Homework. 3: C++Functions

Ignacio Vizzo, E-Mail ivizzo@uni-bonn.de

Handout : 08.05.2020
Handin: 22.05.2020 at 23:59:59 (CET)

General rules

1. You need to provide the build system for this homework. This means, you need to provide as many **CMakeLists.txt** files as you think is needed.
2. Your code will be checked using `clang-format`. If your code in not properly formatted, then the tests will not run.
3. Your code will be static-analyzed by `clang-tidy`, if your code does not comply with the analysis, the tests will not run. If you are using VSCode with the reccomended setup, most of the errors will be visible while you program.
4. **ALL** tasks should be solved within the `homework_3` folder, no need to create `task_x` folders.
5. You get the change to design where the files should, the **ONLY** requirement to pass the tests is that you should provide one header file(put it where you preffer) that's called `homework_3.h`. This is the only header file the hw_bot will use and it should acesss all the functions of this exercise. You should guarantee this header file can be included thorugh the build system.

A Image Browser Web Application (10 points)

1. **(10 points)** In this exercise you will write a small utility library that will create a web application for you. The code from this exercise will be used for your final project application, so the better you solve this exercise, the better your final project will be.

   You will show using the web browser some images taken from the `KITTI` dataset, sequence `00`.

   NOTE 1: Before continue reading, you should download the files provided with this homework sheet and inspect the **web_app** folder, there is an example `HTML` file in there, you can open the file in any web browser and see how the overall results should look like. You could also briefly skim thorugh the `example.html` file to get familiarized with the `HTML` syntax.

   NOTE 2: While is not mandatory for this exercise, it's highly reccomended to write a test program that uses the library and creates a web browser application. The web app could be easily inspected with any browser. The homework bot will not check for any "main" function, so all the tests will be over the functions you write.

A.1 `html_writer` library

In this task you will create a rather small utility library that should help you create HTML web applications, don't panic, it's actually quite easy. The API specification for this library is defined in the `html_writer.hhp` header file provided with this sheet. Your task is to implement all the functions **declared** in this header and provide more if you feel is necessary.

How to create HTML files? To keep it really simple, your task is to print line by line the HTML to the **stdout**, this could be redirected to any html file and that's how you create the web application. This means that if your test application print a debug msg, it will appear in the HTML file and it will most likely be an error. We will learn how to do this properly in the near future, but for now, let's keep it simple.

A.1.1 Requirements to pass the tests

- Your build system must generate a library called **html_writer**. This library must contain all the definitions of the functions declared in the API.
- The library will only process `png` or `jpg` files. if any other type of extension is used, it should print an error to `stderr`.
- When you add an image to the web app using the `AddImage` function you need to consider the following rules:
  (a) You should be able to highlight this image if the user requires to(see web app example, first image)
  (b) The title of the image should be **Only** the filename of the image. It should not contain any other path.
  (c) The score of the image should be reported **Only** using 2 decimal digits.

A.1.2 *Tips*

(a) You could use uhe **fmt** library instead of **iostream**. While there is nothing wrong with it, the main issue with iostreams is best illustrated with an example:n example:

```
std::cout << std::setprecision(2) << std::fixed << 1.23456 << "\n ";
```

Could be re-written as:

```
fmt::print("{:.2f}\n ", 1.23456);
```

More information about **fmt**: `https://fmt.dev/`

(b) The C++standard library provides a filesystem library that could be used to simplify some operations.

A.2 `image_browser` library

In this task you will extend the `html_writer` library adding one level more of abstraction. You will extend this library by implement the functions defined in the `image_browser.hpp`. This small library depends on `html_writer`. The `CreateImageBrowser` function defined in this library is the **most** relevant function of the exercise. This function should be able to create an image browser web application from scratch without the need of calling any other function.

A.2.1 Requirements to pass the tests

- Your build system must generate a library called **image__browser**. This library must contain all the definitions of the functions declared in the API.
- The very first image that is being add to the web application must be highlighted. You will test functionality written in the parent library at this point.
- `CreateImageBrowser` should be completely self contained, if the user call this functions with the right arguments, a complete web application should be printed to the `stdout`.

A.2.2 *Tips*

(a) Don't worry if you don't understand the types defined in the `image_browser.hpp` header file, we will study those types in the near future, for now, it's just a way to make your task easier to solve(and to understand).

(b) While is not mandatory, while inspecting your HTML file in the text editor you could install a formatter like `prettier` to help you find errors.

(c) The HTML output should be correct, you could make usage of `xmlint` to test this fact by just doing:

```
xmllint --html test.html > /dev/null
```

This will print(if any) the errors found in the HTML file.

A.3 Usage Example

```
$ cd cpp-homeworks/homework_3
$ ./bin/create_image_browser > web_app/test.html
$ xdg-open web_app/test.html
```