More on C++ Code-Level Design Features

What is a ‘friend’ in C++

Friend functions and classes

- The following components of a program can be friends to a class:
  - A global function, visible by a class.
  - A member function of other classes in the program, visible to the class.
  - Another class, visible by a class

Example of Global Function as a Friend

- C++ allows to declare a global function as friend of a class.
- Means function will have access to the the private members of a class. Example:

```
// file a.h
void f();

class A{
      int a;
      friend void f();
      public:
      A();
      void print();
};
```

Example of a Class Being Friend Another Class

- C++ allows to declare a class as a friend of another class.
- Means the the declared class will have access to the the private members of a other class. Example:

```
// file: b.h
class B{
private:
    int b;
    friend class A;
public:
    B();
    void print();
    void fun();
};
```

Example of member function of another class to be friend of class

- A C++ class can declare the member function of another class as a friend
- Means only that member function has access to the private member, not the entire class.
- In the following example, member function B::fun is declared as a friend for class C

```
#include "b.h"
class C {
private:
    int c;
public:
     C();
     friend void B::fun();
     void print();
};
```

Questions to be answered

- **Where should the fiend declaration appear? In private, or public part of a class?**
- **How can we make class B declared as a friend of class A, while the class B's definition appears after the class A?**
- **Why shouldn't we make every function and every class friend of each other?**
- **What is a good example of legitimate case for making a class friend of another class?**

- **Answers to the questions will be discussed during the lecture session.**

Static Members in C++

Static Members

- Static data member definition and its initialization should happen in the implementation part (.cpp file). However, if it is not explicitly initialized will be implicitly initialized to zero.
- A member function that accesses only a static member function of a class may also be declared as static.
- A static member function does not have a “this” pointer.
- A static member function cannot be a const member
- A static member function may be invoked through a class object or pointer to a class or can be accessed directly even if no class object is declared.

Static Members

- A static class member acts as a global object among the objects of the same class
- Information hiding can be still enforced
- A static data member is not entered into programs global name space.
- A data member is made static by prefixing its declaration by *static* keyword.
- A static data member is initialized outside the class definition.
- A static data member can be a constant or a class object.

Static Members Example

```
// point.h

class Point
{
  private:
        double xcoordinate;
        double ycoordinate;
        int pointID;
        static int counter;    // incomplete type
  public:
        Point(): xcoordinate(0), ycoordinate(0) { counter++; }
        static int getCounter() { return counter;}
};
```

Using Static Members

```
// main.cpp

int main()
{
   Point a, b;
   …
   // output is 2.
   cout<<
   Point::getCounter();
   …
   …
}
```

```
// point.cpp
// static member definition
int Point::counter=0;
```

Notice: Point::counter will be automatically initialized to zero. Or we can initialize it to any other number. For example:

int Point::counter = 1000;

Class Discussion

How can we make objects of a class,
as a static data member

Let's assume we need to make an object of class Colour as a static data member of class Point. Please notice that class Colour has a ctor.

```
class Colour {
  private:
            int g;
          // assume more data members
  public:
            Colour(int a): g(a) {}     // ctor for class Colour.
          // assume more function
};

class Point{
   private:
            double xcoordinate;
            double ycoordinate;
            int pointID;
   public:
            Point(): xcoordinate(0), ycoordinate(0) { counter++; }
};
```

Answer will be discussed during the lecture.