CS 225

#2: Classes + Pointers

August 30, 2017

Our First Class – Sphere:

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  class Sphere {
 5    public:
 6     double getRadius();
 7
 8
 9
10
11    private:
12
13
14  };
15
16  #endif
```

sphere.cpp

```
 1  #include "sphere.h"
 2
 3  double
      Sphere::getRadius() {
 4
 5
 6  }
 7
 8
 9
10
11
12
13
14
15
```

Public vs. Private:

Situation	Protection Level
Helper function used internally in `Sphere`	
Variable containing data about the `Sphere`	
`Sphere` functionality provided to client code	

Hierarchy in C++:

There `Sphere` class we're building might not be the only `Sphere` class. Large libraries in C++ are organized into _____.

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6    public:
 7      double getRadius();
...   /* ... */
```

sphere.cpp

```
 1  #include "sphere.h"
 2
 3  namespace cs225 {
 4    double
      Sphere::getRadius() {
 5      return r_;
 6    }
 7  }
```

Our first Program:

main.cpp

```
1  #include "sphere.h"
2  #include <iostream>
3
4  int main() {
5    cs225::Sphere s;
6    std::cout << "Radius: " << s.getRadius() << std::endl;
7    return 0;
8  }
```

...run this yourself: run `make main1` in the lecture source code.

Several things about C++ are revealed by our first program:

1. _____
   `main.cpp:4`

2. _____
   `main.cpp:5, main.cpp:1`

3. _____
   `main.cpp:6, main:cpp:2`

Simplify the Syntax

Often, we will find ourselves using significant functionality from a single library and typing `cs225::` or `std::` becomes burdensome. We can import an entire namespace into the global scope with:

_____

main.cpp

```
 1  #include "sphere.h"
 2  #include <iostream>
 3
 4  using namespace std;
 5  using namespace cs225;
 6
 7  int main() {
 8    Sphere s;
 9    cout << "Radius: " << s.getRadius() << endl;
10    return 0;
11  }
```

...run this yourself: run `make main2` in the lecture source code.

Big Idea: Constructor

Default Constructor:
Every class in C++ has a constructor – even if you didn't define one!

- Automatic Default Constructor:

- Custom Default Constructor:

sphere.h

```
…
4  class Sphere {
5    public:
6     Sphere();
…     /* ... */
```

sphere.cpp

```
…
3  Sphere::Sphere() {
4
5
6  }
…
```

Custom Constructors:
We can provide also create constructors that require parameters when initializing the variable:

sphere.h

```
…
4  class Sphere {
5    public:
6     Sphere(double r);
…     /* ... */
```

sphere.cpp

```
…
3  Sphere::Sphere(double r) {
4
5
6  }
…
```

INSIGHT PUZZLE – What happens when we run main.cpp?

main.cpp w/ above custom constructor

```
…
8   Sphere s;
9   cout << "Radius: " << s.getRadius() << endl;
…
```

…run this yourself: run **`make puzzle`** in the lecture source code.

INSIGHT PUZZLE – How do we fix it?

1. _____

2. _____

Pointers – Introduction
Besides classes, the other major component of C++ that will be used throughout all of CS 225 is the use of pointers. Pointers are:

- Extremely power, but extremely dangerous
- A **level of indirection** via memory to the data.

As a level of indirection via memory to the data:

1. _____

2. _____

The addition of a (*) to the end of the type denotes it a pointer:

```
Sphere s1;  // A variable of type Sphere
Sphere *s2; // A variable of type Sphere pointer
```

With a pointer type, members are accessed via the arrow operator (->) instead of the dot operator (.):

```
Sphere s1;  // A variable of type Sphere
cout << s1.getRadius() << endl;

Sphere *s2 = &s1;  // A variable of type Sphere pointer
cout << s2->getRadius() << endl;
```

CS 225 – Things To Be Doing:

1. Lab Sections – Wednesday / Thursday / Friday
2. MP1 will be released this Friday, due Monday, Sept. 11
3. Visit Piazza and the course website often!