part 1:

Monday: +7 * 80% = **+5.6** earned

You fixed your code and got a perfect score on Part 1:

Tuesday: +6 * 100% = **+6** earned *(maximum benefit)*

MP: Extra Credit

We will give **partial credit** and **maximize the value** of your extra credit:

You made a submission and missed a few edge cases in Part 1:

Monday: +7 * 80% = **+5.6** earned

You fixed your code and got a perfect score on Part 1:

Tuesday: +6 * 100% = **+6** earned *(maximum benefit)*

You began working on Part 2, but added a compile error:

Wednesday: +5 * 0% = +0 earned *(okay to score lower later)*

…

The “Rule of Three”

If it is necessary to define any one of these three functions in a class, it will be necessary to define all three of these functions:

1.

2.

3.

Rvalue Reference or Move Semantics

- **Rvalue**

- **Move**

```
Cube(const Cube&& s)noexcept
```

- **Move Assignment**

```
Cube & operator=(const Cube&& s)noexcept
```

The “Rule of Five”

If it is necessary to define any one of these five functions in a class, it will be necessary to define all five of these functions:

1.

2.

3.

4.

5.

The “Rule of Zero”

Corollary to Rule of Five

Classes that **declare** custom destructors, copy/move constructors or copy/move assignment operators should deal exclusively with ownership. Other classes should not **declare** custom destructors, copy/move constructors or copy/move assignment operators

–Scott Meyers

In CS 225