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:

```cpp
class Cube {
public:
    Cube();        // default ctor
    Cube(double length); // 1-param ctor
    Cube(const Cube & other); // custom copy ctor
    ~Cube();        // destructor, or dtor
    ... 

...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      | [] () -> |

Adding overloaded operators to Cube:

```cpp
#pragma once
class Cube {
public:
    Cube();       // default ctor
    Cube(double length); // 1-param ctor
    Cube(const Cube & other); // custom copy ctor
    ~Cube();       // destructor, or dtor
    ... 

    Cube & operator=(const Cube & other);

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

One Very Powerful Operator: Assignment Operator

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

Functionality Table:

<table>
<thead>
<tr>
<th></th>
<th>Copies an object</th>
<th>Destroys an object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy constructor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy Assignment operator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destructor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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. mp_intro due today
2. Daily POTDs every M-F for daily extra credit!