# CSCI 315: Data Structures
# C++ Testing, Pointers, and OOP

Paul E. West, PhD

Department of Computer Science
Charleston Southern University

January 15, 2019

- Objects in C++ share many similarities with Java, but are considered more powerful:
  - They can be created on the stack.
  - C++ allows multiple inheritance.
    - no need for the interface keyword.
  - Templates act like generics, except they are a compile time construct, not run time.
- Remember, C++ is compiled to machine code, Java compiles to byte code!

# Testing

***Testing will make you more productive.***

This means you will spend less time working on problems because you will increase you chances of getting it right the first time AND virtually never make the same mistake twice.

# Warning

- Proven Fact: A computer program cannot sufficient generate test cases for you code.
- When I say proven, I mean Turing wrote a proof that shows beyond the shadow of a doubt that it is impossible for a computer to determine if a program halts.
- If we don't know if it halts how can we test?
- This is your job, and an advantage you have over the machines.

# Unit Test

- For this class Unit testing is sufficient.
- There are many other forms of testing, but this class focuses on (relatively) small units of code.
- There are numerous Unit Testing tools, but no de facto for C++
  - Unlike Java's JUnit
- I have chosen CXXTestGen for this class.
  - It is the simplest which still providing sufficient robustness.

# CXX Test Gen

- Make sure you have it installed:
  - # apt install cxxtestgen
- Simple test case:

```
#include <cxxtest/TestSuite.h>
#include <limits.h>
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

class MyTestSuite1 : public CxxTest::TestSuite {
    public:
        void test1(void) {
            /* Fill in some test cases here for cxx test gen */
            TS_ASSERT(add(1, 2) == 3);
        }
};
```

# Running The Test

1. First we generate the test code using cxxtestgen:
   $ cxxtestgen –runner=ErrorPrinter -o test-runner.cpp simple-test.cpp
2. Then we compile the code:
   $ g++ test-runner.cpp -o test-runner
3. Then we run!
   $ ./test-runner

- Keep in mind that we will want to test code in other files.
- So, we need to include that file in #2, for example if we had a sum.cpp, then #2 would be:
  $ g++ sum.cpp test-runner.cpp -o test-runner
- You make add optimization and profiling if desired.

# Overview

- As I said, in C++ a pointer can point to anything (well anything that is pointable!)

# Stack Pointer

- int *a;
  - declare a variable 'a' that stores the memory address of where an integer resides.
- int b = 32;
  a = &b;
- the & symbol calculates the memory address of the following variable.
- So the above declares 'b' on the stack, sets 'b' to 32, then stores the memory address of 'b' in 'a'.
- *a = 64;
- In this instance the *, mean go to where 'a' is pointing and store 32 there.
- Since 'a' is pointing to 'b', b now has 64.

# Heap Pointer

- int *a = new int;
- This reads, create a variable 'a' on the stack that will store the memory address of an integer. Next allocate an int in memory and store the memory address of that int in 'a'.
- You can manipulate the new int just like before.
- Although, since the integer is on the heap, you must deallocate!
  - delete a;

# Arrays/Pointer Arithmetic

- int ary[100];
  int *a = ary;
- first line creates an array of 100 integers on the stack.
- second line creates a pointer to the array. Notice we didn't need the &!
- *(a+10) = 50; Is the same as:
  ary[10] = 50!

# Casting

- C++ is incredibly powerful, because of casting.

```
int b = 300;
int *a = &b;
double *d = (double*)a;
*d = 3.14159;
```

- Yes, that is legal.
- What is the value of b?

# Casting

- C++ is incredibly powerful, because of casting.

```
int b = 300;
int *a = &b;
double *d = (double*)a;
*d = 3.14159;
```

- Yes, that is legal.
- What is the value of b?
- Undefined!

# There is more

- Pointers can point to other things:
  - functions
  - other pointers
  - Arrays
  - objects

  Lastly, void * means I'm creating a pointer to something.

