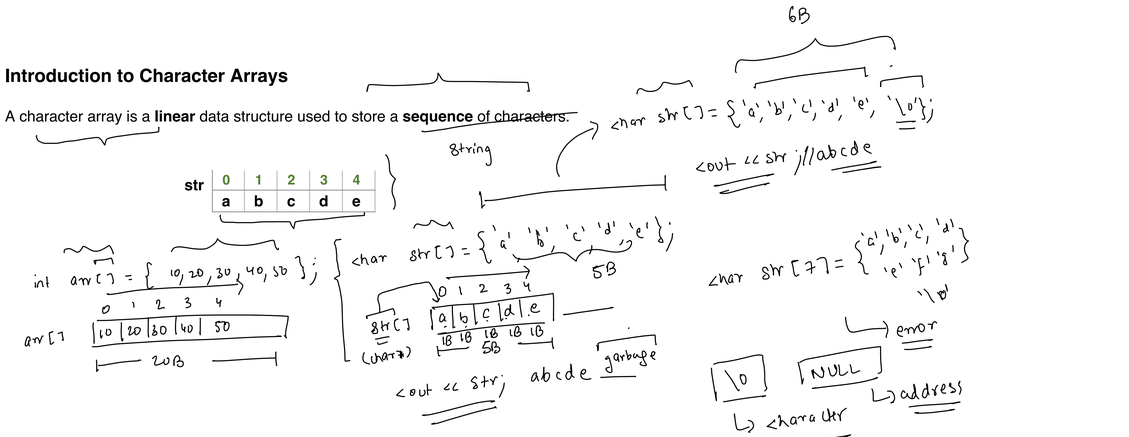
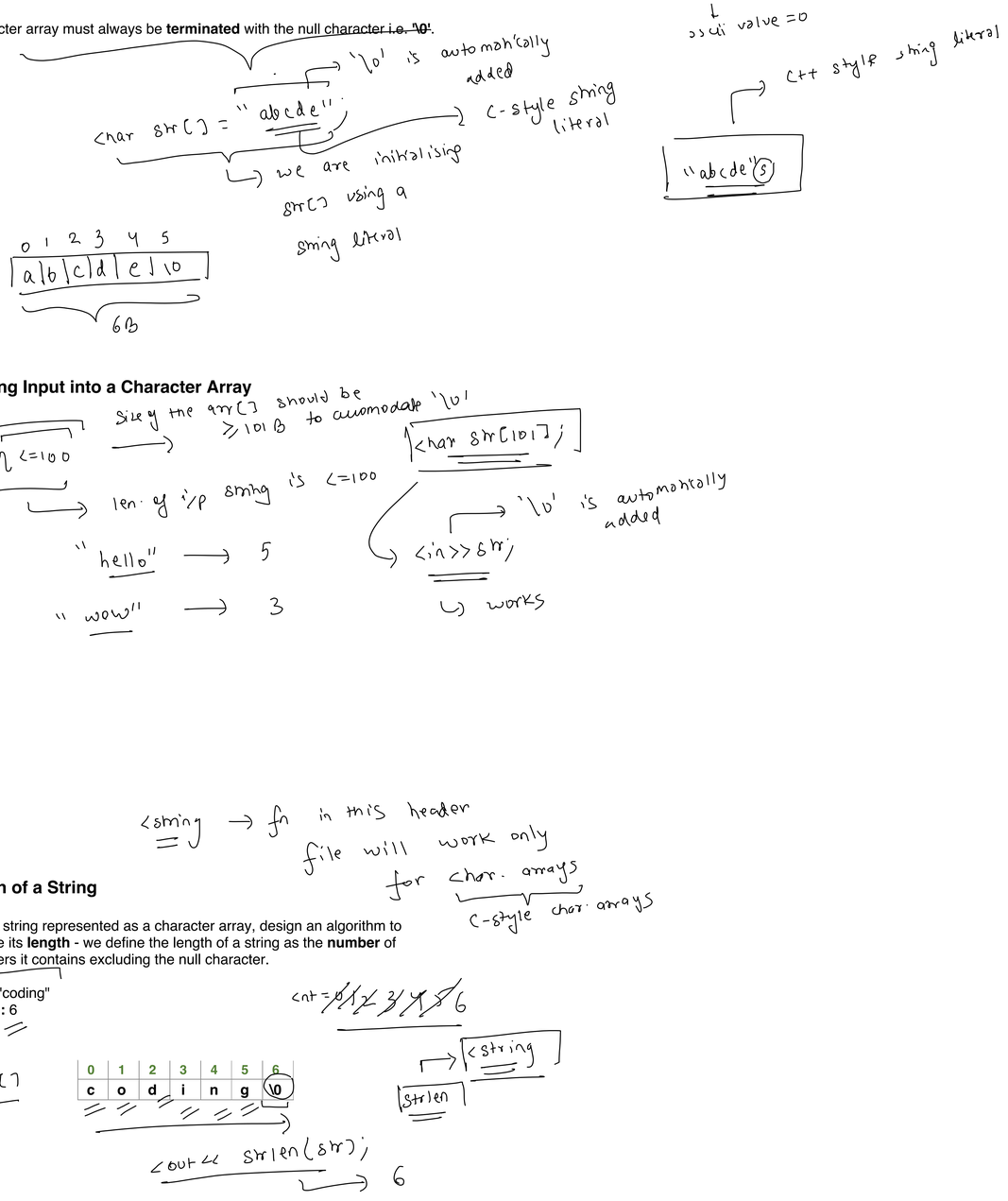
Introduction to Character Arrays

