#2: Classes and Reference Variables

January 19, 2018 · *Wade Fagen-Ulmschneider*

Our First Class – Sphere:

sphere.h

```
#ifndef SPHERE_H_
#define SPHERE_H_

class Sphere {
  public:
   double getRadius();




  private:


};

#endif
```

sphere.cpp

```
#include "sphere.h"

double
  Sphere::getRadius() {


}
```

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

```
#ifndef SPHERE_H
#define SPHERE_H

namespace cs225 {
  class Sphere {
  public:
    double getRadius();
  /* ... */
```

sphere.cpp

```
#include "sphere.h"

namespace cs225 {
  double
  Sphere::getRadius() {
    return r_;
  }
}
```

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:

- Custom Default Constructor:

sphere.h

```
class Sphere {
  public:
   Sphere();
   /* ... */
```

sphere.cpp

```
Sphere::Sphere() {


}
```

Custom, Non-Default Constructors:

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

sphere.h

```
class Sphere {
  public:
   Sphere(double r);
   /* ... */
```

sphere.cpp

```
Sphere::Sphere(double r) {


}
```

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

main.cpp w/ above custom constructor

```
  Sphere s;
  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. ______________________________

2. ______________________________

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

```
int main() {
  int i = 7;
  int & j = i;   // j is an alias of i

  j = 4;                           // j and i are both 4.
  std::cout << i << " " << j << std::endl;

  i = 2;                           // j and i are both 2.
  std::cout << i << " " << j << std::endl;
  return 0;
}
```

...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!