CS 225

#6: The Rule of Three

September 11, 2017

Three ways to return a variable:

joinSpheres-returnByValue.cpp

```
15 Sphere joinSpheres(const Sphere &s1, const Sphere &s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   return result(newRadius);
24 }
```

joinSpheres-returnByPointer.cpp

```
15 Sphere *joinSpheres(const Sphere &s1, const Sphere &s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   return new Sphere(newRadius);
24 }
```

joinSpheres-returnByReference.cpp

```
15 Sphere & joinSpheres(const Sphere &s1, const Sphere &s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22   //
23   // BAD PRATICE: _____________________________________
24   //
25   Sphere *result = new Sphere(newRadius);
26   return *result; // Memory must be delete'd later
27 }
```

```
22   //
23   // ERROR: ___________________________________________
24   //
25   Sphere result(newRadius);
26   return result;
27 }
```

Return by reference takeaway:

Adding our Sphere class:

What should happen when we add two `Sphere` classes?

addSpheres.cpp

```
1 #include "sphere.h"
2
3 int main() {
4   cs225::Sphere s1(3), s2(4);
5   cs225::Sphere s3 = s1 + s2;
6   return 0;
7 }
```

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 Sphere:

sphere.h

```
1  #ifndef SPHERE_H
2  #define SPHERE_H
3
4  class Sphere {
5    public:
…      // ...
17
18
19
20
21
…      // ...
30   private:
31     double r_;
32 };
33
34 #endif
```

sphere.cpp

```
…  /* ... */
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
…  /* ... */
```

One Very Special Operator: _____

1. Similar to the Copy Constructor:

2. Different than the Copy Constructor:

sphere.h

```
1  #ifndef SPHERE_H
2  #define SPHERE_H
3
4  class Sphere {
5    public:
…      // ...
17
18
19
20
21
…      // ...
30   private:
31     double r_;
32 };
33
34 #endif
```

sphere.cpp

```
…  /* ... */
10
11
12
13
14
15
16
17
18
…  /* ... */
```

Big Idea: Destructor

sphere.h

```
1  #ifndef SPHERE_H
2  #define SPHERE_H
3
4  class Sphere {
5    public:
…      // ...
17
18
…      // ...
```

sphere.cpp

```
…  /* ... */
10
11
12
13
14
15
16
…  /* ... */
```

The "Rule of Three":

1.

2.

3.

Why are the big three so important?

Example:

sphere.h

```
1  #ifndef SPHERE_H
2  #define SPHERE_H
3
4  class Sphere {
5    public:
…      // ...
17
18
…      // ...
```

sphere.cpp

```
…  /* ... */
10
11
12
13
14
15
16
…  /* ... */
```

CS 225 – Things To Be Doing:

1. Exam #1 (CBTF) is ongoing; Register for Exam #2
2. MP1 due tonight; grace period until Tuesday @ 11:59pm
3. MP2 released on Tuesday *(start early for +EC!)*
4. Lab Extra Credit → Attendance in your registered lab section!
5. POTDs