

static data member a data member for which one copy is shared by all objects of a class

Some common uses:

- instance counters
- class-wide attribute settings
- class-wide associations

A static member is simply declared with the reserved word `static`.

static function member a member function that can be called independently of the existence of any class object

Static member functions are only allowed to access data members that are static. Why?

The class below provides a logging facility that depends on an association to an output stream.

We want the client to be able to supply the stream, and to turn logging on and off.

We want all objects of this class to be synchronized in the sense that either all log or none logs, and that all log information goes to the same destination.

```
class printQueue {
private:
    . . .
    static std::ostream* Log;    // association to log stream
    static bool loggingOn;      // switch for log operation

public:
    . . .
    static void setLog(std::ostream* logStream);
    static bool turnLoggingOn();
    static bool turnLoggingOff();
};
```

Static data members should not be initialized by a constructor. Why?

Instead, they are initialized within the implementation file for the class, at the file-scoped level:

```
// printQueue.cpp
#include "printQueue.h"
using namespace std;

ostream* printQueue::Log          = NULL;
bool     printQueue::loggingOn    = false;
. . .
```

Note that any class member function can access, or modify, a static data member.

Static member functions are implemented just like any other member function:

```
bool printQueue::turnLoggingOn() {  
  
    if ( Log != NULL )  
        loggingOn = true;  
    else  
        loggingOn = false;  
    return loggingOn;  
}
```

There's nothing here to indicate the function is `static`.

Note how the implementation is designed to guarantee the client cannot turn logging on until it has supplied a pointer to an output stream.

The client uses the static functions to manage logging:

```
. . .  
printQueue::setLog( &cout );      // could use any stream  
. . .  
printQueue::turnLoggingOn();      // begin logging  
. . .  
printQueue::turnLoggingOff();     // end logging
```

The client has complete control of where and when information will be logged by objects of the class.

The class need only provide logging code with a conditional check against the `static bool` member used to toggle it on and off.

Of course, the class also needs to be careful about NULL pointers.