

CSCI251/CSCI851      Autumn-2020  
Advanced Programming      **(LT9)**

Lecture Tutorial 9

# From the Lab:

- When you have multiple inheritance with the same ancestor twice in the hierarchy of a class, you will run into the diamond problem.
  - `HoverCraft` is derived from `SeaTransport` and `LandTransport`, both of which are derived from `Transportation`.
- To deal with this, use Virtual inheritance.
  - When using **virtual inheritance**, the **virtual base class's constructor** is called directly by the most derived class's **constructor**.
    - So, the most derived class needs to directly call the base class.

- Overloading Student ++ ...
- Why return \*this?
  - See [https://en.cppreference.com/w/cpp/language/operator\\_incredec](https://en.cppreference.com/w/cpp/language/operator_incredec).
  - For built-in prefix return references, postfix return values.
    - You could actually return whatever you like but we want consistency in the meaning.
- Postfix vs Prefix: ++X vs X++
  - Dummy variable on the operator for postfix...

```
Student& Student::operator++( )  
{  
    ++year;  
    return *this;  
}
```

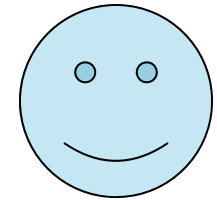
```
Student& operator++( ) ;  
Student& operator++(int) ;
```

```
Student& Student::operator++(int dummy)  
{  
    year++;  
    return *this;  
}
```



```
Student& Student::operator++()  
{  
    ++year;  
    return *this;  
}
```

```
Student& operator++();  
Student operator++(int);
```



```
Student Student::operator++(int dummy)  
{  
    Student temp(stuID,year,WAM);  
    ++(*this);  
    return temp;  
}
```

The dummy variables has a default of 0. Change → `a.operator++(2)`

# The Three Little Software Engineers

- Really the three little pigs...
- Why the timeline?
  - Remember the idea of `main( )` telling a story?
  - Well here it literally does and you might see a sequence of calls in `main( )` that reflects the timeline you generate.

- Probably one of the more significant problems I saw was people having 3 pig classes, or 3 house classes.
- The 4 pigs aren't the same, but they have the same attributes just with different values.
  - While they don't do exactly the same things they could if they were smart enough to do so, or were following instructions.
- Similarly with the houses, they differ in materials and resulting strength.
  - Those can be represented by different attribute values.
  - The success of the attempt of the wolf to blow the house in would be determined by comparing the house strength and the “wolf puff strength”.