17. - ft_strncmp.-

Function based on the definition given in the BSD man pages for “strncmp(3)”.
The library associated is <string.h> (standard C library).

Synopsis:

```
int strncmp(const char *s1, const char *s2, size_t n);
```

Purpose:

Compares two strings lexicographically (alphabetically) up to a specified number of characters (`n`).

Parameters:

- `s1`: The first string to compare.
- `s2`: The second string to compare.
- `n`: The maximum number of characters to compare.

Return Value:

- Returns an integer representing the difference between the first non-matching characters:
    - 0 if the strings are equal up to `n` characters.
    - A negative value if `s1` is lexicographically less than `s2`.
    - A positive value if `s1` is lexicographically greater than `s2`.

Description:

- Iterates through the characters of `s1` and `s2` simultaneously, comparing them until a difference is found or `n` characters have been compared.
- Returns the difference between the characters that first differ.

Code

```
#include "libft.h"

int ft_strncmp(const char *s1, const char *s2, size_t n)
{
    size_t i;

    i = 0;
    if (n == 0)
        return (0);
    while ((n > i) && (s2[i] || s1[i]))
    {
        if ((s1[i] != s2[i]))
            return ((unsigned char)s1[i] - (unsigned char)s2[i]);
        i++;
    }
    return (0);
}
```

Code Explanation

1. **Handles Empty Comparison:** If `n` is 0, returns 0 immediately (strings are considered equal if no characters are compared).

2. **Iterates through Characters:**
    - Continues as long as `i` is less than `n` and either `s1` or `s2` still has characters to compare.
    - Compares the current characters of `s1` and `s2` at index `i`.
3. **Returns Difference:**
    - If a difference is found, returns the difference between the two characters' ASCII values (cast to `unsigned char` for proper comparison).
4. **Returns 0:** If no difference is found within the specified `n` characters, returns 0 (strings are considered equal).

Key Points:

- **Lexicographical Comparison:** Compares strings character by character based on their alphabetical order.
- **Character Casting:** Uses `(unsigned char)` to ensure correct comparison of characters, especially for extended ASCII values.
- **Maximum Character Limit:** Compares up to `n` characters, even if the strings are longer.
- **Null Terminators:** Stops comparison if either string reaches a null terminator (`\0`).

Main Function (Optional)

```
int main(void)
{
    // Define two strings to compare
    char str1[12] = "Hola, mundo";
    char str2[12] = "Adios, mundo";

    // Call ft_strncmp to compare the first 12 characters of the strings
    int result = ft_strncmp(str1, str2, ft_strlen(str1));

    // Interpret the comparison result and print a message
    if (result == 0) {
        printf("strings are equal.\n");
    } else if (result < 0) {
        printf("first string smaller than second one.\n");
    } else {
        printf("first string bigger than second one.\n");
    }

    return (0);  // Indicate successful program execution
}
```

Explanation of the `main` function:

1. **Declares strings:**
    - Creates two char arrays (`str1` and `str2`) to hold the strings to be compared.
2. **Calls `ft_strncmp`:**
    - Invokes the `ft_strncmp` function to compare the strings:
        - `str1`: The first string to compare.
        - `str2`: The second string to compare.
        - `ft_strlen(str1)`: Limits the comparison to the first 12 characters (length of `str1`).

3. **Interprets result:**
    - Checks the returned value of `ft_strncmp`:
        - 0: Strings are equal up to the compared characters.
        - Negative: `str1` is lexicographically smaller than `str2`.
        - Positive: `str1` is lexicographically larger than `str2`.
4. **Prints message:**
    - Prints an appropriate message based on the comparison result.
5. **Returns 0:**
    - Returns 0 to signal successful program termination.