

Class excersise on pointers and dynamic memory allocation

Write a `SensorArray` class that manages a dynamic array of doubles. The number of sensors is determined at runtime and must be allocated on the heap in the constructor. The class must also have a destructor that cleanly frees memory.

Class `SensorArray`

Attributes

- Pointer to the array on the heap
- Size of the array

Constructor

- Custom constructor that has the size of the array as a parameter
- The constructor allocated the memory for this array on the heap
- Prints a message "A array has been allocated @ *address* with a size of ..."

Destructor

- Frees the memory when the array is no longer needed
- Print a message "Confirming memory cleanup"

Methodes

- `FillValues()` : fill the array with random doubles, make use of pointer arithmetic.
- `Average()` : computes the average using pointer arithmetic
- `Min()` : returns the smallest value using pointer arithmetic
- `Max()` : return the largest value using pointer arithmetic
- `Sort()` : Sorts the array using pointer swap, no indexing, only pointer arithmetic
- `GetRawPointer()` : return the internal pointer
- `AddOffset()` : Adds offset to the value it points to

Main

- Ask the user for the size of the array
- Create a `SensorArray` on the stack
- Create a `SensorArray` on the heap
- Fill the array with random numbers with the `FillValues` method
- Output the stats Average, Min and Max by using the corresponding methods
- Demonstrate pointer aliasing using the `GetRawPointer` method
- Demonstrate pointer parameter passing using `AddOffset()`