CS 225

#5: Parameters

September 8, 2017

Our First Class – Sphere:

heap-puzzle3.cpp

```cpp
int *x = new int;
int &y = *x;

y = 4;

cout << &x << endl;
cout << x << endl;
cout << *x << endl;

cout << &y << endl;
cout << y << endl;
cout << *y << endl;
```

x	Type:	y	Type:
&x	Value:	&y	Value:
x	Value:	y	Value:
*x	Value:	*y	Value:

joinSpheres-byValue.cpp

```cpp
/*
 * Creates a new sphere that contains the exact volume
 * of the sum of volume of two input spheres.
 */
Sphere joinSpheres(Sphere s1, 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;
}
```

joinSpheres-byPointer.cpp

```cpp
Sphere joinSpheres(Sphere *s1, 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;
}
```

joinSpheres-byReference.cpp

```cpp
Sphere joinSpheres(Sphere &s1, 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;
}
```

	By Value	By Pointer	By Reference
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			
Safety			

Using the `const` keyword

1. [Function Parameters]:

joinSpheres-byValue-const.cpp

```
Sphere joinSpheres(const Sphere s1, const Sphere s2)
```

joinSpheres-byPointer-const.cpp

```
Sphere joinSpheres(Sphere const *s1, Sphere const *s2)
```

joinSpheres-byReference-const.cpp

```
Sphere joinSpheres(const Sphere &s1, const Sphere &s2)
```

Using the `const` keyword

2. [Classes]:

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H

class Sphere {
  public:
    Sphere();
    Sphere(double r);

    double getRadius();
    double getVolume();

    void setRadius(double r);

  private:
    double r_;
};

#endif
```

sphere.cpp

```
/* ... */
```

Big Idea: Copy Constructor

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H

class Sphere {
  public:
    Sphere();
    Sphere(double r);






    /* getRadius, getVolume,
       setRadius, and r_ */
```

sphere.cpp

```
/* ... */











/* ... */
```

Bringing Concepts Together:

How many times do our different joinSphere files call each constructor?

	By Value	By Pointer	By Reference
`Sphere()`			
`Sphere(double)`			
`Sphere(const Sphere &)`			

CS 225 – Things To Be Doing:

1. Register for Exam #1 (CBTF)
2. lab_debug due Sunday (11:59pm)
3. mp1 due Monday (11:59pm)
4. Complete POTDs