# Lab1 Tool Chain and STL

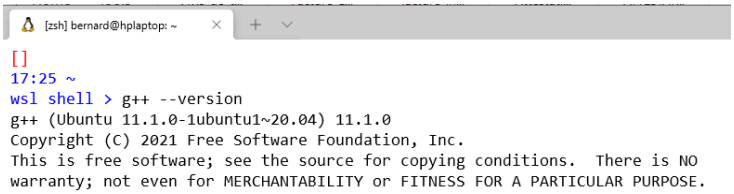
Lab Objectives:

- Install a compilation tool chain on your laptop.
- Compile and test a small program.
- Gain competencies on the C++ Standard Template Library (STL)

## 1 - Tool Chain Install

We shall use GNU g++ compiler for this lab in a Linux environment which could be either a native Linux machine or a Linux virtual machine. Information about the gnu tool chain can be found here: https://gcc.gnu.org/.

1) If you already have a laptop running Linux, you don't have much to do. Just ensure that you can open a terminal and at the terminal prompt check the g++ version, make sure that you have a version 9 or above installed.




If you need to install a newer version of g++, you have to follow the install procedure for you Linux distribution using the proper package management command (`apt`, `yum`, `dnf`, ...).

You could also be in a situation which requires a specific version of `g++` for another project, please check `update-alternatives` to switch between different versions.

2)` If you only have a Windows machine (Windows 10), you can install a Linux virtual environment, several options are possible. I describe the following two in the appendix.

- WSL (Windows Subsystem for Linux)


- MSYS2

# 2 - Check your Tool Chain

1) Please answer the following questions:
   Which Linux distribution are you using, version?
   Which version of `g++` are you using?
   Which version of `gdb` are you using?

2) Using a Linux terminal, create a new folder which shall contains all your projects for the labs of this class, for example:
   `mkdir elec4cpp`,
   go to this new folder:
   `cd elec4cpp`
   and clone the *github* `mean_and_median` archive:
   `git clone https://github.com/elec4/mean_and_median.git`

3) Change to the new folder created by `git` (`cd mean_and_median`) verify that you have all the files:

```
wsl shell > tree -n
.
├── Makefile
├── data
│   ├── data_10.txt
│   ├── data_100.txt
│   ├── data_1000.txt
│   ├── data_10000.txt
│   ├── data_100000.txt
│   ├── full_10.txt
│   ├── full_100.txt
│   ├── full_1000.txt
│   ├── full_10000.txt
│   └── full_100000.txt.gz
├── mean_and_median.c
└── mean_and_median.cpp
```

4) Then compile the file `mean_and_median.cpp` with a `g++` command at the prompt, verify that the program runs well using any of the data files, `data/data_1000.txt` for example.

5) Also check that you can use the `Makefile` to compile the program and compile for the debug version.
   `make`

6) Run the program under the debugger (`gdb`) and answer the question: what is the type of the variable `mid`?

```
auto mid = buf.size() / 2;
```

# 3 - STL `string` and `vector<>`

You shall now write a simple program which prints an histogram of data points read from a simple text file using the following requirements:

- When reading the data file, only consider values between 0 and 7999.99
- Create buckets of equal width: 100, a value $v$ belongs to bucket $b$ if: $100b \leq v < 100b + 100$.
- To display an histogram on the console, you shall print 60 stars for the bucket with the largest count, the number of stars for the other buckets shall be proportional.
- The output of your program on the file `data/data_100000.txt` must match exactly the following:

```
shell> histogram.exe data/data_100000.txt
number of elements = 99957, median = 1715.81, mean = 2124.64
    0       0
  100       0
  200     334 ****
  300     857 ***********
  400    1706 **********************
  500    2392 ********************************
  600    3130 *******************************************
  700    3684 ****************************************************
  800    3948 *******************************************************
  900    4270 ************************************************************
 1000    4466 ****************************************************************
 1100    4432 ***************************************************************
 1200    4327 ************************************************************
 1300    4175 ********************************************************
 1400    4031 ******************************************************
 1500    3897 ****************************************************
 1600    3739 ***************************************************
 1700    3388 *********************************************
 1800    3334 ********************************************
 1900    3140 *******************************************
 2000    2856 ***************************************
 2100    2656 ************************************
 2200    2471 *********************************
 2300    2340 ********************************
 2400    2155 *****************************
 2500    1958 **************************
[ lines deleted to fit in the page, not part of the expected output]
 4800     350 ****
 4900     342 ****
 5000     290 ***
 5100     280 ***
 5200     279 ***
 5300     239 ***
[ more lines deleted to fit in the page, not part of the expected output]
```

60 stars

Hints:

