Week 2 — *C++ starter: Hello world*

The goal of the present exercises is to discover a simple "Hello" program.

Exercise 1: *Exploiting the main arguments*

- Get the sources by updating your cloned copy of the SP4E repository

```
git pull upstream master
```

- Change directory into the sources

```
cd exercises/week2/hello/sources
```

- Compile the program by using the CMake (either ccmake command or CLion)
- Observe the command that is automatically typed.
- Launch the program

```
./hello
```

- What is the nature (type) of `argc` and `argv` arguments to the function main ?
- To convert a string argument to an integer variable you can use the 'atoi' routine (use the 'man' command if you seek for the information about that function)
- Modify the main function so that the message printed to screen should be 'Hello N' with N being a parameter passed when launching the program.

Exercise 2: *First loop*

- Modify the program so that the program first computes the series

$$S_n = \sum_{k=1}^{n} k \tag{1}$$

  where n should be taken as an argument and the result should be printed aside of the 'Hello'.
- How many operations are necessary to perform this series computation ?
- Considering the analytic prediction what is the overhead ?

Exercise 3: *Source file and file headers*

- The call to the compiler is explicitly expressed in the `Makefile` file. Please open the file and consider the details.
- Split the obtained program in three files:
  1. hello.cc
  2. series.cc
  3. series.hh

The file `series.hh` should contain the declaration of a function `computeSeries`.

```
int computeSeries(int Niterations);
```

The file `series.cc` should contain the definition of that function.

```
int computeSeries(int Niterations) {
  // ...
}
```

- Modify the `CMakeLists.txt` so that it compiles first the object files `hello.o` and `series.o` and then only make the linking operation to provide the final executable.

Exercise 4: ***Debugging***

- Compile the `hello` code using `ccmake` .. and setting `CMAKE_BUILD_TYPE` to `Debug`. For CLion users, in the *Setting*, *CMake Setting*, change the build type to `Debug`.
- launch `gdb` manually from a terminal or use the interface of your favorite IDE (we recommend CLion)

```
gdb hello
```

- Set a breakpoint to the main function

```
(gdb) break main
```

- Run the program and pass the correct arguments

```
(gdb) run arg1 arg2 arg3...
```

- Step over each instruction of your main function until the end of the program

```
(gdb) next
...
(gdb) next
```

- re-run the program, but this time enter the function you created using the `step` command.
- Advance into the loop and print the content of your counter

```
(gdb) print i
```

- make a conditional breakpoint for when the counter is equal to 10

```
(gdb) break if i == 10
(gdb) continue
```

- Change the value of the counter (during the execution of the program) back to zero

```
(gdb) set var i = 0
(gdb) print i
```