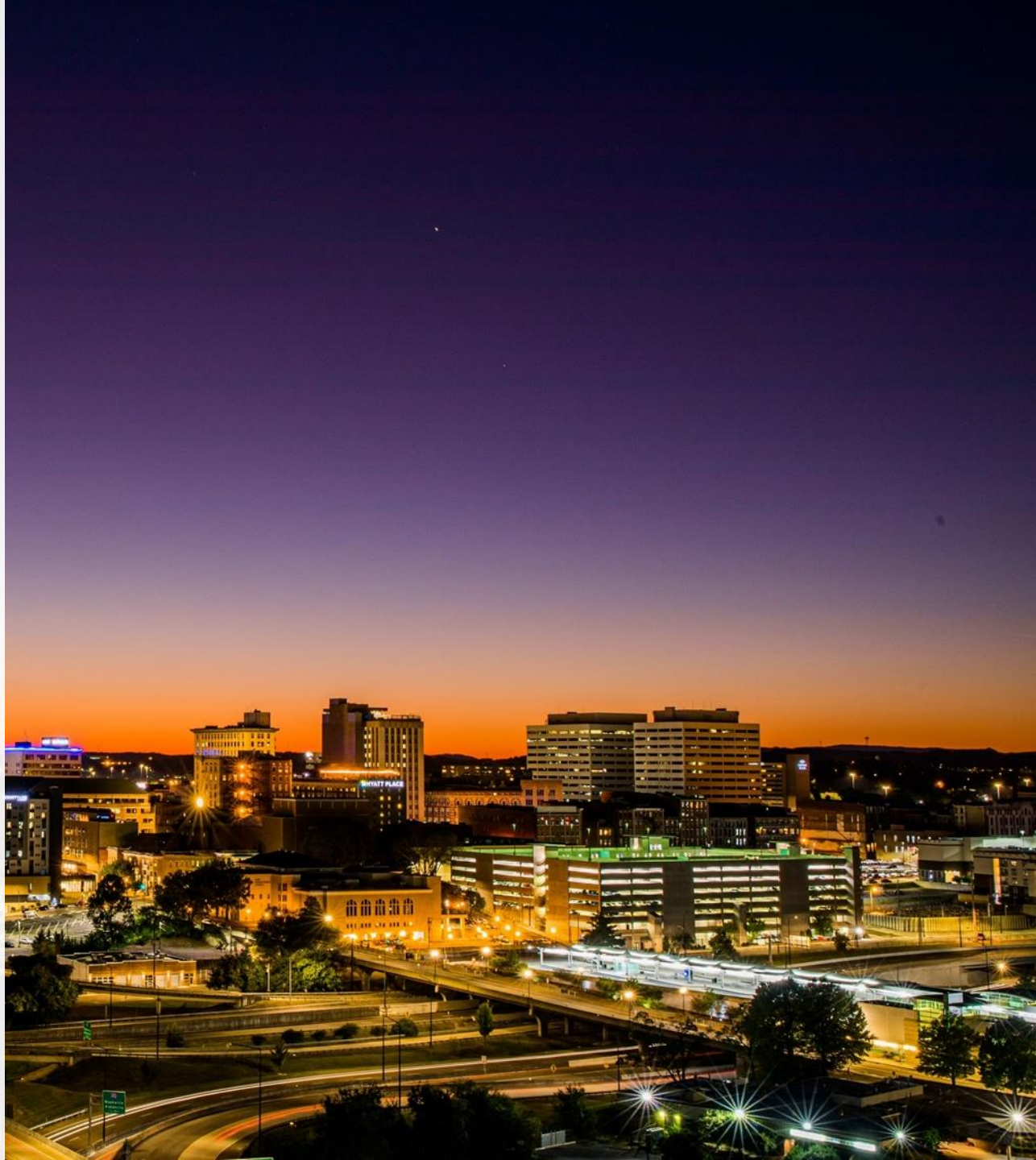
NE697: Introduction to Geant4

C++ Classes, Const

September 14th, 2021

Dr. Micah Folsom

THE UNIVERSITY OF TENNESSEE KNOXVILLE



Today's Agenda

- Administrative items?
  - Videos for auditors will be uploaded this week

- Assignments graded, need to post and send out feedback

Last Time, On NE697…

- Value vs pointer
- Stack vs heap
- Function input types

- class vs struct
- Access levels, encapsulation

C++ Function Input Types

- Read-only input?
  - Pass-by-constant-reference
- Read/write input?
  - Pass-by-reference
- Expect input to be a pointer?
  - Pass-by-pointer
- Pass-by-value: it's not going to kill you, but…
  - Makes a copy (no persistence), not guaranteed to be constant
  - Assign to a local variable, it's still just 1 copy
- Const pointer and/or pointer to a const value
  - Usually, the goal is to modify the object
  - Making the pointer itself constant ends up being really annoying!

```cpp
// Pass-by-value
void by_value(int x) {
  // x is a copy, so while its value changes locally, the input variable is
  // not modified
  x += 1000;
  return;
}

// Pass-by-reference
void by_ref(int& x) {
  // We got the address of the data, so the input variable is modified (not
  // copied) - lasting consequences
  x += 1000;
  return;
}

// Pass-by-const-reference
void by_const_ref(int const& x) {
  // We can't even compile this - the const guarantees we can't change the
  // value, and we avoid the copy
  //x += 1000;
  return;
}

// Pass-by-pointer
void by_pointer(int* x) {
  // We got the address of the data, so the input variable is modified (not
  // copied) - lasting consequences
  (*x) += 1000;
  return;
}
```

C++ Const Correctness

```
2  // There are multiple ways to write constant values
3  // One is more like English, which is tempting...
4  const int SIZE1 = 10;
5  // But this is more correct!
6  int const SIZE2 = 100;
7
8  // The reason is because of how const evaluates: it applies to its left,
9  // and if there's nothing there, then to its right
10 // Because of this, if you put const in "the correct place" (where you put
11 // things to its left, first, the way its evaluated), then if you read
12 // the type from right to left, you will always have the correct
13 // interpretation
14 /*
15  int const -> "a constant int value"
16
17  int const* -> "a pointer to a constant int value"
18
19  BAD: const int* -> "???" behaves the same as above!
20
21  int* const -> "a constant pointer to an int value"
22
23  int const* const -> "a constant pointer to a constant int value"
24 */
```

- const applies left, then right
  - Think/read right to left
- Read it out loud
  - SIZE1: “int value constant”
  - SIZE2: “constant int value”
- Applies to variables, function arguments

C++ Const Correctness

- Member functions of classes can also be const
  - Guarantees that the attached object does not change
  - “const” comes after the function signature
    - int const& get_id() const;
- Perfect for our *getter* functions
- What type of input argument should the *setter* take?

- [DEMO]
  - Update the Point class accessors
  - Constructors and destructors

C++: Project Upgrade

- We now have a fully defined class!
  - But it's all defined above main() ☹
- Remember: declarations in .hpp, implementation in .cpp

- [DEMO]
  - Moving Point to its own source files
  - Organizing project code files: src/ and include/
  - Updating CMakeLists.txt for the new structure