CS 225

Data Structures

September 7 - Parameters

Wade Fagen-Ulmschneider

heap-puzzle3.cpp

```
 5 int *x;
 6 int size = 3;
 7
 8 x = new int[size];
 9
10 for (int i = 0; i < size; i++) {
11   x[i] = i + 3;
12 }
13
14 delete[] x;
```

joinCubes-byValue.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(Cube c1, Cube c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

joinCubes-byPointer.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(Cube * c1, Cube * c2) {
16   double totalVolume = c1->getVolume() + c2->getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(c1, c2);
33
34   return 0;
35 }
```

joinCubes-byRef.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(Cube & c1, Cube & c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

Parameter Passing Properties

	By Value `void foo(Cube a) { … }`	**By Pointer** `void foo(Cube *a) { … }`	**By Reference** `void foo(Cube &a) { … }`
Exactly what is copied when the function is invoked?			
Does modification of the passed in object modify the caller's object?			
Is there always a valid object passed in to the function?			
Speed			
Programming Safety			

MP1

Due: Monday, Sept. 10th (11:59pm)

Share your art work:

- On our piazza, in the "MP1 - Artwork Sharing" thread
- On social media:
  - If your post is **public** and contains **#cs225**, I'll throw it a like/heart and so will some of your peers! ☺

My promise: I will look at <u>all</u> the artwork after the submission deadline. Course staff and I will give **+1** to all that stand out!

Using `const` in function parameters

joinCubes-byValue-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube c1, const Cube c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

joinCubes-byPointer-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube * c1, const Cube * c2) {
16   double totalVolume = c1->getVolume() + c2->getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(c1, c2);
33
34   return 0;
35 }
```

joinCubes-byRef-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube & c1, const Cube & c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

TERMINAL ••• 1: wsl

```
waf@siebl-2215-02:/mnt/c/Users/waf/Desktop/cs225/_lecture/05-parameters$
make
clang++ -std=c++1y -stdlib=libc++ -O0 -Wall -Wextra -pedantic -lpthread -
lm joinCubes-byValue-const.cpp cs225/Cube.cpp -lm -o joinCubes-byValue-co
nst
joinCubes-byValue-const.cpp:16:24: error: member function 'getVolume' not
 viable: 'this' argument has type 'const cs225::Cube', but function is no
t marked const
  double totalVolume = c1.getVolume() + c2.getVolume();
                       ^~
./cs225/Cube.h:9:14: note: 'getVolume' declared here
      double getVolume();
             ^
joinCubes-byValue-const.cpp:16:41: error: member function 'getVolume' not
 viable: 'this' argument has type 'const cs225::Cube', but function is no
t marked const
  double totalVolume = c1.getVolume() + c2.getVolume();
                                        ^~
./cs225/Cube.h:9:14: note: 'getVolume' declared here
      double getVolume();
             ^
2 errors generated.
Makefile:19: recipe for target 'joinCubes-byValue-const' failed
make: *** [joinCubes-byValue-const] Error 1
waf@siebl-2215-02:/mnt/c/Users/waf/Desktop/cs225/_lecture/05-parameters$
```

`const` as part of a member functions' declaration

Cube.h

```
1  #pragma once
2
3  namespace cs225 {
4    class Cube {
5      public:
6        Cube();
7        Cube(double length);
8        double getVolume();
9        double getSurfaceArea();
10
11     private:
12       double length_;
13   };
14 }
15
16
17
18
19
20
```

Cube.cpp

```
1  #include "Cube.h"
2  namespace cs225 {
3    Cube::Cube() {
4      length_ = 1;
5    }
6
7    Cube::Cube(double length) {
8      length_ = length;
9    }
10
11   double Cube::getVolume() const {
12     return length_ * length_ *
13            length_;
14   }
15
16   double
     Cube::getSurfaceArea() const {
17     return 6 * length_ *
18            length_;
19   }
20 }
21
```

joinCubes-byValue-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube c1, const Cube c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

Copy Constructor

[Purpose]:

All copy constructors will

Copy Constructor

Automatic Copy Constructor

Custom Copy Constructor

Cube.h

```
 1  #pragma once
 2
 3  namespace cs225 {
 4    class Cube {
 5      public:
 6        Cube();
 7        Cube(double length);
 8
 9        Cube(const Cube & other);
10
11        double getVolume() const;
12        double getSurfaceArea() const;
13
14      private:
15        double length_;
16    };
17  }
18
19
20
```

Cube.cpp

```
 7  namespace cs225 {
 8    Cube::Cube() {
 9      length_ = 1;
10      cout << "Default ctor"
             << endl;
11    }
12
13    Cube::Cube(double length) {
14      length_ = length;
15      cout << "1-arg ctor"
             << endl;
16    }
17
18
19
20
21
22
23
24
25
 …    // ...
```

joinCubes-byValue-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube c1, const Cube c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

Calls to constructors

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

joinCubes-byPointer-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube * c1, const Cube * c2) {
16   double totalVolume = c1->getVolume() + c2->getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(c1, c2);
33
34   return 0;
35 }
```

joinCubes-byRef-const.cpp

```
11 /*
12  * Creates a new Cube that contains the exact volume
13  * of the volume of the two input Cubes.
14  */
15 Cube joinCubes(const Cube & c1, const Cube & c2) {
16   double totalVolume = c1.getVolume() + c2.getVolume();
17
18   double newLength = std::pow( totalVolume, 1.0/3.0 );
19
20   Cube result(newLength);
21   return result;
22 }
23
24
25
26
```

```
28 int main() {
29   Cube *c1 = new Cube(4);
30   Cube *c2 = new Cube(5);
31
32   Cube c3 = joinCubes(*c1, *c2);
33
34   return 0;
35 }
```

MORE MONEY, MORE TIME!

Deadlines EXTENDED for the following scholarships:

September 12 John Deere WCS Scholarship ($2,000 x4!)

September 12 JPMorgan Chase WCS Scholarship ($2,500 x2!)

September 12 NEW! IMC Trading Scholarship ($7,500 x2!)

Find details at: http://go.cs.illinois.edu/AwardDeadlines

Questions? Contact Samantha at shendon@illinois.edu

ILLINOIS
Computer Science