Draft of an exam for the AMSC course

Luca Formaggia

A.A. 2022/2023

1 Instructions

The exam consist of a series of questions concerning aspects of C++ programming or parallel programming with MPI and OpenMP, followed by questions concerning the development of a simple program.

You should create a folder named `NameSurname` (where of course Name and Surname are yours), please *avoid spaces in file and foler names!*. If a question implies production of code, you should

- Create a sub-folder names QuestionN, where N refers to the question number.
- Create the code in that subfolder.

Eventually, you eliminate temporary and object files (if any), create a zip of the `NameSurname` folder and upload it on webeep following the instructions on the webeep site.

2 Exam

1. Write the type of the following automatic variables

```
auto x=1;
auto y=3.14;
auto z=4ul;
```

2. What's the difference between `const int a=10;` and `constexpr int a=10;`?

3. Why is this code wrong?

```
#include <memory>
std::unique_ptr<double> pd=std::make_unique(350.7);
std::unique_ptr<double> pq{pd};
*pq=90.0;
```

4. Write a generic (template) function that takes any three values, possibly of different type, and returns their product as an automatically deduced type.

5. What is the difference between `MPI_Gather` and `MPI_Gatherv` and which is a typical use of the latter?

6. Rewrite the following piece of code creating a parallel version for OpenMP and one for MPI (you can reorganize the loops in a more standard fashion.)

```
std::vector<double> a;
std::vector<double> b;
double result=0.;
...// a and b are filled with values
auto starta=a.cbegin();
auto startb=b.crbegin;
for (auto const & va=starta;va!=a.cend(),++a)
   {
        result += *va * *startb;
        ++startb;
    }
 std::cout<<result;
```

7. Write a working function and the corresponding test code (as separate files) for the following algorithm to sort a vector of values in ascending order (bubble sort)

   - Loop over the index s of the vector v;
     - Loop for i ranging from 0 to the size of the vector minus s;
     - if $v_i > v_{i+1}$ swap the two elements;
   - return v;

   The code must take in input a standard vector as a constant vector and return the vector with the sorted elements. The test code should create a vector with elements not ordered, print the elements, run the sorting function and print the sorted vector. You must provide a simple Makefile so that the executable for the test code is produced just by typing "make". To gain more points, you may write a function template instead of a function.

8. A possible parallel implementation of the bubble sort algorithm is based on the successive application of odd-even phases. During the even phase elements i and $i+1$ with $i = 0, 2, 4, \ldots$ are swapped, if necessary; in the odd phase instead, $i = 1, 3, 5, \ldots$. Odd and even phases may be run in parallel since all possible swaps are independent. Another difference is that here you stop when no swap have occurred

after the succession of an odd and even phase (you can keep track of the swaps with a boolean). Implement an OpenMP parallel version. Again the code must be working and compiled with the aid of a Makefile. Extra points for an MPI version.