CS11600: Introduction to Computer Programming (C++)

Lecture 10

Svetlozar Nestorov
University of Chicago

Outline

- Static objects
- Static data members
- Static functions
- Friend classes
- Read about nested and local classes (5.9, 5.10).

Static Objects

- Two kinds of static objects: internal and external.
- Internal: within a function.
- External: in a file, outside of any function.

Internal Static Objects

- An internal static objects is created once for a program run when the statement that declares it is *first* executed.
- All subsequent times, the same memory location and current value are used.
- Why use internal static objects?
  - Faster programs
  - Counters
  - Better alternative to global objects

Example

```
void drink() {
  static int count(0);
  static const int max = 5;

  if (count >= max) {
      cout << “Sorry, too many drinks!”;
      return;
  }
  count++;
  /* Do the drinking - cheers! */
  …
}
```

External Static Objects

- An external static object is created once before main() is called.
- An external static object is accessible only within the file that defines it.
- Alternative to globals, no risk of name conflict.

Static Data Members

- A static data member of a class is *shared* by all instances of this class.

```
static Type name;
```

- Initialization syntax:

```
Type Class::name = value;
const Type Class::name(value);
```

- Must appear only once, not within a function.

Usage

- Alternative to globals and external static objects.
- Encapsulation principle – hide everything as deep as possible:
  - Make a constant visible only within the class that uses it, not the file that defines the class.
- Counters: keep track of number of objects of a given class.

Static Member Functions

- Primarily used to access static data members.
- Can be called without an object.
- Declaration syntax:

```
static Type name(arguments);
```

- Call syntax:

```
Class::name(arguments);
```

- Can also be called with object notation (`.` or `->`).
- Static member functions have access *only* to static data members.

Friend Classes

- Allow private data members of a class to be accessed directly by a friend class objects.
- Friend concept is *not* symmetric in C++!
- Why do we need friend classes?

```
class Name1 {
   friend class Name2;
   …
}
```