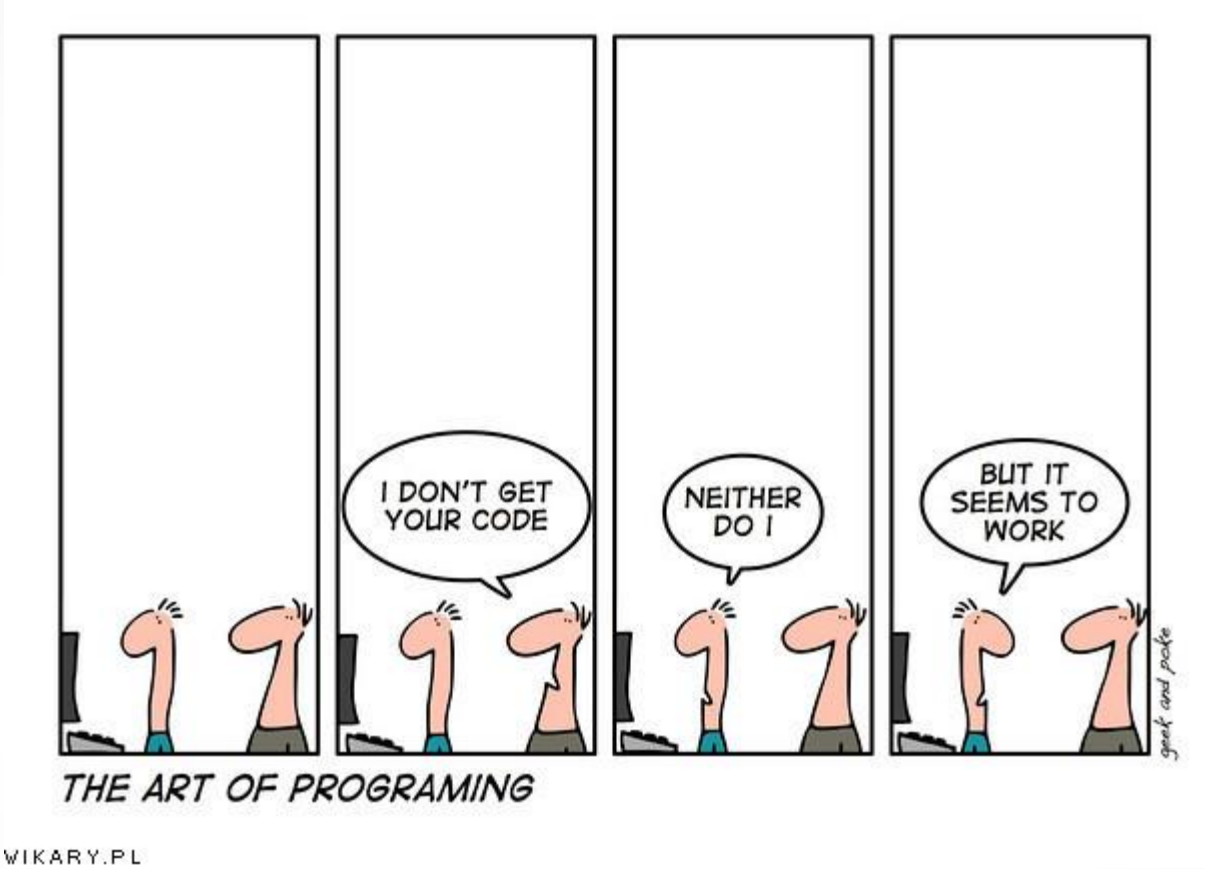
C++

SWE315

Lesson 4

Prof. Zachi Baharav
zbaharav@cogswell.edu


I DON'T GET YOUR CODE
NEITHER DO I
BUT IT SEEMS TO WORK
geek and poke
THE ART OF PROGRAMING
WIKARY.PL

Lesson 4

- In this lesson:
  - Class :Time example!
    - Namespace
    - Scope
    - Functions: Pass by value/reference
      - const
    - Operator overloading: >>, <<
      - friends
  - Next time:
    - Arrays, pointers (we'll skip mostly).
    - 'this'
  - Programs:
    - Time
  - Lab work (time permitting)

Namespace

- Declarative region (block)
- Potential scope
- Scope

- Using declaration

  using std::cout ;

- Using directive

  using namespace std;

const Member functions

- Consider the code:

```
const Time midnight(24,0);
midnight.show();
```

- The compiler will complain: const cannot be modified.
- In the past, solved by const parameters, but here there are no parameters.
- Solution:

```
void show() const;
```

And

```
void Time::show() const {  }
```

Reference variables

- Main use is as a formal argument to a function.
- If you use reference as an argument, the function works with the original data instead of a copy.
- Great for structures and class-objects!

- Passing by reference:

```
void swapr(int & a, int & b);
// In declaration and implementation
```

- And in the calling program:

```
int a,b;
swapr(a,b);
```

- You are just letting the compiler/linker know to pass by reference! Other than that, the code is the same.

Reference

- Most common mistake:
  - **Returning reference to a memory location that ceases to exist when the function terminates.**

```
Time & NewZelandWatch(Time & t)
{
        Time newTime(0,0);        // always beginning of time there
        return newTime;
}
```

- Typically, the return reference refers to a reference passed into the function, so the calling function actually has it to start with!

Overloading

- Define a function

operator‘op’(argument list)

- Very useful in Class definition, where one (invoking) argument is implicit.
  - Two equivalent ways of calling:

total = coding + fixing;

total = coding.operator+(fixing)

- Is this valid?

t4 = t1+ t2 + t3 ;

t4 = t1.operator+(t2+t3);

friends

- Remember: Overloaded operators can be done using either member or non-member functions.
- Non-member function is not invoked by an object!
  - Instead, all are explicit arguments.
  - So:
    - t2 = 2.75*t1
  - Can be translated by the compiler to
    - t2 = operator*(2.75, t1);
- With a non member overloaded operator, the first argument is the left item, and the second is the right.

END