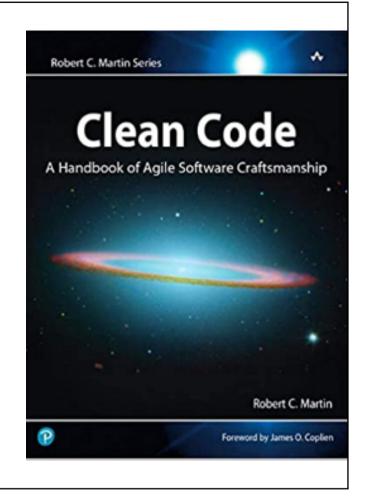
Style How to write clean code

```
struct thingy {
 int x;
   double y;
} ;
int calcualte_result(struct thingy x, struct thingy y) {
   if((x.x - y.y) > (y.x - x.y)) {
   return 0;
   } else if ((y.x - x.y) > (x.x - y.y)) {
      return 1
   } else {
     return -1;
}
int main(void) {
   struct thingy x;
   x.x = 50;
   x.y = 5.0;
  y.x = 45;
   y.x = 2.5;
  calculate_result(x, y);
```

Book suggestion

- I don't recommend many books
- This is a good one



1511 has a style guide

Follow the style guide (will be marked)

There is no *right* style guide, but you should follow it

Constants

Constants and Enumerations

Use #define or enum to give constants names.

You are only allowed to use #define's for literals (i.e. numbers, strings, ch

#defines must be written in ALL_CAPS_WITH_UNDERSCORES. enum nar lower_snake_case, and fields must be written in UPPER_SNAKE_CASE. You enum — in other words, do not use an enum to represent a specific nume

Explanation

Unexplained numbers, often called magic numbers, appearing in code ma

If a number appears multiple times in the code, bugs are created when the

A similar problem is that a number may appear in the code for multiple re like 10, and if the code needs to be changed it can be hard to determine w be changed.

Example

```
#define DAYS_OF_WEEK 7
enum days {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, WEEKEND};
// ....
int array[DAYS_OF_WEEK];
int i = 0;
while (i < DAYS_OF_WEEK) {
    a[i] = i;
    i++;
}
// ....</pre>
```

Don't Do This

// BAD - enum fields are not captialized

Let's fix this up:

```
struct thingy {
   int x;
   double y;
};
int calcualte_result(struct thingy x, struct thingy y) {
  if((x.x - y.y) > (y.x - x.y)) {
   return 0;
   } else if ((y.x - x.y) > (x.x - y.y)) {
       return 1
  } else {
      return -1;
  }
}
int main(void) {
  struct thingy x;
   x.x = 50;
   x.y = 5.0;
  y.x = 45;
  y.x = 2.5;
   calculate_result(x, y);
```

Command Line Arguments

So far...

– We can pass input into functions:

```
int cool_calculation(int
x, int y)
```

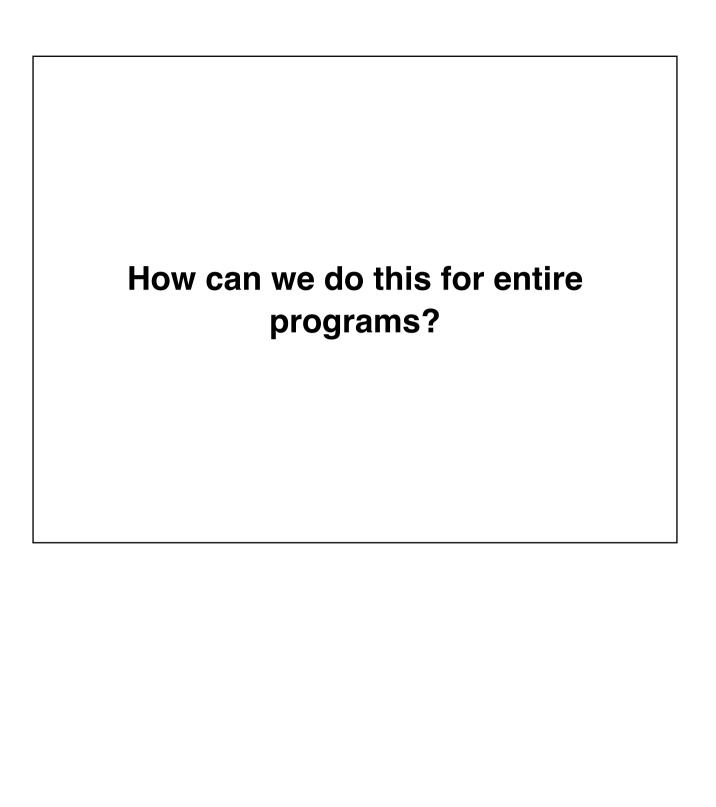
int x, int y are the input, or arguments into the function

