COMSC-200 Lecture Topic 0
Course Orientation

Introduction
Object Oriented Programming C/C++
Prereq: COMSC-165 (formerly 265)
  make sure you can do Ch.1-8
  self-review exercises
You will learn...
  C++ OOP Syntax (C++99)
  Problem Solving with OOP
  Intro to the STL
About the instructor
  Ph.D. Purdue, Mech.Engr.
  programmer since 1969
class website http://cs.dvc.edu
this is NOT an online class
  attend lecture
  attend lab

Course Goals
Prepare for COMSC-210, Data Structures
Prepare for xfer to UC/CSU COMSC program
Solve shortest-route problem; Simulate elevators;
  Learn GUI programming
Grading policy
  redos, late work, sequence
12 hours per week *(3 lec + 3 lab + 6)*
Learning process: read, quiz, lecture, lab, project
Syllabus and Course Outline (with schedule)
  academic honesty policy

Lecture Period
sign the sign-in sheet (look for "see me")
no electronics during lecture (except photos)
lectures recorded (MP3), files posted
command-line compiling used in lecture
  "lowest common denomiator"
  vendor-, system-, and compiler-independent

Lab Period
Microsoft Visual C++
  on PCs in ATC, ET, and L bldgs (*USB drive recommended*)
  Express 2013 for Web version available for free download
  IDE mode: static library, no precompiled headers
g++ Mac download ; PC MinGW ; PC Cygnus (see instructor)
Upload **.cpp** source files to the COMSC server
  workInProgress, Google Drive, OneDrive, Dropbox
free resources
  DreamSpark accounts
  onedrive.live.com

Textbooks
"C++: How To Program: Late Objects Version" by Deitel
  other editions may be suitable
  click HERE for TOC
we cover ch.9 and beyond
other required reading:
  the lecture notes
  www links in the lecture notes
optional: Burns "Intro To Programming Using C++ and Java"
  basic C++ reference

Lab Assignments
Highly structured assignments, 1 per week
Strict about coding practices
  debugging techniques
  alignment and indenting
about "redos"
  no grade until work is complete and correct
  2 point penalty for not following instructions
  no point penalty for "learning opportunities"

just because your program works for you,
  it does *not* mean that
  you did it right!
programming conventions and lab 0

Matters of Style
my prototypes: nameless parameters
```
float getAvg(int, int*);
```
my pointers and refs: trailing `*` and `&`
```
node* start = 0;
```
I mix `endl` and `\n`
I indent with spaces instead of tabs
I don't use `enum`
I don't use `return 0;` anymore

Programming Conventions
mostly C++99, with C++11 auto specifier and range for-loops

do *not* use `using namespace std;`
  use (e.g.) `using std::cout;`
distinguish between C libraries and C++ libraries
  C library names start with "c"
  C libraries to *not* use `using std::`'s

do *not* read `int`s or `double`s directly from console

```
// do NOT use this anymore
int x;
double y;
cin >> x >> y;

// use THIS instead -- requires cstdlib and string
int x;
double y;
char buf[100];
cin >> buf; x = atoi(buf); // requires #include <cstdlib>
cin >> buf; y = atof(buf); // requires #include <cstdlib>
```

accommodate "round-off error"
  ...in calcs using `double`s

```
// do NOT use this anymore (for example)
double y = ...; // result of a calculation
if (y == 100)

// use THIS instead (for example)
if (99.999 < y && y < 100.001)
```

g++ command line compiling with C++11
```
g++ -std=c++11 hello.cpp
```

Parameters
"mutable copy" `void fun(Time);` or `int...`

Console I/O Formatting

formatting numeric output

```
cout.setf(ios::fixed|ios::showpoint);
cout << setprecision(2) ...
```

console and file I/O PDF

`cout.setf(ios::left, ios::adjustfield);` or `left` manipulator
`cout.setf(ios::right, ios::adjustfield);` or `right` manipulator

Changes From Comsc-110

do *not* use `using namespace std;`
use (e.g.) `using std::cout;`
distinguish between C libraries and C++ libraries
C library names start with "c"
C libraries do *not* use `using std::`'s

do *not* read `int`s or `double`s directly from console

```
// do NOT use this anymore
int x;
double y;
cin >> x >> y;

// use THIS instead: "string buffer method"
int x;
double y;
char buf[100];
cin >> buf; x = atoi(buf); // requires #include <cstdlib>
cin >> buf; y = atof(buf); // requires #include <cstdlib>
```

"immutable copy" `void fun(const Time);`
"mutable reference" `void fun(Time&);`
"immutable reference" `void fun(const Time&);`

For-loop Syntax

`for (i; i < 10; i++)` is NOT okay

XCode On OSX Yosemite

go to a Terminal session, and type the command: `g++`
If not installed, you'll see a prompt saying so
and inviting you to download and install it
accept the invitation, and in a few minutes...
...you'll have g++ (fully C++11 compatible, too).

Three program design consideration for COMSC 200

1. Pausing a the end of a program *(optional)*:

```
  cout << "Press ENTER to continue..." << endl;
  cin.get();
} // main -- return 0; not required
```

2. Programmer identification in *all* CPP and H files

```
// Lab LAB NUMBER HERE, LAB TITLE HERE
// Programmer: YOUR NAME HERE
// Editor(s) used: XP Notepad
// Compiler(s) used: VC++ 2010 Express
```

3. Programmer identification in program output

```
int main()
{
  // identifying output statements
  cout << "Lab 1, Problem Solving With C++, Part b\n";
  cout << "Programmer: Joe Student\n";
  cout << "Editor(s) used: XP Notepad\n";
  cout << "Compiler(s) used: VC++ 2010 Express\n";
  cout << "File: " << __FILE__ << endl;
  cout << "Complied: " << __DATE__ << " at " <<  __TIME__ << endl << endl;
  ...
}
```