CS 216

Lecture 7

February 24th, 2014
Administrivia
So where are we...
~20 source files
1 makefile
2 executables
PA1.2 deadline extended: Tuesday at midnight.
PA1 Post-mortem
Why?
CS215: Programming
CS216: Engineering
This means scale.
Things to learn from PA1
Provided code
Reading and modifying code is a crucial skill.
Large(ish) project
Automation
Benefits of code reuse
Requirements focused on result, not process.
No singular solution
Visual Studio

vs.

g++
Platform is not important.
New stuff
static methods
A static method can be called without an instantiated object, but has no access to any member variables.
// In the header file
class DungeonLevel
{
    public:
        static DungeonLevel * generateRandomLevel();

        // Other declarations and such
};

// In the .cpp file, it's otherwise normal
DungeonLevel * DungeonLevel::generateRandomLevel()
{
    // Implementation goes here
}

// And then elsewhere with code
int main(int argc, char * argv[])
{
    DungeonLevel * pLevel = DungeonLevel::generateRandomLevel();
}
So then, what’s the point?
It avoids polluting the global namespace.
Static methods can access private methods and variables from objects of the class.
For example, private constructors.
Testing
The big question: How do we know code is correct?
System testing
Unit testing
Regression testing
You’ve done system testing.

Quite a bit, probably.
Run a program, see if it works.
Unit testing is the interesting one.
The idea is to isolate the specific code and test only that code.
This can be done in a few different ways, but mostly it’s done by isolating specific classes.
Regression testing is the next step.
PA2
We need to be able to generate dungeon levels.
Each level will be a two dimensional vector of tiles.
And we’ll have requirements for what should be in a level.
(e.g., a certain number of room spaces, all rooms connected, and one up stairway and one down stairway)
And then, for the fun part – you will need to write a unit test to verify that the code is correct.