A character array is a **linear** data structure used to store a **sequence** of characters.



A character array must always be **terminated** with the null character i.e. **'\0'**.



Reading Input into a Character Array

<string> → fn in this header file will work only for char. arrays
C-style char. arrays

Length of a String

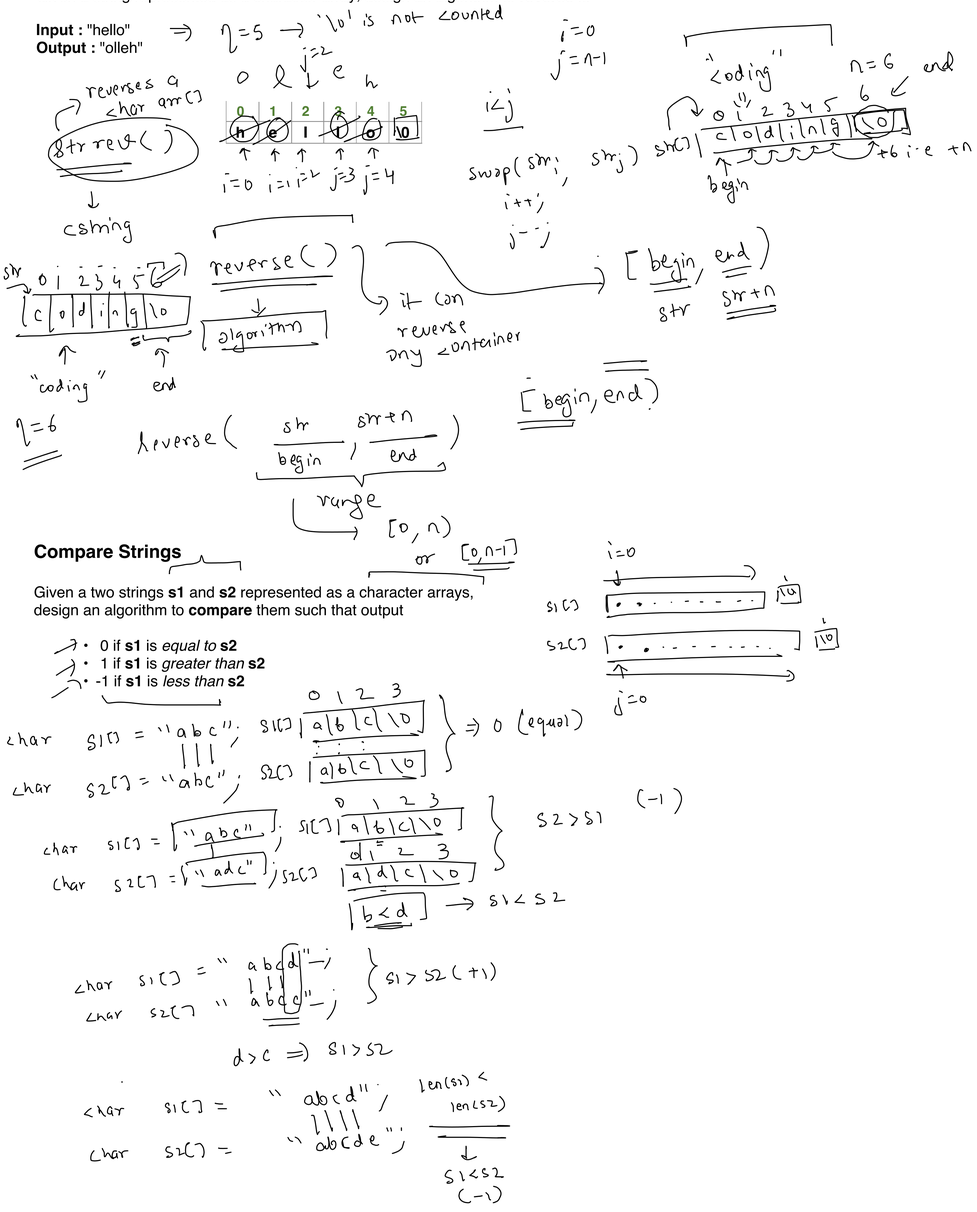
Given a string represented as a character array, design an algorithm to compute its **length** - we define the length of a string as the **number** of characters it contains excluding the null character.

Input : "coding"
Output : 6

Reverse a String

Given a string represented as a character array, design an algorithm to **reverse** it.

Input : "hello"
Output : "olleh"



Compare Strings

Given a two strings **s1** and **s2** represented as a character arrays, design an algorithm to **compare** them such that output

- 0 if **s1** is *equal to* **s2**
