IIT Bo

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: More on Constructors

Quick Recap of Relevant Topics

- Object-oriented programming with structures and classes
- Data members and member functions
- Accessing members and controlling access to members
- Constructor and destructor functions

Overview of This Lecture

IIT Bo

- Closer look at constructors
- Example usage in C++ programs

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

- Examples taken from this book are indicated in slides by th citation **AGRBook**

Recap: Constructor and Destructor Functions

IIT Bo

- **Constructor:** Invoked **automatically** when an object of the class is allocated
  - Object is allocated first, then constructor is invoked on object
  - Convenient way to initialize data members
- **Destructor:** Invoked **automatically** when an object of the class is de-allocated
  - Destructor is invoked on object first, then object is de-allocated
  - Convenient way to do book-keeping/cleaning-up before de-allocating object

Recap: Constructors/Destructor of Class V3

```
class V3 { private: double x, y, z;
                    double length() { … }
  public:
    V3 (double vx, double vy, double vz) {
      x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
    ~V3() { if (length() == 0.0) {
                cout << "Zero vector!!!";
              return;
     }
    … Other member functions of V3 …
};
```

Constructor functions

Destructor function

Recap: Invoking Member Functions

- Member functions of a class can be usually invoked explicitly on a receiver object of the same class

```
class V3 {
  private: double x, y, z;
                double length() { … }
  public:
    V3 scale(double const factor) { … }
    void printLength() { … }
    … Other member functions …
};
```

```
int main() {
   V3 a (1.0, 2.0, 3.0);
   V3 b = a.scale(4.0);
    b.printLength();
    return 0;
}
```

Recap: Invoking Member Functions

- Member functions of a class can be usually invoked explicitly on a receiver object of the same class

```
class V3 {
  private: double x, y, z;
                double length() { … }
  public:
   V3 scale(double const factor) { … }
   void printLength() { … }
   … Other member functions …
```

```
int main() {
   V3 a (1.0, 2.0, 3.0);
   V3 b = a.scale(4.0);
    b.printLength();
    return 0;
}
```

Can we do the same with constructors and destructors

Invoking Constructors/Destructors Explicitly

- Usually **an error** to call a destructor explicitly in a program
- **OK** to call a constructor explicitly in a program

```
int main() {
        V3 a(1.0, 1.0, 1.0);
        V3 b = a.sum( V3(2.0, 2.0, 2.0) );
        a.printLength(); b.printLength();
        return 0;
 }
```

Explicit constructor invokation

Looks like normal function ca

Name of (constructor) function is name of class

Creates temporary object an invokes constructor on it

Invoking Constructors Explicitly [Ref. AGRBook]

IIT Bo

- An interesting implementation of “sum” in class V3

```
class V3 {
   private: double x, y, z;
   public:
    … Other members, constructors, destructor …
    V3 sum (V3 const &b) {
        return V3(x+b.x, y+b.y, z+b.z);
     }
};
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of the following lead to correct constructor calls

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

Equivalent to V3 a(0.0, 1.0, 2.0);

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

Equivalent to V3 b(1.2, 1.0, 2.0);

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

Equivalent to V3 c(1.2, 1.3, 2.0);

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Equivalent to V3 d(1.2, 1.3, 1.4);

Initialization Lists

Specifies how different data members of the receiver object are initialized before execution of the constructor begins

```
class V3 {
  private: double x, y, z;
  public:
    V3(double vx, double vy, double vz) : x(vx), y(vy), z(vz) {
       … Rest of constructor code comes here …
     }
     … Other member functions of class V3 …
};
```

Note the :

Initialization List

Initialization Lists

Specifies how different data members of the receiver object are initialized before execution of the constructor begins

```
class V3 {
  private: double x, y, z;
  public:
    V3(double vx, double vy, double vz) : x(vx), y(vy), z(vz) {
       … Rest of constructor code comes here …
     }
     … Other member functions of class V3
};
```

Note the :

Initialization List

Can be any expression in vx, vy, vz

An Interesting Example [Ref. AGRBook]

IIT Bo

```
class Point { private: double x, y;
                public: Point (double p, double q) {x = p; y = q; retu
};

class Disk { private: Point center; double radius;
                public: Disk(double x, double y, double r) :
                            center(Point(x, y)), radius(r) { return; }
};
```

Summary

- Closer look at constructor functions
  - Invoking constructors explicitly
  - Specifying default values of parameters
  - Initialization lists