We can use the input to determine how the function runs

```
int cool_calculation(int x, int y) {
    if (x > 0) {
        // do something when x is

positive
    } else {
        // do something if x is

negative
    }
}
```



Command Line Arguments

Command Line Arguments

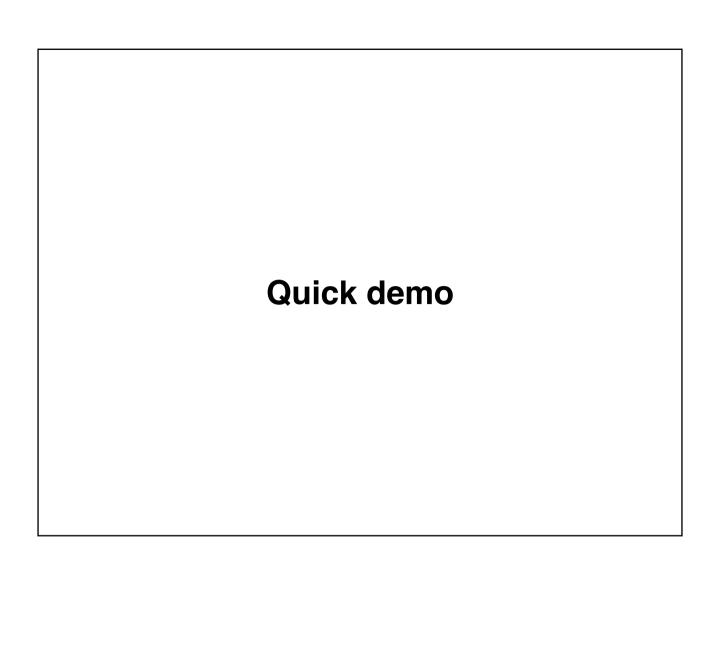
- We can provide input via user input (scanf)
- Maybe we don't want the input to come from the user, or we already have the input
- We would like to be able to pass input to a program
- We can modify main to allow for CLI

before

```
int main(void) {
}
```

after

```
int main(int argc, char *argv[]) {
    //...
}
```



String to int

- Sometimes we want to read in numbers
- But all standard input is textbased
 - 6 is really "6"

Use the atoi() function to convert strings to integers

- Stands for ASCII to IntegerIncluded in stdlib.h
- atoi(const char *str)
- atol, atof and atoll all exist(long, float, long long)

One more thing:

– Counting while loops is common :

```
int i = 0
while (i < SOME_NUM) { i++; }</pre>
```

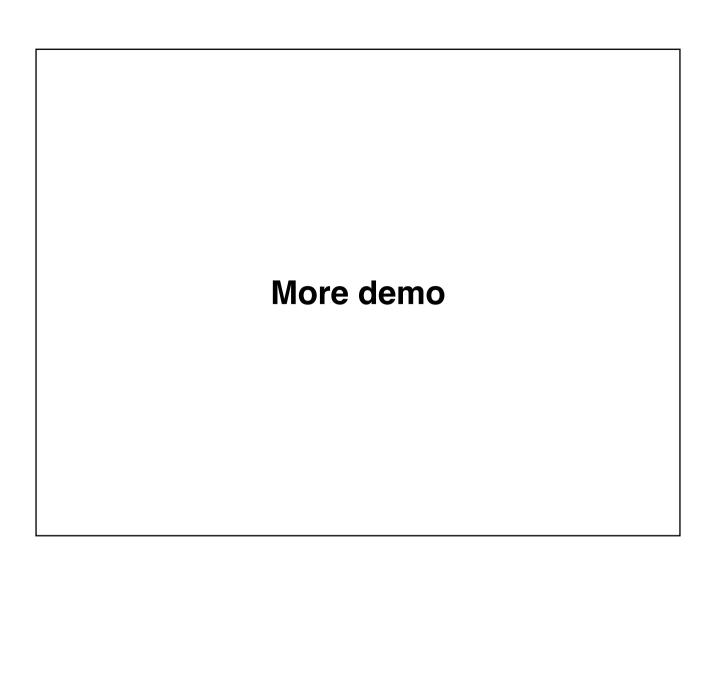
 So common, that a syntactical sugar exists that makes it a little easier

While loop

For loop

```
for (int i = 0; i < SOME_NUM; i++) {
    ...
}</pre>
```

We save a whopping 2 lines of code!



Feedback

https://forms.office.com/r/K3PjvWebtD