- 1 if **s1** is *greater than* **s2**
- -1 if **s1** is *less than* **s2**

Copy Strings

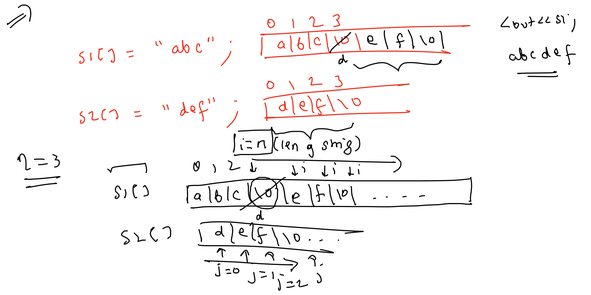
Given a two strings **s1** and **s2** represented as a character arrays, design an algorithm to **copy** the contents of s2 into s1.

note : assume length of s2 <= length of s1.

Concatenate Strings

Given a two strings **s1** and **s2** represented as a character arrays, design an algorithm to **concatenate / append** s2 to s1.

note : assume s1 has enough capacity to accommodate characters of both s1 & s2.



String Tokenization

Given a string represented as a character array, design an algorithm to **tokenize** it based on a **delimiting** string also represented as a character array.

Tokenization of a string is a process of **splitting** a string into **tokens** (substrings) such that consecutive tokens are separated by one or more delimiting characters.

Example

Input : "abc.def.ghi" and "."
Output : "abc", "def", "ghi"