THE UNIVERSITY OF VERMONT
COLLEGE OF ENGINEERING &
MATHEMATICAL SCIENCES

Reference Variables

and passing by reference

CS 124 / Department of Computer Science

What is a reference variable?

A reference variable is just another name we give a variable.

```
int foo = 255;

int& anotherName = foo
```

We use the & to indicate we are creating a reference variable. Notice that this is different from using the same symbol to retrieve an address.

```
int* x = &foo;
```

Reference variables are just a reference

```
int foo = 255;
int& refFoo = foo;
int* fooPtr = &foo;
int* refFooPtr = &refFoo;

std::cout << foo << std::endl;  // prints 255
std::cout << refFoo << std::endl;  // prints 255
std::cout << fooPtr << std::endl;  // prints an address
std::cout << refFooPtr << std::endl;  // prints same address
```

```
255
255
0x7ffeeb898788
0x7ffeeb898788
```

Reference variables aren't copies

```
int foo = 255;
int& refFoo = foo;
refFoo = 42;

std::cout << foo << std::endl;  // prints 42
```

We assigned a value of 42 to `refFoo` but `foo` is changed too. `foo` and `refFoo` are one and the same thing, with different names.

What are reference variables good for?

Why go to the trouble?

What are reference variables good for?

Why go to the trouble?

PASS 👏

What are reference variables good for?

Why go to the trouble?

PASS 👏 BY 👏

What are reference variables good for?

Why go to the trouble?

PASS 👏 BY 👏 REFERENCE 👏

Pass by reference

Make your function argument(s) reference(s) and then your function can *change the value of the variable outside the function.*

Pass by reference

```
void swap(int& x, int& y) {  // takes inputs as reference
    int temp = x;
    x = y;
    y = temp;
}


int a = 42;
int b = 77;
swap(a, b);
std::cout << a << " " << b << std::endl;
```

Prints...

```
77 42  // a and b have been swapped
```

Pass by reference

```
void collatz(int& x) {  // takes input as reference
    if (x % 2) {
        x = 3 * x + 1;
    } else {
        x = x / 2;
    }
}


int foo = 255;
collatz(foo);
std::cout << foo << std::endl;
```

Prints...

```
766  // foo has changed!
```

Review of usages in "Lecturer" class

When we coded our lecturer class, there were a number of places where we passed by reference. We didn't discuss what was going on in detail at the time. Now we'll revisit these points in the code.

Review of usages in "Lecturer" class

```
friend std::ostream& operator << (std::ostream& outs, const Lecturer& lec) {
    outs << std::setw(30) << lec.getName()
         << std::setw(5) << lec.getCourse1()
         << std::setw(5) << lec.getCourse2()
         << std::setw(8) << lec.getCourse3()
         << std::setw(20) << lec.getOffice();
    return outs;
}
```

Review of usages in "Lecturer" class

```
friend std::ostream& operator << (std::ostream& outs, const Lecturer& lec) {
    outs << std::setw(30) << lec.getName()
         << std::setw(5) << lec.getCourse1()
         << std::setw(5) << lec.getCourse2()
         << std::setw(8) << lec.getCourse3()
         << std::setw(20) << lec.getOffice();
    return outs;
}
```

Review of usages in "Lecturer" class

```
friend bool operator<(const Lecturer& lhs, const Lecturer& rhs) {
    return lhs.getName().length() < rhs.getName().length();
}
```

Review of usages in "Lecturer" class

```
friend bool operator<(const Lecturer& lhs, const Lecturer& rhs) {
    return lhs.getName().length() < rhs.getName().length();
}
```

Review of usages in "Lecturer" class

```
void readLecturersFromFile(std::string filename,
    std::vector<Lecturer>& lecturers) {

        ...

        Lecturer lec(name, office, course1, course2, course3);
        lecturers.push_back(lec);

        ...


}
```

Review of usages in "Lecturer" class

```
void readLecturersFromFile(std::string filename,
    std::vector<Lecturer>& lecturers) {

        ...

        Lecturer lec(name, office, course1, course2, course3);
        lecturers.push_back(lec);

        ...


}
```