3/5/24, 12:28 PM

OneNote

Char stry [10] = { H, E, L, L, 0 }

Char stry [10];

Char stry [10];

(hor * HELLO NOVEL

Char *

Char

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.

3. String Initialization

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.

Text: "Strings can be initialized in different ways."
Examples:

char str[] = "hello";
char str[6] = {'h', 'e', '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
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.

Modifying string literals causes undefined behavior.
 Note: Emphasize the importance of security considerations and proper memory management.

https://clickandfind-my.sharepoint.com/personal/zingirian_clickandfind_onmicrosoft_com/_layouts/15/Doc.aspx?sourcedoc={a0cadf98-59b4-4d67-b3a4-21516a59a24c}&action=view&wd=target%28CN2024.one%7C3756d2e0-a09c-4a5d-9a6a-40cbeee6ac6d%2F2.4 STRINGS AND BUFFERS%7C99ae7fca-e047-440a-998c-1667b791c027%2F%29&wdorigin=NavigationUrl