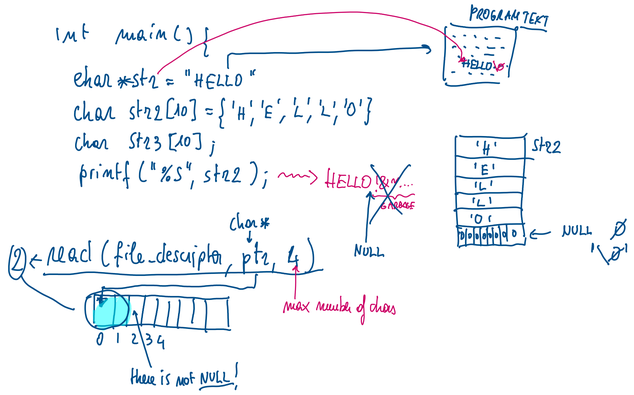
2.4 STRINGS AND BUFFERS

mercoledì 28 febbraio 2024 21:20



1. Strings in C
   - Text: "In C, strings are arrays of characters terminated by a null character '\0'."
   - Note: Explain that a string in C is not a data type but a convention for representing text.
2. String Declaration
   - Text: "Strings can be declared as character arrays."
   - Example: char str[6] = "hello";
   - Note: Highlight the null terminator at the end of the string, making the actual size one more than the number of visible characters.
3. String Initialization
   - Text: "Strings can be initialized in different ways."
   - Examples:
     - char str[] = "hello";
     - char str[6] = {'h', 'e', 'l', 'l', 'o', '\0'};
   - Note: Discuss automatic sizing of arrays and the importance of the null terminator.
4. String Manipulation
   - Text: "C provides standard library functions for string manipulation."
   - Examples: strcpy, strcat, strlen, strcmp vs. memcpy, memcmp
   - Note: Mention that these functions are part of <string.h> and briefly describe what each function does.
5. Pointer Representation
   - Text: "Strings can also be represented using pointers."
   - Example: char *str = "hello";
   - Note: Explain that this creates a pointer to a string literal, which is stored in read-only memory, and contrast with array-based strings that are mutable.
6. Common Pitfalls
   - Text: "Beware of common pitfalls when working with strings in C."
   - Points:
     - Forgetting the null terminator.
     - Buffer overflow vulnerabilities.
     - Modifying string literals causes undefined behavior.
   - Note: Emphasize the importance of security considerations and proper memory management.