EPFL Fall 2019
Master level
MATH-458
November 5, 2019

Programming Concepts in Scientific Computing

Introduction to Standard Library (STL)

The goal of the present exercises is to discover and manipulate the STL structures in C++11 fashion.

Exercise 1: *std::vector*

- In a main function, create a vector of double.

```
std::vector<double> array;
```

- In order to append items to the vector you can use the method 'push_back'. For example.

```
array.push_back(1.);
array.push_back(2.);
array.push_back(10.);
```

- Append 100 values ranging from 0 to 2π (π is defined under the standard name M_PI)
- Print the size of the vector to screen

```
std::cout << array.size() << std::endl;
```

- If you want to resize the vector you can use the 'resize' method. If you enlarge the vector, what values will be contained in the extra part of the vector ? What is the 'assign' method doing ?
  As example C++ Reference website is of great help to explore methods of STL classes.
- In order to iterate over the values contained in the array a standard loop can be written like:

```
std::vector<double>::iterator it = array.begin();
std::vector<double>::iterator end = array.end();
for (; it != end ; ++it){
  double & val = *it;
  ... what you want ...
}
```

- Write such a loop with C++11 syntax, using a **range-loop** and the **auto** keyword in order to construct another array (named sin_array) that contains the sinus values of these other values.
- Re-Write it with a foreach.
- Iterate once more to generate a file containing two columns (CSV format) with the values of array and sin_array. Scientific notation, with 20 digits of precision is expected. You can apply a lambda functor to factor this loop to the maximum.

Exercise 2: *std::map*

- Define a structure Triplet. Remember that a structure acts like a class but with public members by default. Class members are private by default, which means they cannot be accessed outside of the class scope.

```
struct Triplet {

 Triplet(){}; // Default constructor

 Triplet(double x, double y, double z) {
   // Constructor to be implemented here
 };

 double coords[3]; // Public member
};
```

- Implement a constructor building Triplet object with a set of coordinates.

```
Triplet origin_coordinates(0.,0.,0.);
```

- In the main function, create a map from string to Triplet.

```
std::map<std::string,Triplet> map;
```

- Let us use this map to store relative coordinates of planets. In order to append items to the map you can use the '[ ]' operator:

```
map["sun"] = origin_coordinates;
```

- Add the earth, with the associated coordinates Triplet being (1.,0.,0.) in astronomical unit.
- What happens if you try to append an item with an already-existing entry ?
- Prevent this issue using the *map::find* method:

```
auto it = map.find("earth");
```

- Loop over the entries of the map and output the entire content to screen