# Class Organization

- Enforced through access keywords
  - public: for interface
  - private: to make implementation inaccessible
  - protected: access for subclasses only
- In Java
  - each member is prefixed with a keyword
  - another access level: package-access
- In C++
  - public, private, and protected sections
  - friend keyword used to break encapsulation (don't use!)

# Inheritance

- Feature that allows a class to be defined based on another class
  - methods and attributes are inherited
- Java and C++ difference
  - Java: public class A extends B
  - C++: class A: public B ;
    (different types of inheritance)
- Multiple inheritance possible in C++, not in Java
- But in Java, one may implement several interfaces

# Parametric Polymorphism

- We will get to later...

# Pointers

- In Java a pointer was called a reference to an object:
  - String str = new String("Hello!");
- In C++ a pointer stores the memory location of something else
- For objects:
  - std::string *str = new std::string("Hello!);

# A better explanation

- In Java a pointer only existed to objects.
- In C++ a pointer may point to anything.

# A better explanation

- In Java a pointer only existed to objects.
- In C++ a pointer may point to anything.
  - Simple, right? *que maniacal laughter*

# Constructor

- Constructor
  - place where you include code that initializes the object
- Default Constructor
  - no additional info required
- User-defined Constructor
  - with parameters that specify values or sizes
- Java and C++ behave the same way with constructors

# Arrays

- In Java arrays are actually objects that store length and other information.
- In C++, an array is a sequence list of data. (Nothing else is stored.)
- int x[20]; Button b[20];
  - Valid declarations in C++, not in Java (why?)
  - Creates 20 ints and 20 Button objects

# Pointers and Arrays

- In C++, there is a close relationship between pointers and arrays
- Instead of int x[20]; can issue
  int *x; x = new int[20];
  to allow for dynamic allocation
  - Usage of the array (e.g., x[3] = 5;) identical in both cases
  - To deallocate, use delete [] x;

# Constructors in Java and C++

- In Java,
  - a constructor is invoked only through the new keyword
  - recall that all object variables are references
- In C++,
  - a constructor is called upon variable declaration, or explicitly through new with pointers, or in other situations
  - other types of constructors

# C++ Destructor

- Special method whose signature is a  followed by the name of the class
- e.g.,  SomeClass();
- Particularly if the class contains pointers and the constructor contains calls to new, a destructor needs to be defined
- e.g., SomeClass()  A = new int[20];
  SomeClass()  delete [] A;

# C++ Control Over Copy and Assignment

- In C++, the semantics of "a = b" (assignment) can be specified
  - by defining the copy-assignment operator
- In C++, there is a copy constructor
  - specifies what happens during object copying, e.g., when function parameters are passed
- There is more low-level control
  - shallow copy vs deep copy

# Methods

- Defines object behavior
- Static methods vs instance methods
- Method overloading
  - within class, two methods with the same name but different signatures
- Method overriding
  - same signatures across different classes (subclass and superclass)

# Operator Overloading

- In C++, operators like =, +, *, ==, etc. can be defined, just like methods
- Example:

```
class Matrix {
    // ...
    Matrix operator+(Matrix m) { }//
};
c = a + b;   // equiv to c = a.operator+(b);
```

# Method Binding

- Let Teacher be a subclass of Employee
  - Also, suppose promote() is a method defined in both classes
- Employee variables can refer to Teachers
  - In Java, Employee e; e = new Teacher();
  - In C++, Employee *e; e = new Teacher;
- e.promote() (or (*e).promote() ) calls which promote() method?

# Static vs Dynamic Binding

- In C++, Employee's promote() is called
  - Determined at compile time and deduced from the type of the variable (static binding)
- In Java, Teacher's promote is called
  - Determined at run-time because the actual type of the referred object is checked then (dynamic binding)

# There is more

- While there are many more features I want to pause here before continuing

# homework

- There is a lab **and** project assigned today
- I recommend completing lab05 before starting the project.