Today

- More on functions (textbook chapter 5).
  - The textbook chooses not to cover reference parameters until Chapter 6 (where it covers pointers).
- Things to notice from the example programs I gave you.
- The conditional operator

More on functions: reference parameters

- Last class we talked about how functions can pass parameters
  - How the value of those parameters might change inside the function
  - But in the *calling function*, the value of the parameters does not change
- We also talked about *scope*
  - How variables are defined within either a *global* or a *local* scope
  - How *local* variables, e.g., those that are defined within a function, "go away" when the function exits

- In C++, there is a feature of functions called *reference parameters*.
- This lets you pass what is called the "address" of a variable to a function.
- This means that it is the variable itself rather than a copy that gets passed to the function.
- As a result, when the function exits, if the value of the variable has changed inside the function, then the new value can be retained outside the function.

Reference parameters: classic example

- The classic example of using reference parameters is a function called `swap()`

```
void swap( int &a, int &b )
{
  int tmp;
  tmp = a;
  a = b;
  b = tmp;
  return;
}
```

Other things to notice

- The fox and rabbit example shows you that:
  - You can have a function with many parameters that don't return a value
    `displayPosition`
  - You can have a function with no parameters that returns a value
    `makeRandomMove`
  - You can have a function with parameters that return values
    `wrapAround`
  - You can have a function with many returns
    `wrapAround`
  - And we already knew that we could have functions that take no parameters and return no values.

The conditional operator

- C++ contains a compact version of `if else`, which can sometimes be useful.
- `<condition> ? <if true> : <if false>`
- If the condition is true, the bit between the `?` and the `:` gets executed.
- If the condition is false, the bit between the `:` and the `;` gets executed.

Random number generation

- The fox and rabbit example uses random numbers.
- `rand()`
  generates a random number in the range 0 to `RANDMAX`.
- `rand() % m`
  generates a random number between 0 and `m - 1`.
- `n + rand() % m`
  generates a random number between `n` and `n + m - 1`.
- What does
  `1 + rand() % 6`
  do?

- The random numbers generated by `rand` depends on the *seed*.
- The seed is set by
  `srand`
- A typical way to do this is:
  `srand(time(NULL))`
- This will generate a new seed every time the program is run (more or less).

- Thus

```
if (a >= b)
{
  y = a;
}
else
{
  y = b;
}
```

can be written as:

```
y = a >= b ?  a :  b;
```


Summary

- The main point of this lecture was to introduce reference parameters.
- The rest of the lecture pointed out some things we had already covered, but maybe hadn't spent as much time on as we should.
- Oh, and we finished covering all the material up to the end of Chapter 5.