COMSC-200 Lecture Topic 1
C++ Basics

□ Reference
Deitel Ch.1-8
Burns Ch.4, 5, and 10

□ History of C++
C, 1972, Bell Labs (from original B language)
ANSI standard C
C++, early '80s, Bell Labs, adds OOP
C++ Standard Library
C++11, the new standard
ANSI standards vs. non-standard C++ *(Microsoft)*
Java, VB, and C#

□ Promotion In Mixed Arithmetic
`int->float->double`, `bool,char->int`
  example: `'A' + 1` is 66, not 'B'
demotion: truncating a floating point
  into an integer in assignment
  `int area = 3.14 * rad * rad;`

□ Storing a DOUBLE in an INT
avoid: `int x = fabs(...);`
prefer: `int x = (int)fabs(...);`

□ Working With Absolute Values
use the `cstdlib` function
  `abs(x)` for int x
use the `cmath` function
  `fabs(x)` for float x or double x

□ C++ Header Files
function prototypes
.h files
Standard Library header files *(p.169-170)*
  use **cmath** instead of **math.h**
angle brackets and quotes in `#include`
  `#include <`*standard library filename*`>`
  `#include "`*user-written header filename*`"`
  or `class` *`className`*`;` "forward declaration"
the `std` namespace
  `using std::cout;`

□ Common Library Includes
from cctype: toupper, tolower
from cmath: sqrt, pow, exp, fabs
from cstdlib: abs, atoi, atof, rand
from iostream: cin, cout, endl, ios
C-strings vs C++-strings
  from cstring: strlen, strcpy, strcmp
  from string: string, getline

□ Testing Whole Number Equality
avoid `=` vs `==` errors:
  `if (0 == x) ...`

□ `const` Pointers
pointers can access TWO values:
  their own value: a memory address
  the value stored at that address
so `const` needs to have TWO meanings:
  protect the memory address value
  protect the value at that address
so here's how:
  `int* const p` protects the memory address value
    "constant mutating"
  `const int* p` protects the value at that address
    "variable read-only"
  `const int* const p` protects *both*
    "constant read-only"
refer to these as "leading" and "trailing" consts
`void*` is a generic pointer

□ C++ Reference Variables
`int& x = a;` means "x" is an alias for "a"
`x = b;` means set whatever "x" references to b's value
  ...does NOT mean that "x" now references "b"
use in function parameters instead of pointers
use for passing `struct` instances
  and objects
require initialization upon declaration
  ...cannot be `NULL`
cannot have arrays of reference variables
  ...or STL containers
references as return variables *(be careful!)*

□ Default Parameters
specify in function prototype *xor* definition
  preference: in prototype
`void ShowWindow(bool=true);`

□ Compiler Variations
some compilers let you get away with coding mistakes, so...
ALWAYS use parentheses with function calls
ALWAYS end a value-returning function with a `return`
ALWAYS specify the return type for functions (exceptions...)

□ 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

□ Problem-Solving Tools
parsing text files
converting text to numbers

debugging techniques

"syntax errors" prevent compilation
"logic errors" compile, but don't run right
compiling techniques
console and file I/O PDF

Three important debugging techniques

1. Simple tracing: see the progress of the program (example)

```
cout << __FILE__ << " at line " << __LINE__ << endl;
```

2. Debug statements: enable via compile command (example)

```
#ifdef DEBUG
  cout << "The value of 'a' in function fun() is [" << a << "]\n";
#endif
```

To work in command-line mode, include the sequence `DEBUG` in the compile command. E.g.:

```
cl /DDEBUG HelloWorld.cpp -EHs
```

```
g++ HelloWorld.cpp -DDEBUG
```

In the Visual C++ IDEs you have to change Project Settings or Properties:

- VC++ 2005 and up: "C/C++" in tree, "Command Line" in tree, type `/DDEBUG`

3. Assertions: program terminates when unexpected values occur (example)

```
#include <cassert>
  ...
  assert(x != 0); // if x is ever zero, then I made a logic error somewhere!
```