Object Oriented Programming with C++
Labwork 2 – *C++ special features*

Exercise 1: Reference

Let be the following program excerpt:

```
int main() {
  const int array[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
  const int *p1 = array+1;
  const int *p2 = array+8;
  swap( --p1, ++p2 );
  std::cout << *p1 << "␣and␣"<< *p2 << std::endl; // should write 9 and 0 ...
  return 0;
}
```

Complete and then test this program. The swap function must use references to pointers `p1` and `p2`.

Exercise 2: Default arguments

Write a small program that offers one function with some arguments, some with Default Value (DF), other not. More precisely, we want a first integer with DF 0, a second integer with DF 1, a float argument without DF, and at last another integer with DF 2.

Check the behavior using all the possible combinations (in number of arguments) in the function `main`.

Exercise 3: Function overloading

Write homonymous functions that print their unique argument and its type. The argument can be of type `int`, `unsigned long`, `float` or `double`. Check various calls, including others fundamental type (e.g. pointer and string). What happens? How to solve this to obtain a result?

Exercise 4: Linking with C module

1. Build a C-*module* that exposes one global variable named `vi` of type `int` and initialized at 10, and a function of prototype double f(const int i); which implementation may be for instance { return exp((double)i); }.
2. Write a C++ program that prints the value of the global variable `vi`, and then calls function `f` using some values chosen by program-user, stopping the process for instance with value 0.

Exercise 5: Dynamic memory management

You just have to test the two programs presented on Lecture2 slides 18 and 19.

Exercise 6: Namespaces

First, test the program proposed on slide 22, Lecture 2.

Then modify it so that in function `A::f` you can add a call to the function `B::f`.

At last, in namespace B, add a function `g` that calls the function `A::f`. Call this function from the `main` function.

Exercise 7: A simple complex example

Write a library for complex numbers manipulation. You should start with a structure for complex numbers. Then add the important inline functions, like addition, multiplication, transformation between Cartesian representation to and from polar representation.

Write a testing program, that checks all your methods. Notice that, generally, for testing purpose the programmer who write the testing unit is not the one who wrote the tested unit ...