Use the -> Operator to Access Fields after Dereferencing

One more operator:

- **->**
- **dereference and access a field**

Rather than writing

```
(*s).top
```

,

we can write

```
s->top
```

.

The two expressions are equivalent.

Revised Function to Check Whether a stack_t is Empty

```
// Returns 1 if stack is empty, or
// 0 if stack is not empty.


int32_t stack_empty
    (const struct stack_t* s)
{
    return (500 == s->top);
}
```

Use the -> operator.

A Function to Initialize a stack_t

Notice the human naming convention: the **stack_** prefix tells programmers that the function deals with a **stack_t**.

```
void stack_init (struct stack_t* s)
{
    s->top = 500;
}
```


What Other Operations Do We Want for stack_t?

What other operations might we write for our stack?

- **Check whether a stack_t is full,**
- **push a string onto a stack_t, and**
- **pop a string from a stack_t.**

The first is easy.

For push/pop, we need to make choices.