CS 225

#7: Inheritance

January 31, 2018 · *Wade Fagen-Ulmschneider*

Destructor

The last and final member function called in the lifecycle of a class is the destructor.

Purpose of a **destructor**:

The **automatic destructor**:

1. Like a constructor and copy constructor, an automatic destructor exists only when no custom destructor is defined.
2. [Invoked]:
3. [Functionality]:

Custom Destructor:

sphere.h

```
5  class Sphere {
6    public:
7      Sphere();           // custom default ctor
8      Sphere(double r);   // 1-param ctor
9      Sphere(const Sphere & other);   // custom copy ctor
10     ~Sphere();          // destructor, or dtor
11     ...
```

...necessary if you need to delete any heap memory!

Overloading Operators

C++ allows custom behaviors to be defined on over 20 operators:

Arithmetic	`+ - * / % ++ --`
Bitwise	`& | ^ ~ << >>`
Assignment	`=`
Comparison	`== != > < >= <=`
Logical	`! && ||`
Other	`[] () ->`

General Syntax:

Adding overloaded operators to Sphere:

sphere.h

```
1  #ifndef SPHERE_H
2  #define SPHERE_H
3
4  class Sphere {
5    public:
…      // ...
17
18
19
20
…      // ...
```

sphere.cpp

```
…  /* ... */
10
11
12
13
14
15
16
17
18
…  /* ... */
```

One Very Powerful Operator: Assignment Operator

sphere.h

```
Sphere & operator=(const Sphere & other);
```

sphere.cpp

```
Sphere & Sphere::operator=(const Sphere & other) { ... }
```

Functionality Table:

	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

In nearly all object-oriented languages (including C++), classes can be extended to build other classes. We call the class being extended the **base class** and the class inheriting the functionality the **derived class**.

Derived Class: `Planet`

Planet.h

```
1  #ifndef PLANET_H_
2  #define PLANET_H_
3
4  #include "Sphere.h"
5
6  class Planet : public cs225::Sphere {
7     // Empty!
8  };
9
10 #endif
```

In the above code, `Planet` is derived from the base class `Sphere`:

- All public functionality of `Sphere` is part of `Planet`:

planet-main.cpp

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

- [Private Members of `Sphere`]:

Adding New Functionality:

Planet.h

```
6  class Planet : public cs225::Sphere {
7    public:
8      Planet(std::string name, double radius); // ctor
9    private:
10     std::string name_;
11 };
```

Functions: non-virtual, virtual, and pure virtual

- The **virtual** keyword:

Sphere.cpp

```
Sphere::print_1() {
  cout << "Sphere" << endl;
}

Sphere::print_2() {
  cout << "Sphere" << endl;
}

virtual Sphere::print_3() {
  cout << "Sphere" << endl;
}

virtual Sphere::print_4() {
  cout << "Sphere" << endl;
}

// .h:
virtual Sphere::print_5() = 0;
```

Planet.cpp

```
// No print_1() defined
// in Planet


Planet::print_2() {
  cout << "Earth" << endl;
}

// No print_3() defined
// in Planet


Planet::print_4() {
  cout << "Earth" << endl;
}

Planet::print_5() {
  cout << "Earth" << endl;
}
```

	`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:

1. Theory Exam #1 is ongoing – ensure you're signed up!
2. Attend and complete lab_memory *(due Sunday)*
3. MP2 is ongoing *(extra credit due Monday)*
4. Daily POTDs every M-F for daily extra credit!