- Start from a copy of `mean_and_median.cpp` , rename it as `histogram.cpp`
- You must update the Makefile to facilitate compilation
- For additional static check, you can use `cpplint` a simple checker from Google. A recent version of python (> 3.5) must be available on you distribution, you can then install easily as show below:

```
wsl shell > pip3 install cpplint
Collecting cpplint
  Downloading cpplint-1.5.5-py3-none-any.whl (77 kB)
     |████████████████████████████████| 77 kB 1.3 MB/s
Installing collected packages: cpplint
  WARNING: The script cpplint is installed in '/home/bernard/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-lo
cation.
Successfully installed cpplint-1.5.5
```

run it:

```
wsl shell > ~/.local/bin/cpplint  mean_and_median.cpp
mean_and_median.cpp:0:  No copyright message found.  You should have a line: "Copyright [year] <Copyright Owner>"  [legal/copyright] [5]
mean_and_median.cpp:21:  Line ends in whitespace.  Consider deleting these extra spaces.  [whitespace/end_of_line] [4]
mean_and_median.cpp:27:  Line ends in whitespace.  Consider deleting these extra spaces.  [whitespace/end_of_line] [4]
mean_and_median.cpp:29:  Line ends in whitespace.  Consider deleting these extra spaces.  [whitespace/end_of_line] [4]
Done processing mean_and_median.cpp
Total errors found: 4
```

- Don't write low-level function, like the search of a max value in a vector:

```
  vector<uint32_t> v;
...
  uint32_t max = 0;
  for (int i = 0; i < v.size(); ++i) {
    if (v[i] > max) {
      max = v[i];
    }
  }
```

The STL library is very large, a bit of searching and you will find how to replace this low-level code by a simple call to an STL method.

# 4 - STL: Trees and Hashes

If you are not familiar with binary trees and hash tables, please read these well done explanatory pages: https://en.wikipedia.org/wiki/Red-black_tree and https://en.wikipedia.org/wiki/Hash_table

## 4.1 One Way

The file `data/full_10.txt` has two columns: the first is a unique identifier, the second is a number.
Example

```
shell> cat data/full_10.txt
fea0536b7a94    2615.93
84fd1c80659c     863.93
8a5b74971f70    1990.52
2ffeabe25cb0    2815.77
44e5db4dbe09    1181.31
179d95de9a47    1321.13
2b3ace2e711d     455.36
911708687b4d     812.47
8789d2ce5b3b    2638.90
bbd9c8b1b456   17301.72
```

You shall now write a small program (`map1.cpp`) which reads the given file, then proposes a prompt to the user. The user enters an identifier, when the user hits the return key, the program shall display the number of the second column matching the given identifier. After the display, the program iterates and proposes again a prompt to the user...
If the user enters the word END the program stops, if the user enters an invalid identifier, the program displays an error message.

Expected results:

```
shell> map1 data/full_10.txt
query> 8789d2ce5b3b
value[8789d2ce5b3b]= 2638.9
query> 8a5b74971f70
value[8a5b74971f70]= 1990.52
query> 8a5b74972f70
This ID does not exists
query> END
Bye...
```

In this example, the prompt is `query>`. I have used italic font for the user inputs, the red color clearly shows the difference between the two identifiers: a valid one and a invalid one.

Hints:

1. To read a file with several fields:
   https://en.cppreference.com/w/cpp/io/basic_istream/operator_gtgt

2. You certainly need an STL container from this list:
   https://en.cppreference.com/w/cpp/container

3. Code snippet to help you get started

```
    ...
    string qin;
    for (;;) {
      std::cout << "query> ";
      std::cin >> qin;
      ...
  }
  std::cout << "Bye..." << std::endl;
}
```

4. No more than 50 lines of codes is needed here

## 4.2 Two Ways

You shall now improve your previous program to obtain a better program (`map2.cpp`) to perform forward and backward searches. When the user enters a number $v$, the program shall return all the identifiers with value $v \pm 1\%$.
Select your container carefully, multiple identifiers can have the same value.

Example:

```
shell> map2 data/full_1000.txt
query> 44e2d4b8d7aa
value[44e2d4b8d7aa]= 1358.56
query> +5000
value[375df8b1ac86]= 5022.42
query> +614
value[f6a5f1e9f733]= 612.69
value[7860f4b10a57]= 615.25
value[1201267a89a7]= 615.25
query> END
Bye...
```

Hints:

1. No more than 15 additional lines of code to write here.
2. If you are iterating over a vector of identifiers, you will only get partial credit.

# 5 - Deliverable for this Lab

1) Your source code with files: `histogram.cpp`, `map1.cpp` and `map2.cpp`

2) A *pdf* document with screen captures to show that all your programs worked as specified as well as some additional details on your implementation.

3) The document must explain your choice of containers in Section 4.1 and 4.2

4) The document must allow explain the complexity in big O notation of a query, with *n* as the number of lines of an input file ( Section 4.1 and 4.2).

# 6 - Extra Credit

Use pseudo number generators and distributions of the STL to write a program to produce a data file such that it's histogram would be equivalent to the one you have obtained with `data_100000.txt`

# 7 - Appendix: Linux on Windows

Best method by far: WSL.

## 7.1 MSYS2: Method

From https://www.msys2.org/
Follow the installation instructions.

