# C-String Basics

**The library `string` type works as if it were built into the C++ language. It uses** C++ features to allow a `string` to act as a built-in type. C-strings are more primitive:

- C-strings are **`char` arrays** with a **sentinel terminator**, the `NUL` character `'\0'`.
- C-strings can be passed to functions without overhead.
- "String literals" automatically include the terminating `NUL`.

The literal ***"Hello, CS 150"*** contains `13` characters—`12` for the meaningful characters plus one extra for the terminating `NUL`. The compiler generates:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H | e | l | l | l | o | | C | S | | 1 | 5 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

C-string functions all **assume** that this `NUL` exists; some insert it for you. Without a `NUL`, functions don't know when the string stops, either returning garbage or crashing. The length of a C-style string is **not stored explicitly**; the `NUL` serves as a sentinel, and your program loops through the characters, counting them when it needs to find the size.

> *Don't confuse `'0'` with `'\0'`. One has the ASCII value `48` and the other `0`.*