CS 225

#2: Classes and Reference Variables

January 19, 2018 · *Wade Fagen-Ulmschneider*

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 main` and `./main` 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`
4. However, our program is unreliable. **Why?**

Default Constructor:

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

- Automatic Default Constructor:

  1. Provided automatically if we don't define a constructor
  2. initializes all the variables to default value
  3. has zero parameters

- Custom Default Constructor:

sphere.h

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

sphere.cpp

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

Custom, Non-Default 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  }
...
```

Puzzle #1: How do we fix our first program?

main.cpp w/ above custom constructor

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

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

Solution #1:

Solution #2:

The beauty of programming is both solutions work! There's no one right answer, both have advantages and disadvantages!

Pointers and References – Introduction
A major component of C++ that will be used throughout all of CS 225 is the use of references and pointers. References and pointers both:

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

As a level of indirection via memory to the data:

1. Creating an alias to existing data
2. Multiple variables may modify the same memory

Often, we will have direct access to our object:

```
Sphere s1;  // A variable of type Sphere
```

Occasionally, we have a reference or pointer to our data:

```
Sphere & s1;  // A reference variable of type Sphere
Sphere * s1;  // A pointer that points to a Sphere
```

Reference Variable
A reference variable is an alias to an existing variable. Modifying the reference variable modifies the variable being aliased. Internally, a reference variable maps to the same memory as the variable being aliased:

main-ref.cpp

```
3  int main() {
4    int i = 7;
5    int & j = i;   // j is an alias of i
6
7    j = 4;                              // j and i are both 4.
8    std::cout << i << " " << j << std::endl;
9
10   i = 2;                              // j and i are both 2.
11   std::cout << i << " " << j << std::endl;
12   return 0;
13 }
```

...run this yourself: run **`make main-ref`** and **`./main-ref`** in the lecture source code.

Three things to note about reference variables:

1. Always contains a reference to data (cannot be `NULL`)
2. Never creates new memory
3. reference variables are defined when initialized and reference cannot be changed

CS 225 – Things To Be Doing:

1. Sign up for "Exam 0" (starts Tuesday, Jan. 23rd)
2. Complete lab_intro; due Sunday, Jan. 21st
3. MP1 released today; due Monday, Jan. 29th
4. Visit Piazza and the course website often!