

A CLOSE Look at the Anatomy of User-defined Stream IO

A function that is declared as a 'friend' is a function inside a class that is NOT a member function, but can access the private member data in that class.

The function's name, `operator<<` (note no space) denotes that we are overloading the `<<` operator. This will work for many other operators as well, such as `+`, `-`, `=`, `==`, etc. We'll be coming back to this!

The declaration:

```
friend ostream & operator<<(ostream& s, const classType& instanceName);
```

The overloaded `<<` operator needs to return something that the `cout` object will understand. `Cout` works with `ostream`s, so hence we will return a reference to an `ostream`.

Parameter two is the instance of the class we are overloading `<<` for. Note that it is also a reference, 'cause it could be a big data structure we don't want to waste space copying. However, it has a `const` on it so we don't change it by accident.

Parameter one is a reference to the current `ostream` being outputted. This means we can change its contents by outputting to it, because it's a reference.

The Implementation

```
friend ostream & operator<<(ostream& s, const classType& instanceName){  
    // output the class data to ostream s in any way we need to  
    s << "Data Field 1 is: " << instanceName.data1 << " , and Data Field  
    2 is: " << instanceName.data2;  
    return s; // important!  
}
```

The Different Flavors of Memory in a Program

A short test:

Which of the three ways of using memory we discussed in class (Static, Automatic, and Dynamic) are each of the marked lines in the following code?

```
// Main.cpp
// Another useless program by Steve Martin
// 7-9-01

int evilGlobal = 1; // — 1 — //

void foo (int a, int b) { // — 2 — //
    a = b;
}

int main ( ) {

    int *pointer = new int; // — 3 — //

    int steve = 99; // — 4 — //

    foo(*pointer, steve);

    return 0;
}
```

Answers:

1. Static
2. Automatic
3. Dynamic
4. Automatic