CS 225

Data Structures

Destructor

[Purpose]:

Destructor

[Purpose]: Free any resources maintained by the class.

Automatic Destructor:

1. Exists only when no custom destructor is defined.

2. [Invoked]:

3. [Functionality]:

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6      public:
 7        Sphere();
 8        Sphere(double r);
 9        Sphere(const Sphere &s);
10        ~Sphere();
11
12
13
14
 …        // ...
26      private:
27        double r_;
28
29    };
30  }
31
32  #endif
```

sphere.cpp

```
 1  #include "sphere.h"
 2
 3  namespace cs225 {
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
 …      // ...
    }
```

Operators that can be overloaded in C++	
Arithmetic	+ - * / % ++ --
Bitwise	& | ^ ~ << >>
Assignment	=
Comparison	== != > < >= <=
Logical	! && ||
Other	[] () ->

sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6      public:
 7        Sphere();
 8        Sphere(double r);
 9        Sphere(const Sphere &s);
10        ~Sphere();
11
12
13
14
 …        // ...
26      private:
27        double r_;
28
29    };
30  }
31
32  #endif
```

sphere.cpp

```
 1  #include "sphere.h"
 2
 3  namespace cs225 {
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
 …      // ...
    }
```

One Very Special Operator

Definition Syntax (.h):

```
Sphere & operator=(const Sphere& s)
```

Implementation Syntax (.cpp):

```
Sphere & Sphere::operator=(const Sphere& s)
```

Assignment Operator

Similar to Copy Constructor:

Different from Copy Constructor:

Assignment Operator

	Copies an object	Destroys an object
Copy constructor		
Assignment operator		
Destructor		

The “Rule of Three”

If it is necessary to define any one of these three functions in a class, it will be necessary to define all three of these functions:

1.

2.

3.

Inheritance

Planet.h

```
1  #ifndef PLANET_H_
2  #define PLANET_H_
3
4  #include "Sphere.h"
5
6  public Planet :
7     public cs225::Sphere {
8
9  }
10
11 #endif
12
13
14
15
16
17
18
19
20
21
22
```

class

Planet.cpp

```
1  #include "Planet.h"
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

Derived Classes

[Public Members of the Base Class]:

```
5  int main() {
6    Planet p;
7    p.getRadius(); // Returns 1, the radius init'd
8                   // by Sphere's default ctor
…    ...
…  }
```

[Private Members of the Base Class]:

Planet.h

```
1  #ifndef PLANET_H_
2  #define PLANET_H_
3
4  #include "Sphere.h"
5
6  public Planet :
7     public cs225::Sphere {
8    public:
9
10
11
12
13
14
15
16
17   private:
18
19
20 }
21
22 #endif
```

Planet.cpp

```
1  #include "Planet.h"
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

Virtual

Sphere.cpp

```
1  Sphere::print_1() {
2    cout << "Sphere" << endl;
3  }
4
5  Sphere::print_2() {
6    cout << "Sphere" << endl;
7  }
8
9  virtual Sphere::print_3() {
10   cout << "Sphere" << endl;
11 }
12
13 virtual Sphere::print_4() {
14   cout << "Sphere" << endl;
15 }
16
17 // In .h file:
18 virtual Sphere::print_5() = 0;
19
20
21
22
```

Planet.cpp

```
1  // No print_1() in RedBall.cpp
2
3
4
5  Planet::print_2() {
6    cout << "Earth" << endl;
7  }
8
9  // No print_3() in RedBall.cpp
10
11
12
13 Planet::print_4() {
14   cout << "Earth" << endl;
15 }
16
17 Planet::print_5() {
18   cout << "Earth" << endl;
19 }
20
21
22
```

Runtime of Virtual Functions

	`Sphere obj;`	`Planet obj;`	`Planet r;` `Sphere &obj = r;`
`obj.print_1();`			
`obj.print_2();`			
`obj.print_3();`			
`obj.print_4();`			
`obj.print_5();`			

CS 225 – Things To Be Doing

Theory Exam 1 is ongoing

More Info:
https://courses.engr.illinois.edu/cs225/sp2018/exams/

MP2 Released!

Extra Credit Deadline: Monday, Feb. 5th – Up to +7 Extra Credit
Due Date: Monday, Feb.12th

lab_memory

Due: Sunday, Feb. 4th

POTD

Every Monday-Friday – *Worth +1 Extra Credit /problem (up to +40 total)*