CS 225

#7: Overloading

February 5, 2020 · *G Carl Evans*

Destructor

The last and final member function called in the lifecycle of a class is the destructor.

Purpose of a **destructor**:

The **automatic destructor**:

1. Like a constructor and copy constructor, an automatic destructor exists only when no custom destructor is defined.
2. [Invoked]:
3. [Functionality]:

Custom Destructor:

Cube.h

```
5  class Cube {
6    public:
7      Cube();                  // default ctor
8      Cube(double length);     // 1-param ctor
9      Cube(const Cube & other);   // custom copy ctor
10     ~Cube();                 // destructor, or dtor
11     ...
```

...necessary if you need to delete any heap memory!

Overloading Operators

C++ allows custom behaviors to be defined on over 20 operators:

Arithmetic	`+ - * / % ++ --`
Bitwise	`& | ^ ~ << >>`
Assignment	`=`
Comparison	`== != > < >= <=`
Logical	`! && ||`
Other	`[] () ->`

General Syntax:

Adding overloaded operators to Cube:

Cube.h

```
1  #pragma once
2
3  class Cube {
4    public:
…      // ...
10
11
12
13
14
…      // ...
```

Cube.cpp

```
…  /* ... */
40
41
42
43
44
45
46
47
48
…  /* ... */
```

One Very Powerful Operator: Assignment Operator

Cube.h

```
Cube & operator=(const Cube & other);
```

Cube.cpp

```
Cube & Cube::operator=(const Cube & other) { ... }
```

Functionality Table:

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

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 and Move Semantics

Cube.h	
	`Cube & operator=(const Cube && other) noexcept;` `Cube(Cube && other) noexcept;`
Cube.cpp	
	`Cube & Cube::operator=(const Cube && other) noexcept{ ... }` `Cube(Cube && other) noexcept { ... }`

Lvalue

Rvalue

Why Move?

1.

2.

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

CS 225 and Rule Three/Five/Zero

In CS 225 We will:

Inheritance

In nearly all object-oriented languages (including C++), classes can be extended to build other classes. We call the class being extended the **base class** and the class inheriting the functionality the **derived class**.

CS 225 – Things To Be Doing:
1. Quiz 1 starts Friday! **2.** lab_memory this week in labs *(due Sunday)* **3.** *MP2 released (EC due Monday)* **4.** Daily POTDs every M-F for daily extra credit!