CS 225
Data Structures
/ * Creates a new sphere that contains the exact volume
 * of the volume of the two input spheres. */

Sphere joinSpheres(const Sphere &s1, const Sphere &s2) {
    double totalVolume = s1.getVolume() + s2.getVolume();

    double newRadius = std::pow((3.0 * totalVolume) / (4.0 * 3.141592654), 1.0/3.0);

    return result(newRadius);
}

int main() {
    Sphere *s1 = new Sphere(4);
    Sphere *s2 = new Sphere(5);

    Sphere s3 = joinSpheres(*s1, *s2);

    delete s1; s1 = NULL;
    delete s2; s2 = NULL;

    return 0;
}
*/
* Creates a new sphere that contains the exact volume
* of the volume of the two input spheres.
* /

Sphere *joinSpheres(const Sphere &s1, const Sphere &s2) {
    double totalVolume = s1.getVolume() + s2.getVolume();
    double newRadius = std::pow((3.0 * totalVolume) / (4.0 * 3.141592654), 1.0/3.0);
    return new Sphere(newRadius);
}

int main() {
    Sphere *s1 = new Sphere(4);
    Sphere *s2 = new Sphere(5);
    Sphere *s3 = joinSpheres(*s1, *s2);
    delete s1; s1 = NULL;
    delete s2; s2 = NULL;
    return 0;
}
/ * Creates a new sphere that contains the exact volume
  * of the volume of the two input spheres.
*/
Sphere & joinSpheres(const Sphere &s1, const Sphere &s2) {
  double totalVolume = s1.getVolume() + s2.getVolume();
  double newRadius = std::pow((3.0 * totalVolume) / (4.0 * 3.141592654), 1.0/3.0);
  Sphere *result = new Sphere(newRadius);
  return *result;
}

int main() {
  Sphere *s1 = new Sphere(4);
  Sphere *s2 = new Sphere(5);
  Sphere s3 = joinSpheres(*s1, *s2);
  delete s1; s1 = NULL;
  delete s2; s2 = NULL;
  return 0;
}
/*
 * Creates a new sphere that contains the exact volume
 * of the volume of the two input spheres.
 */

Sphere & joinSpheres(const Sphere &s1, const Sphere &s2) {
    double totalVolume = s1.getVolume() + s2.getVolume();
    double newRadius = std::pow((3.0 * totalVolume) / (4.0 * 3.141592654), 1.0/3.0);
    Sphere result(newRadius);
    return result;
}

int main() {
    Sphere *s1 = new Sphere(4);
    Sphere *s2 = new Sphere(5);
    Sphere s3 = joinSpheres(*s1, *s2);
    delete s1; s1 = NULL;
    delete s2; s2 = NULL;
    return 0;
}
Exam #2

• Will be a coding exam!
  • Similar format as the POTDs
  • You will compile locally, then get autograder feedback.
  • One or two problems.
    • MP 1
    • Labs
    • In-class code
Honors Section

• Starts Friday, September 22

• Trying to get it to be 5pm; the time in Banner needs to be changed

• Topics:
  • Functional programming
  • Data structures that are *immutable*
  • Programming “in the large”
  • Clojure
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
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 be doing any late adds.

If you’ve registered late, everything so far is due this **Tuesday, Sept. 12 @ 11:59pm**.

- lab_intro
- lab_debug
- mp1
#include "sphere.h"

int main() {
    cs255::Sphere s1(3), s2(4);
    cs255::Sphere s3 = s1 + s2;
    return 0;
}
### Operators that can be overloaded in C++

<table>
<thead>
<tr>
<th>Category</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>+  -  *  /  %  ++  --</td>
</tr>
<tr>
<td>Bitwise</td>
<td>&amp;</td>
</tr>
<tr>
<td>Assignment</td>
<td>=</td>
</tr>
<tr>
<td>Comparison</td>
<td>==  !=  &gt;  &lt;  &gt;=  &lt;=</td>
</tr>
<tr>
<td>Logical</td>
<td>!  &amp;&amp;</td>
</tr>
<tr>
<td>Other</td>
<td>[ ]  ( )  -&gt;</td>
</tr>
</tbody>
</table>
```cpp
#ifndef SPHERE_H
#define SPHERE_H

namespace cs225 {

class Sphere {
   public:
      Sphere();
      Sphere(double r);
      Sphere(const Sphere &s);

   private:
      double r_; 

};

} /* namespace cs225 */
#endif /* SPHERE_H */
```

```cpp
#include "sphere.h"

namespace cs225 {

// ...

} /* namespace cs225 */
```

One Very Special Operator

Definition Syntax (.h):

```cpp
Sphere& operator=(const Sphere& s)
```

Implementation Syntax (.cpp):

```cpp
Sphere& Sphere::operator=(const Sphere& s)
```
Assignment Operator

Similar to Copy Constructor:

Different from Copy Constructor:
What constructors and operators are called?

```
1. Sphere s1, s2;
2.
3. s2 = s1;
4.
5. Sphere s3 = s1;
6.
7. Sphere *s4 = &s3;
8.
9. Sphere &s5 = s2;
```
Destructor
```cpp
#ifndef SPHERE_H
#define SPHERE_H

namespace cs225 {

class Sphere {
public:
    Sphere();
    Sphere(double r);
    Sphere(const Sphere &s);

    // ...

private:
    double r_;  
};
}

#endif

#include "sphere.h"

namespace cs225 {

// ...

... // ...
```
The “Rule of Three”
Towards a more advanced Sphere...
```cpp
#ifndef SPHERE_H
#define SPHERE_H

namespace cs225 {

class Sphere {

public:
    Sphere();
    Sphere(double r);
    Sphere(const Sphere &s);

    // ...

private:
    double r_; 

};
}

#endif
```

```cpp
#include "sphere.h"

namespace cs225 {

// ...
}
```
Exam 1 is happening now
Exam 2 registration is available (programming exam)
More Info: https://courses.engr.illinois.edu/cs225/fa2017/exams/

Finish MP1 – Due Tonight (11:59pm)
*MP1 Grace period until Tuesday @ 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)*