18.- ft_memchr.-

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

Synopsis:

```
void *memchr(const void *s, int c, size_t n);
```

Purpose:

Locates the first occurrence of a character (`c`) within a memory block (`s`) of a specified size (`n`).

Parameters:

- `s`: The memory block to search within.
- `c`: The character to search for.
- `n`: The maximum number of bytes to search.

Return Value

Returns a pointer to the first occurrence of `c` in `s`, or `NULL` if `c` is not found within the first `n` bytes.

The memchr() function returns a pointer to the byte located, or NULL if no such byte exists within n bytes.

Description

Iterates through the bytes of `s` up to `n` bytes, comparing each byte with `c`.
Returns a pointer to the matching byte's position in `s` if a match is found.

The memchr() function locates the first occurrence of c (converted to an unsigned char) in string s.

Code

```
#include "libft.h"

void *ft_memchr(const void *s, int c, size_t n)
{
    unsigned char *ptr;
    unsigned char cs;
    size_t i;

    ptr = (unsigned char *)s;
    cs = (unsigned char)c;
    i = 0;
    while (i < n)
    {
        if (*(ptr + i) == cs)
        {
            return (ptr + i);
        }
        i++;
    }
    return (NULL);
```

```
}
```

Code Explanation

1. **Cast to unsigned char:** Converts `s` and `c` to `unsigned char` pointers for byte-level comparison.
2. **Iterates through bytes:**
    - Continues as long as `i` is less than `n` (the specified search limit).
    - Compares the current byte at `ptr + i` with `cs`.
3. **Returns pointer to match:**
    - If a match is found, returns the pointer to the matching byte's position in `s`.
4. **Returns NULL:** If no match is found within `n` bytes, returns `NULL`.

Key Points:

- **Memory Block Search:** Works with any memory block, not just strings.
- **Byte-Level Comparison:** Compares bytes directly for efficient searching.
- **Maximum Search Limit:** Searches up to `n` bytes, even if the memory block is larger.
- **Character Casting:** Uses `(unsigned char)` for proper comparison of characters, including extended ASCII values.

Main Function (Optional)

```
int main(void)
{
    const char str[15] = "Hello friend";
    char *ptr = ft_memchr(str, 'o', ft_strlen(str));
    if (ptr) {
        size_t position = (ptr - str) + 1;
        printf("First 'o' in the string is in pstn.: %lu.\n", position);
    } else {
        printf("Character 'o' not found in string, dear friend.\n");
    }

    return (0);
}
```

Explanation of the `main` function:

1. **Declares string:**
    - Creates a char array `str` containing the string to be searched.
2. **Calls `ft_memchr`:**
    - Invokes the `ft_memchr` function to find the first 'o' in the string:
        - `str`: The memory block to search (the string).
        - 'o': The character to search for.
        - `ft_strlen(str)`: Limits the search to the length of the string.

```
(size_t position = (ptr - str) + 1;  // Offset by 1 for human-readable position)
```

3. **Checks result:**
    - Checks the returned value of `ft_memchr`:

    - Not `NULL`: The character was found, and `ptr` points to its position.
    - `NULL`: The character was not found within the specified range.
4. **Prints message:**
    - Prints a message indicating whether the character was found and its position if found.
5. **Returns 0:**
    - Returns 0 to signal successful program termination.

ADDENDUM:

The notation `void *memchr;` in the context of the function `void *memchr(const void *s, int c, size_t n);` can be confusing. Let's break it down:

1. memchr function:

This function searches for the first occurrence of a character (`c`) within a specific memory area defined by a pointer (`s`) and a maximum number of bytes to search (`n`). It returns a `void*` pointer to the location of the character if found, or `NULL` if not found.

*2. void return type:**

The `void*` return type signifies that the function can return a pointer to **any** data type. This is because `memchr` can search for characters in various memory regions, not just in character arrays. It allows flexibility in usage without requiring specific type casting.

3. The void*memchr; declaration:

This **standalone declaration** is likely not related to the function definition. It's declaring a variable named `memchr` with a `void*` type, meaning it can hold a pointer to any data type. However, without context about its usage, it's impossible to explain its exact purpose.

Clarification:

- The function definition is `void *memchr(const void *s, int c, size_t n);`, not `void *memchr;`.
- The standalone declaration `void *memchr;` needs more context to understand its usage.

Key Points:

- `memchr` function searches for characters in memory regions.
- `void*` return type allows flexibility for different data types.