Fields of a Structure are Just Like Other Variables

Fields act
- **like any other variable**
- **of the field's type.**

With our book example,

`book.pages` has type `int32_t`,

`book.price` has type `double`, and

`book.author` has type `char*` (the `author` field is an array of characters, so the field name has type `char*`).

Structure Types Must By Default Include "`struct`"

By default,
- the **name of a structure type** in **C**
- **must include** the keyword `struct`.

For example:

```
struct book_t a_book, another_book;
```


Structure Assignment Copies All Bits

```
struct book_t a_book, another_book;
// … some code to fill in a_book

// What does this assignment do?
another_book = a_book;
```

Copies all bits from `a_book` into `another_book`.

Pass Pointers to Structures, not Structures, as Arguments

```
struct book_t a_book, another_book;
// … some code to fill in a_book
another_book = a_book;

// Why pass a structure's address?
my_book_printer (&another_book);
```

To avoid copying the entire structure onto the stack.

