Arduino Lightweight Memory Allocator

A lightweight, configurable memory allocator for **Arduino** projects. This library provides a familiar `malloc`, `free`, `calloc`, and `realloc` interface on top of custom backends optimized for small embedded systems.

Unlike the default AVR `malloc`, this allocator avoids large runtime overheads and makes it easier to experiment with different allocation strategies (e.g., bitmap allocator, pool allocator, guarded allocator).

Features

- Drop-in replacement for `malloc`, `free`, `calloc`, and `realloc`.
- Backend abstraction layer (`internal_allocator.h`) for pluggable allocation strategies
- **Single level bitmap allocator backend** included as the default implementation
- Minimal overhead — only the selected backend is linked into your binary
- Portable: works on Arduino boards and can also be tested on desktop environments

Project Structure

- `src/ arduino_malloc.h` / `arduino_malloc.c`**:** Public API (`malloc`, `free`, `calloc`, `realloc`.). Adds small headers to track allocation sizes.
- `src/core/internal_allocator.h`**:** Dispatch layer that forwards allocation requests to the chosen backend.
- `src/core/allocators/1l_bm_allocator.h / .c`**: One-level bitmap allocator** backend. Manages a fixed-size heap divided into aligned blocks.
- `src/core/heap.h`**:** Defines heap parameters (**HEAP_SIZE**, **HEAP_START`) and abstracts Arduino vs. desktop builds.
- `core/utils.h` Utility functions used by the backend. (e.g., bit manipulation like `ctzb` and `clzb`).

Getting Started

1. Include the allocator

```
#include "arduino_malloc.h"
```

2. Allocate and free memory

```
void setup() {
    Serial.begin(9600);

    uint8_t* buffer = (uint8_t*) malloc(64);
    if (buffer) {
        Serial.println("Allocated 64 bytes!");
```

```
        free(buffer);
    }
}

void loop() {
    char *text = (char*)malloc(10);
    free(text);
}
```

Current Status

- malloc
- free
- calloc (to be implemented)
- realloc (to be implemented)
- One-level bitmap allocator backend
- Additional backends (to be implemented)

Testing on Desktop

For easier debugging, the allocator can be built in a desktop environment by providing a static heap array:

```
uint8_t heap[HEAP_SIZE];
```

This allows unit testing without uploading to Arduino hardware.

Example Output (Uno, 512B heap):

```
void *ptr1 = malloc(32)  -> success
void *ptr2 = malloc(64)  -> success
free(ptr1)               -> freed
void *ptr3 = malloc(512) -> fails (not enough blocks)
```

📜 License

MIT License. See LICENSE for details.