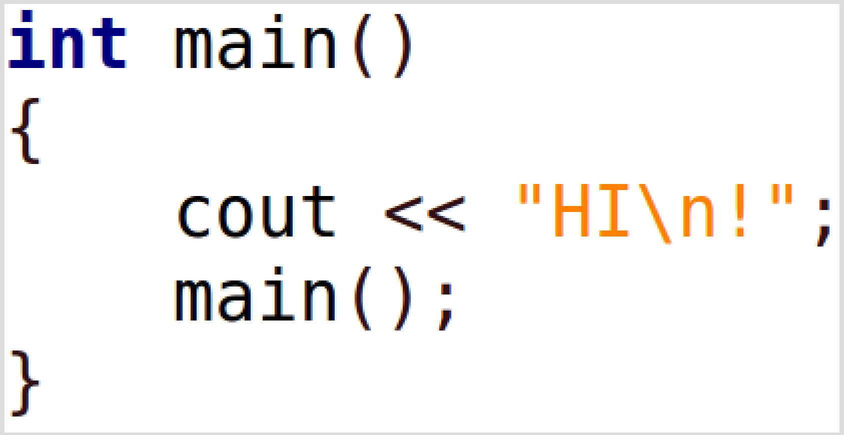
Recursion

Ch 14

Highlights

- recursion



```
int main()
{
    cout << "HI\n!";
    main();
}
```

Recursion

No fancy blue words or classes this chapter

Recursion is simply calling a method from inside itself

This copy will re-run the method on any new arguments or information

(See: badRecursion.cpp)

Click to exit presentation...

Recursion

If you forget your stopping case, you will not get an infinite loop but crash the program

This is because every function call takes up more memory, so you constantly ask for more memory

Eventually the memory (stack) cannot store anymore

Recursion basics

Good recursion must have 2 parts:

- A recursive call on a **smaller** problem
- An ending case

(see: https://www.youtube.com/watch?v=-xMYvVr9fd4)

In order to use recursion, you must be able to identify a subproblem that is very similar to the original problem

Each step must get you closer to the solution

Recursion basics

For recursion, you can basically **assume** your function works as you want it to (even though you have not written it)

If you have the ending case and reduction step correct, then it will!

Recursion: Family tree

Person

↓

Descendant

Recursion: In words

A child couldn't sleep, so her mother told
a story about a little frog,
    who couldn't sleep, so the frog's mother told
    a story about a little bear,
        who couldn't sleep, so bear's mother told
        a story about a little weasel
            ...who fell asleep.
        ...and the little bear fell asleep;
    ...and the little frog fell asleep;
...and the child fell asleep. (See: story.cpp)

Recursion: Basic example

Remember, code starts in main and runs from top to bottom in sequence (normally)

When you call a function you go execute all the function's code is run before going back to the original code

Code order is important in recursion!

(See: stringRecursion.cpp)

Recursion

What if I wanted to just count down to zero?
countdown(5) would show:

```
5
4
3
2
1
0!
```

(see: countdown.cpp)

Recursion: Fibonacci

The Fibonacci numbers are defined as:

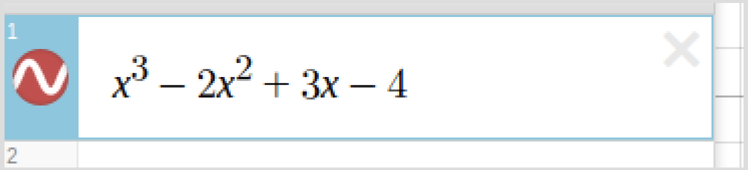
F(n) = F(n-1) + F(n-2)

In other words, you add the previous two to get the next

This is recursion! Computing F(n) involves solving smaller problems of F(n-1)
(See: fibonacciRecursion.cpp)

Recursion: Root finding

Find a root of:
(see: rootFind.cpp)

