IIT Bombay

Computer Programming

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

Session: Friends and Static Members

Quick Recap of Relevant Topics

IIT Bombay

- Object-oriented programming with structures and classes
- Accessing data members and member functions
- Constructors and destructors
- Function calls with structures/classes
- Operator overloading

Overview of This Lecture

- Friend classes and functions
- Static data members and static member functions

Acknowledgment

IIT Bombay

- 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 the citation **AGRBook**

Friend Functions

- Normally, “private” members of a class are accessible only to member functions of the class
  - Data encapsulation or hiding
- Occasionally it may be desirable to bypass this access restriction for a few specific non-member functions
  - Should these functions be made members of the class?
  - Should we make all members of the class public?
- C++ provides a better solution:

A “friend” declaration allows a class to explicitly allow specific non-member functions to access its private members

Friend Functions

```
class Point { private: double x, y;
  public:
    … Member functions …
 };
```

```
bool collinear(Point &p1, Point &p2, Point &p3) {
   // Not a member of class Point
   double temp;
   temp = p1.x*(p2.y – p3.y) + p2.x*(p3.y – p1.y) + p3.x* (p1.y – p2.y);
   return (temp == 0);
}
```

Friend Functions

```
class Point { private: double x, y;
  public:
  friend bool collinear(Point &p1, Point &p2, Point &p3);
```

Can be in public or private section of class Point

```
bool collinear(Point &p1, Point &p2, Point &p3) {
  // Not a member of class Point
  double temp;
  temp = p1.x*(p2.y – p3.y) + p2.x*(p3.y – p1.y) + p3.x* (p1.y – p2.y);
  return (temp == 0);
}
```

Friend Functions

```
class Point { private: double x, y;
  friend bool collinear(Point &p1, Point &p2, Point &p3);
   public:
    ... Member functions ...
 };
```

```
bool collinear(Point &p1, Point &p2, Point &p3) {
   // Not a member of class Point
   double temp;
   temp = p1.x*(p2.y – p3.y) + p2.x*(p3.y – p1.y) + p3.x* (p1.y – p2.y);
   return (temp == 0);
}
```

Friend Functions

- In general,

  A function **func** can be "friend" of several classes **C1, C2, …**

  func can access private members of classes **C1, C2, …**

  A class **C** can have several "friend" functions **func1, func2, ...**

  Each of func1, func2, … can access private members of **C**

Friend Classes

- Various members of class **C1** may need access to private members of class **C2**

```
class Point { private: double x, y;
  public:
    … Member functions …
 };
```

```
class PointsInPlane { private: int numPoints; Point pointArray[100];
  public: bool collinear(Point &p1, Point &p2, Point &p3) { … }
          bool isEquiLateral(Point &p1, Point &p2, Point &p3) { … }
          … Other member functions …
};
```

Friend Classes

- Entire class **C1** can be declared "friend" of class **C2**

```
class Point { private: double x, y;
   public:
    friend class PointsInPlane;
     … Member functions …
 };
```

```
class PointsInPlane { private: int numPoints; Point pointArray[100];
   public: bool collinear(Point &p1, Point &p2, Point &p3) { … }
           bool isEquiLateral(Point &p1, Point &p2, Point &p3) { … }
           … Other member functions …
};
```

Static Data Members [Ref. AGRBook]

IIT Bombay

```
class Point {
  private: double x, y;
  public:
    static int count;
    Point() { count++; return; }
    Point(double a, double b) {
      x = a; y = b; count++; return;
    }
};

int Point::count = 0;
```

C++ keyword

Static Data Members [Ref. AGRBook]

IIT Bombay

```
class Point {
   private: double x, y;
   public:
     static   int count;
     Point() { count++; return; }
     Point(double a, double b) {
        x = a; y = b; count++; return;
     }
};

int Point::count = 0;
```

Declaration of static public data member

Single copy of static data member "count" shared across all objects of class Point

Inside class Point, referred to as simply "count"

Static Data Members [Ref. AGRBook]

IIT Bombay

```
class Point {
  private: double x, y;
  public:
    static   int count;
    Point() { count++; return; }
    Point(double a, double b)
      x = a; y = b; count++; return;
    }
};

int Point::count = 0;
```

Referring to member count of class Point

Note use of scope resolution operator ::

Necessary when referring to a member outside the class definition

Static Data Members [Ref. AGRBook]

IIT Bombay

```
class Point {
   private: double x, y;
   public:
     static   int count;
     Point() { count++; return; }
     Point(double a, double b) {
        x = a; y = b; count++; return;
     }
};

int Point::count = 0;
```

Creation and initialization of static public data member

Note this is not tied to creation of objects of class Point

Static Data Members [Ref. AGRBook]

IIT Bombay

```
class Point {
   private: double x, y;
   public:
     static   int count;
     Point() { count++; return; }
     Point(double a, double b) {
        x = a; y = b; count++; return;
     }
};

int Point::count = 0;
```

```
int main () {
  Point a, b, c(0.0, 0.0);
  cout << “Count of points “;
  cout << Point::count << endl;
  return 0;
}
```

All constructor calls update the same static data member. So this counts the number of points created.

Static Data Members [Ref. AGRBook]

```
class Point {
   private: double x, y;
   public:
     static   int count;
     Point() { count++; return; }
     Point(double a, double b) {
        x = a; y = b; count++; return;
     }
};

int Point::count = 0;
```

```
int main () {
  Point a, b, c(0.0, 0.0);
  cout << “Count of points “;
  cout << Point::count << endl;
  return 0;
}
```

Accessing count outside the class Point requires scope resolution operator

Static Member Functions [Ref. AGRBook]

IIT Bombay

```
class Point {
  private:
    double x, y;
    static  int count;
  public:
   Point() { count++; return; }
   Point(double a, double b) {x = a; y = b; count++; return;}
   static void resetCount() { count = 0; return; }
   void printCount() {cout << count << endl; return;}
};
int Point::count;
```

Declaration of static private data member

Creation of static private data member

Static Member Functions [Ref. AGRBook]

```
class Point {
  private:
    double x, y;
    static  int count;
  public:
   Point() { count++; return; }
   Point(double a, double b) {x = a; y = b; count++; return;}
   static void resetCount() { count = 0; return; }
   void printCount() {cout << count << endl; return;}
};
int Point::count;
```

Declaration of static public member function

Declaration of non-static public member function

Use of Static Member Functions

IIT Bombay

Static member function not invoked on object of class Point

```
int main () {
  Point::resetCount();
  Point a, b, c(0.0, 0.0);
  cout << “Count of points “;
  ~~cout << Point::count << endl~~;
   a.printCount();
  return 0;
}
```

Invocation of static public member function in “main”

Requires scope resolution operator

Summary

IIT Bombay

- Friend functions and friend classes and their usage
- Static data members, static member functions and their usage