Overloading class new & delete
for managing our own chunk of memory

Introduction

Your goal is to create an allocator for a specific class. The allocator enables allocating and releasing a block from pre allocated memory.

The fact that all the blocks are of the same size saves us the need of keeping and managing block sizes. It eases the management of the memory blocks. We need to keep only a list of the free blocks.

Algorithm

For keeping a list of free blocks, we need a pointer to the head of the list. The head points to the first free block. Since the content of the block is garbaged, we can use it for our needs. We choose to put a pointer to the next free block in the block content space. The last block points to NULL.

When allocation request is received, we will return the pointer pointed by *head* and update the *head* to point to the next free block in list.

When release request is received, we will point *head* to the released address and point the content to the previous *head*.

Implementation

We will use the terrific ability to override new and delete operators.

First let's create a Person class with data: fullname (char[32]), id (unsigned int), age (unsigned char).

The class should have two static members:

- `s_pool` - Pool of memory to allocate the Person from (**not** array of Person)
- `s_firstFree` - head of the free list

The static class members must be initialized on system load. (before main - how do we do that?)

Initialization (all free): Make every block point to the next block. The last block points to `NULL`. `s_firstFree` points to the first block.

You should overload new and delete in the class and prevent using new[] and delete[]. In the new you should allocate the data from the pool (pop from the free list) and in delete release it (push to the free list).