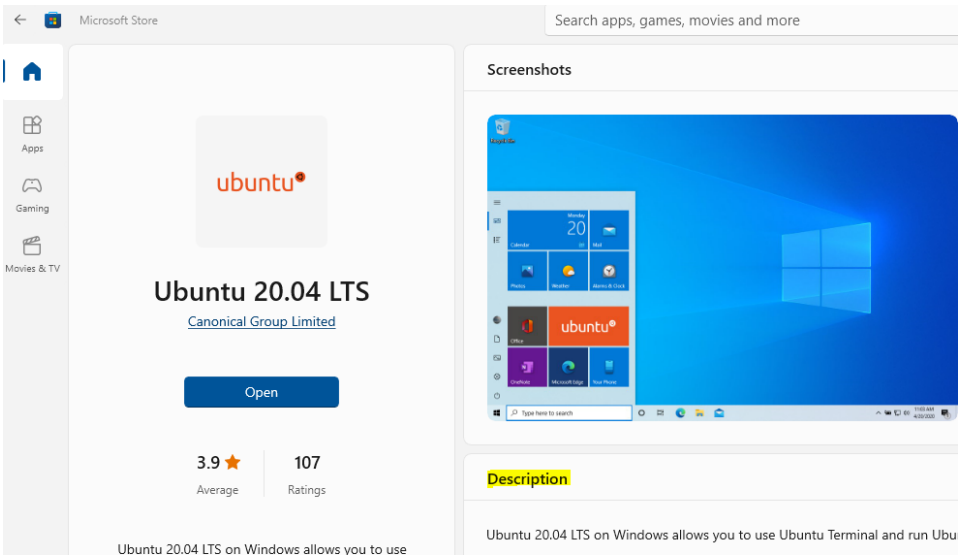
Once at the terminal, you have to install these extra packages to complete the lab:

```
shell> pacman -Sy python
shell> pacman -Sy git make tar wget
shell> pacman -Sy base-devel
shell> pacman -Sy mingw-w64-x86_64-gcc
shell> pacman -Sy mingw-w64-x86_64-gdb
```

## 7.2 WSL

WSL is the Windows Subsystems for Linux, a recent addition of Windows 10 which allows you to run a Linux kernel and open a Linux terminal on your Windows machine.

1) Select Ubuntu from Microsoft store:




Read Carefully the “Description” section (in yellow)

## Description

Ubuntu 18.04 on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

Please note that Windows 10 S does not support running this app.

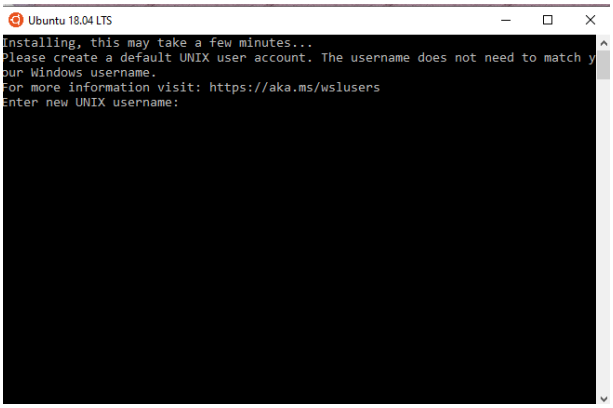
To launch, use "ubuntu1804" on the command-line prompt (cmd.exe), or click on the Ubuntu tile in the Start Menu.

To use this feature, one first needs to use "Turn Windows features on or off" and select "Windows Subsystem for Linux", click OK, reboot, and use this app.

The above step can also be performed using Administrator PowerShell prompt:
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux

This app installs the Ubuntu 18.04 LTS release on Windows.

2) After the install, you will see this window:



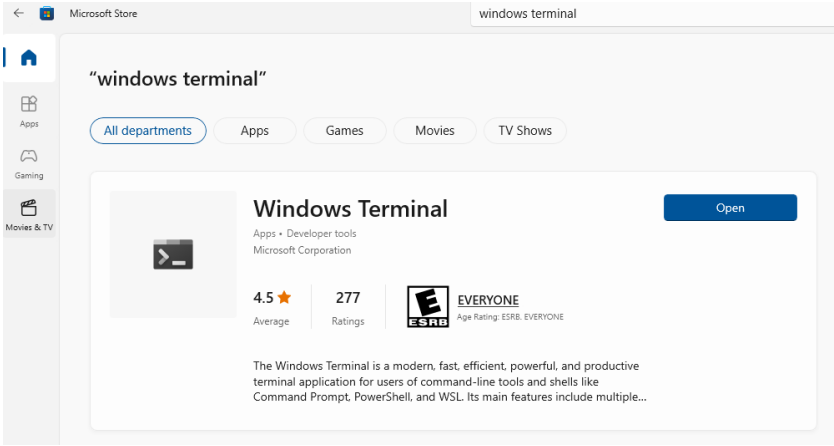

3) Additional packages must be installed:

```
shell> sudo apt-get update
shell> sudo apt-get upgrade
shell> sudo apt install gcc++
shell> sudo apt install gdb
```

4) Check that everything is fine:

```
wsl shell > git --version; g++ --version; python3 --version
git version 2.25.1
g++ (Ubuntu 11.1.0-1ubuntu1~20.04) 11.1.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

If you are not happy with default terminal, you can use the “Windows Terminal” from the store:




It’s a tab-based terminal, you can run in different tabs `Windows owershell` and `Linux`: