Call-by-Value Demands Copies of Structure Arguments

If you pass a structure to a **C** function,
- **call-by-value semantics demand**
- that the **compiler make a copy** of the structure.

Every function call must make a new copy.

Structures can be large.

Doing so is rarely acceptable.*

*A complex number composed of two floating-point numbers is an example of a possible exception.

Let's Define a Stack Structure to Solve a Problem

Let's do an example. Let's **develop**
- a **stack structure** and
- some **operations on a stack**,
- then use the stack to solve a problem.

Our stack structure?

`struct stack_t`

The task:
- **read input line by line,**
- **then print it out in reverse.**

Compiler Must Be Able to Know a `struct`'s Size

```
struct stack_t {
    // 500 lines of up to 200 chars
    char data[500][200];
    int32_t top;
};
```

Why only 200 characters per line?

And why only 500 lines?

Fields must have known size.

Fields Can Have Pointer Types

But …

wait a minute …

a pointer has known size, too!

Later, we will learn how to allocate memory dynamically.

For now, we have to pick values, so
- at most 500 lines, and
- at most 200 characters per line.