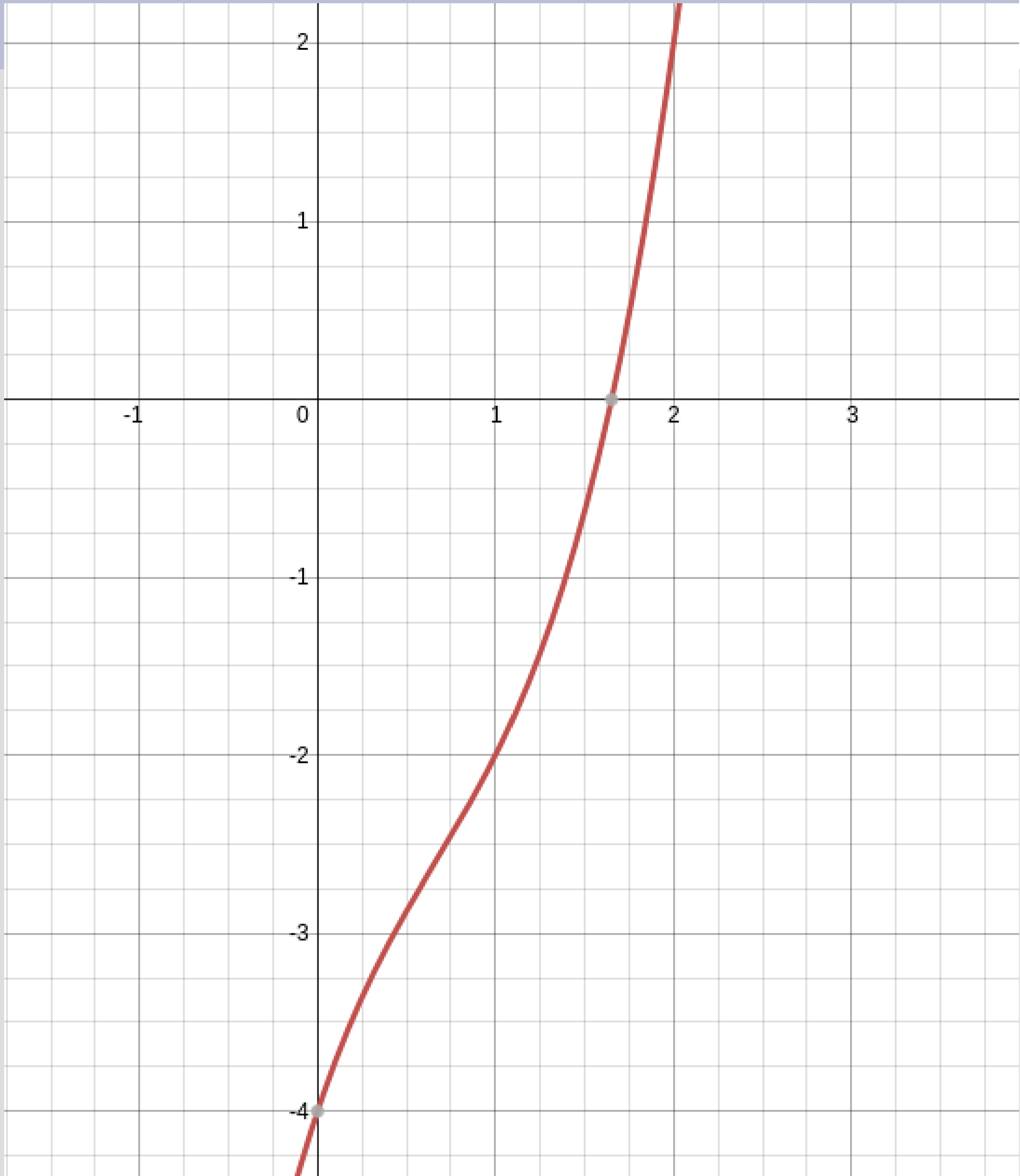


1

$x^3 - 2x^2 + 3x - 4$

2

Method:
1. Find one positive y and 1 neg. y
2. Find midpoint (of x values)
3. update y-pos/neg



Miscellaneous notes

Try googling “recursion” and click on the spelling suggestion

Recursion is very powerful and used in many advanced algorithms

It will give you a headache for a while...
=(