CS 225

Data Structures

Wade Fagen-Ulmschneider

joinSpheres-returnByValue.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

joinSpheres-returnByReference2.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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
24
25   Sphere result(newRadius);
26   return result;
27 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

joinSpheres-returnByReference.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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
24
25   Sphere *result =
        new Sphere(newRadius);
26   return *result;
27 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

joinSpheres-returnByPointer.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere *s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

Upcoming: Theory Exam #1

Theory Exam #1

- Starts tomorrow

- **Topic List:**

  https://courses.engr.illinois.edu/cs225/sp2018/exams/exam-theory1/

- **Review Session:**

  Monday, 7:00pm, 1404 Siebel Center

Topics Covered

Topics from lecture:

- Classes in C++
  - Public members functions
  - Private helper functions
  - Private variables
  - Constructors
  - Automatic default constructor
- Namespaces in C++
  - Creating a class that is part of a namespace (eg: Sphere is part of the cs225 namespace)
  - Using a class from a namespace (eg: cs225::Sphere)
  - Purpose and usefulness of namespaces
- Variables
  - Four properties: name, type, location (in memory), and value
  - Primitive vs. user-defined
- Memory
  - Indirection in C++:
  - Reference variables
  - Pointers
  - Differences and trade-offs between each type
  - Stack memory
  - Heap memory
- Functions: Calling and Returning
  - Pass by value, by reference, and by pointer
  - Return by value, by reference, and by pointer

Assignments referenced:

- lab_intro
- lab_debug
- MP1

MP1 Deadline

Programming is hard!

MP1 Deadline

Programming is hard!

Every MP in CS 225 will have an automatic 24-hour grace period after the due date.

Due: Monday, 11:59pm

Grade Period until: Tuesday, 11:59pm

MP1 Deadline

Programming is hard!

Every MP in CS 225 will have an automatic 24-hour grace period after the due date.

Due: Monday, 11:59pm

Grade Period until: Tuesday, 11:59pm

Since the MP will past-due, **there are absolutely no office/lab hours on Tuesdays**.

Registration

The last chance to register for CS 225 is today.
We will not being doing any late adds.

If you've registered late, everything so far is due this **Tuesday, Jan. 30 @ 11:59pm**.

- lab_intro
- lab_debug
- mp1

Copy Constructor

Copy Constructor

Automatic Copy Constructor

Custom Copy Constructor

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6      public:
 7        Sphere();
 8        Sphere(double r);
 9
10
11        double getRadius() const;
12        double getVolume() const;
13
14        void setRadius(double r);
15
16      private:
17        double r_;
18
19    };
20  }
21
22  #endif
```

sphere.cpp

```
 1  #include "sphere.h"
 2  #include <iostream>
 3
 4  using namespace std;
 5
 6  namespace cs225 {
 7    Sphere::Sphere() : Sphere(1) {
 8       cout << "Default ctor" << endl;
 9    }
10
11    Sphere::Sphere(double r) {
12      cout << "1-param ctor" << endl;
13      r_ = r;
14    }
15
16
17
18
19
20
21
…     // ...
```

Calls to constructors

	By Value `void foo(Sphere a) { … }`	By Pointer `void foo(Sphere *a) { … }`	By Reference `void foo(Sphere &a) { … }`
`Sphere::Sphere()`			
`Sphere::Sphere(double)`			
`Sphere::Sphere(const Sphere&)`			

joinSpheres-returnByValue.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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   Sphere result(newRadius);
24
25   reutrn result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

joinSpheres-returnByPointer.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere *s3 = joinSpheres(s1, s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

joinSpheres-returnByReference.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
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   // !: Not good practice,
24   //    memory not free'd!
25   Sphere *result =
        new Sphere(newRadius);
26   return *result;
27 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

Universe.h

```
 1 #ifndef UNIVERSE_H_
 2 #deifne UNIVERSE_H_
 3
 4 #include "Sphere.h"
 5 using namespace cs225;
 6
 7 class Universe {
 8   public:
 9     Universe();        // default ctor
10     Universe(Sphere s, Sphere *q, Sphere &r);  // 3-param
11     Universe(const Universe & other);
12     // …
13   private:
14     Sphere p_, *q_, &r;
15 };
16
17 #endif
```

Universe.cpp

```
10 Universe::Universe(const Universe & other) {
11   p_ = other.p_;
12   q_ = other.q_;
13   r_ = other.r_;
14 }
```

Universe.cpp

```
10 Universe::Universe(const Universe & other) {
11   p_ = other.p_;
12   q_ = other.q_;
13   r_ = other.r_;
14 }
```

Universe.cpp:7:2: error: constructor for 'Universe' must explicitly initialize the reference member 'r_'

Universe.cpp

```
10 Universe::Universe(const Universe & other) {
11   p_ = other.p_;
12   q_ = other.q_;
13   r_ = other.r_;
14 }
```

Universe.cpp:7:2: error: constructor for 'Universe' must explicitly initialize the reference member 'r_'

Universe.cpp

```
10 Universe::Universe(const Universe & other) :
11   p_(other.p_), q_(other.q_), r_(other.r_) {
12
13
14 }
```

Constructor Initializer List

Universe.cpp

```
16 Universe::Universe(const Universe & other) {
17   // Deep copy p_:
18
19
20
21   // Deep copy q_:
22
23
24
25   // Deep copy r_:
26
27
28
29 }
```

Destructor

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6      public:
 7        Sphere();
 8        Sphere(double r);
 9        Sphere(const Sphere &s);
10        ~Sphere();
11
12
13
14
 …        // ...
26      private:
27        double r_;
28
29    };
30  }
31
32  #endif
```

sphere.cpp

```
 1  #include "sphere.h"
 2
 3  namespace cs225 {
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
 …      // ...
    }
```

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

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6      public:
 7        Sphere();
 8        Sphere(double r);
 9        Sphere(const Sphere &s);
10        ~Sphere();
11
12
13
14
 …        // ...
26      private:
27        double r_;
28
29    };
30  }
31
32  #endif
```

sphere.cpp

```
 1  #include "sphere.h"
 2
 3  namespace cs225 {
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
 …      // ...
    }
```

CS 225 – Things To Be Doing

Theory Exam 1 starts tomorrow

More Info: https://courses.engr.illinois.edu/cs225/sp2018/exams/

MP1 – Due Tonight (11:59pm)

MP2 Release: Tuesday, Sept 12th – Up to +7 Extra Credit for Early Submission

POTD

Every Monday-Friday – *Worth +1 Extra Credit /problem (up to +40 total)*