Exercise session 03

Object oriented programming. Classes and access control in C++.

Advanced Programming - SISSA, UniTS, 2024-2025

Giuseppe Alessio D'Inverno

15 Oct 2024

Exercise 1 (1/5)

1. Create a class named `DataProcessor` with private data members for a data array and its size. The data array should be represented as a `double *data`.
2. Implement a constructor that takes an array of floating-point numbers and its size as input and initializes the class data members.
3. Implement a copy constructor, a copy assignment operator and the destructor.
4. Add a metod `n_elements()` that returns the number of elements in the array.
5. Test all these functionalities in the `main` function by creating proper instances of `DataProcessor` and displaying the results.

Exercise 1 (2/5)

1. Add methods to compute minimum and maximum values.
2. Add a method to compute the mean (average) of the data.
3. Add a method to compute the standard deviation of the data.
4. Add tests to validate these new functionalities.

Exercise 1 (3/5)

1. Organize the `DataProcessor` class by separating declarations and definition into separate header (`data_processor.hpp`) and source (`data_processor.cpp`) files.
2. Create a main program file that includes the header and demonstrates the use of the `DataProcessor` class for data analysis.
3. Compile the program using the following command:

   ```
   g++ -Wall -Wpedantic -O3 data_processor.cpp main.cpp -o data_processor
   ```

Exercise 1 (4/5)

1. Overload the output stream operator `<<` as a `friend` function to allow printing the list of values in the stored data, separated by a comma.
2. Overload the `[]` operator to allow indexing and accessing individual data elements. This operator will be used for both read and write access.
   ⚠️ **The folder `examples` contains two examples showing how to safely implement read and write access operators.**
3. Overload the `+` operator in the `DataProcessor` class to allow adding two `DataProcessor` objects. The result should be a new `DataProcessor` object containing the element-wise sum of the data arrays. The operator should also print an error if the two operands do not have the same size.
4. Add tests to validate these new functionalities.

Exercise 1 (5/5)

1. Ensure the const-correctness of all member variables and methods by adding proper `const` qualifiers.
2. Add a `static` member function `get_n_instances()` that returns how many instances of `DataProcessor` objects are currently active.
3. Implement a free function

   ```cpp
   double compute_correlation(const DataProcessor &dp1, const DataProcessor &dp2);
   ```

   that computes the Pearson correlation coefficient between two datasets with the same size.
4. Add tests to validate these new functionalities.