# Week 8

Diana Gage

www-personal.umich.edu/~drgage

slides by Diana Gage and Leah Bar-On Simmons

# Midterm Student Feedback

- **Some of your suggestions:**
  - Less review, more practice problems
  - More challenging material
  - Poll class on what to go over
  - "*Be up-to-date with topics…*"
  - More collaboration… the problems are going to be tough, so you'll **have to** work together!

# Classes: Review

- A class is a container that can hold **different** types of objects (objects with different data types)

- Another type of container we've learned, where all objects must be the **same data type**: array

# Classes: Review

- A class is a container that can hold **different** types of objects (objects with different data types)
- A class is a user-defined data type
  - Just like int and double and string are data types, so is the class you define
- A class is a way to group together related information

# Header files vs. source files

- Header files have the .h extension
- Source files have the .cpp extension

- Header files are where class definitions and function declarations go
- Source files are where function implementations go → remember :: (*source resolution operator*)
  - Ex: with Person class, in .cpp file with int getter –

**int Person::get_num()**

# Public vs. Private

- A private member variable or member function means that only members of the class can access it
- Member variables are usually private, and we use setters and getters to access them
- A public member variable or function means that any part of the code can access that function or variable. Setters and getters are always public.

# Passing Classes into Functions

- You almost always want to pass classes **by reference** into functions

- Why?
  - *Think of arrays: arrays are **automatically** passed by reference because they are large containers (take up a lot of space in memory)*
  - *Classes are also containers that usually hold a lot of information → **who wants to use pass by value and copy all of that??***

# const with functions

- Const after the statement means the member function is not allowed to modify the class's member variables (read-only)
- i.e.
  - string Card::printCard() const{}
- Important to understand which functions should be const, and which shouldn't
- **You cannot call non-const functions from const functions**

# const parameters

- ie bool goodCard(const Card& card1,
  const Card& card2){}
- The const means the cards passed in as parameters cannot be modified by the functions

# Classes: Practice Question

- Which of the following statements is FALSE?
  a. The entities which follow line 7 cannot be used outside of the class
  b. The private declaration is in effect until it is changed
  c. Only the entity on line 8 is private; line 9 is not private
  d. There is no syntax error on line 4
  e. You can access x and text inside member functions of MyClass

```
1     class MyClass
2     {
3     public:
4         MyClass( int, string );
5         void display();
6
7      private:
8         int x;
9         string text;
10     };
```

# Classes: Practice Question

- Which of the following statements is FALSE?
  a. The entities which follow line 7 cannot be used outside of the class
  b. The private declaration is in effect until it is changed
  c. **Only the entity on line 8 is private; line 9 is not private**
  d. There is no syntax error on line 4
  e. You can access x and text inside member functions of MyClass

```
1     class MyClass
2     {
3     public:
4         MyClass( int, string );
5         void display();
6
7      private:
8         int x;
9         string text;
10     };
```

# Today: write our own class

- Our own version of Stay in the Blue App ☺
  - Using the charts on: http://www.brad21.org/bac_charts.html
    ****slightly modified….***

- We will have:
  - Header Files
  - .cpp files with implementations
  - Main.cpp to use our classes

- **TOPICS COVERED IN EXERCISE:**
  - Primarily Classes
  - Also: 2D arrays, 1D arrays

# In the Header files (.h)

- Some member functions
- Public and private members
- Default constructors
- Non-default constructors
- Getters
- Setters

# .cpp files

- Have all the implementations of the declarations in the header files
- Be careful with syntax here!!!

# In main.cpp we want to…

- Make 2 instances of the Person class, 3 instances of the Drink class
- Use a default constructor at least once to practice (not in Assignment 4)
- Access elements and change them, using getters and setters

# Final Product

- Our final product will be uploaded to discussion resources along with these slides
- Think about ways you could improve our code!

# Plan of Attack:

- Implement functions in Drink.cpp first
- Then functions in Person.cpp
  - Some of these require Drink objects, so Drink.cpp needs to be completed!
- Then main!