We Check for Initialization on Each API Call

We choose the last option:
initialize on the first API call.

Why?

Dynamic allocation is already "expensive"
(>200 cycles on my Cygwin desktop in April 2018).

And only one check is needed.
- User cannot call **`mem220_free`** first.
- Need to add a check to **`mem220_allocate`**.
- Other API calls call **`mem220_allocate`** to obtain a block before using file-scope variables.

Initialization Uses File-Scope Variable and Static Function

What's the mechanism?

A **file-scope variable** and a **static function**
(not accessible outside the file).

```
static int32_t init_done = 0;
static void mem220_init ();
// And, at start of mem220_allocate…
if (!init_done) { mem220_init (); }
```

(**`mem220_init`** sets **`init_done`** to 1.)

Start with Local Variables and Initialization

Let's look at **`mem220_allocate`**.

```
void* mem220_allocate
    (size_t n_bytes)
{
    size_t      block_size;
    int32_t     bin;
    mem_block_t* new_block;
    if (!init_done) {
        mem220_init ();
    }
```

Number of bytes needed (including header).

Index into array of free block lists.

Pointer to new block.

Calculate Necessary Values and Check Arguments

```
block_size = n_bytes +
             sizeof (*new_block);
if (n_bytes == 0 ||
    block_size > MEM220_MAX_ALLOC) {
    return NULL;
}
bin = log2_ceil (block_size);
```

We need **`n_bytes`** plus a **`mem_block_t`**.

