Shahed X-1

The standard for code-sources. Easy-to-understand.

I don't like large documentations that are impossible to read (600 pages each), let alone follow them.
Therefore I am. I'm Isfahan.

1. Please add information in the readme.md file about how to configure your project from A to Z on all available platforms.

1.1. If it's about the Windows OS, please provide support for non-VSCode IDEs (like CLion)

1.2. If possible, use CMake for the project configuration.

2. It would be helpful to add a file named "shahed.rst" in the root of the code space, which provides comprehensive information about the project's sources (the main information about the project can be found in the readme.md file).

Here's an example for a project pseudo-linuxkernel (shahed.rst file):

The main implementations is located in the "linux/" folder. Inside this folder, you will find the following subfolders:

- .gitignore
- ico.png
- linux/
    - drivers/ - Contains drivers for Ethernet, Mouse, Keyboard, USB ports, etc.
    - legacy/ - Legacy files for supporting old versions
    - pre-install/ - Pre-installation files and programs
    - secure/ - Information about users access rights
    - startup/ - Code that runs when computer starts
    - terminal/ - Terminal commands
    - …
- readme.md
- some_more/ - Additional files |-_-|
    - …
    - …

2.0.1. Please note, that this file should contain the majority of the defines used in the project.

Here's an example (continuation of shahed.rst):

- PI 3.14 - The value pf PI
- Linux_INT int64_t - Integer data type
- Linux_STR std::string - String data type

2.0.2. The file should also include the major principles or conventions followed in the project.

For example, If your projects uses prefixes

"PUB_" for all functions callable from outside,
"PRIV_" for private functions, and
"FUNC_" for other specific functions,

Make it explicitly clear in the file.

2.1. In every code folder, there should be similar file "{folder_name}.rst" that provides an explanation of what the folder does and information about each file it contains.

Here's an example for the "drivers" folder from the previous paragraph:

- drivers/
    - available.h - Checks if drivers are available for your computer
    - ethernet.c - Ethernet port driver
    - ethernet.h
    - initial.cpp - Initial drivers for the computer
    - keyboard.c - For Keyboard (USB only, not PS/2)
    - keyboard.h
    - legacy/
        - parallel.c - Parallel port driver
        - parallel.h
        - vga.c - VGA port driver
        - vga.h
        - ps2.c - Driver for old keyboards (PS2)
    - mouse.c - Mouse driver (USB and PS/2)
    - mouse.h
    - usb.c - USB support driver
    - usb.h
    - …

2.1.1. Attention! For folders like "**legacy/**" not required to have an rst file.

This approach is only applicable to **small intuitive** folders that provide a description of all the files in preceding larger folder (like "**drivers/**")

2.1.2. Description of the header file is not required if it only contains declarations similar to corresponding source (C/C++) file.

2.3. In every file, it is allowed to provide a more detailed description at the beginning of the file, although this is optional. However, a description is required for every function.

Here's an example for the "keyboard.c" file in the "drivers/" folder:

```
// The function determines the connected keyboards
// It returns an STL vector of integers with the USB port numbers where keyboards are plugged in
int determine_plugged(int numbers_of_ports):
    // numbers_of_ports: the number of available ports on the computer
    std::vector<int> res;
    for (int x = 0; x < numbers_of_ports; ++x) {
        // do something
    }
    return res;
```

2.3.1. In addition to providing descriptions for each function, you have the option to split a large function into smaller ones. These smaller functions can be inline functions that are only called from the parent function. Alternatively, you can choose to keep everything in one large function but break it down into logical parts and clearly indicate what each part does.

3.1. For smaller projects, you can organize your source files as follows:

- dep/ - Miscellaneous files such as .jpg, .mp4, and others (this folder can be named "misc" as well)
- include/ - Header files (.h, .hpp, and others)
- src/ - Source files (.c, .cpp, and others)

Optional:

- platform/ - Platform-dependent things
- tests/. - Test files

3.2. For larger, it's recommended to manually organize your code into folders based on your specific project architecture. Take the time to

carefully plan and structure your project before creating the necessary folders.

4. While using RST format for documentation is not required, it’s preferred. You can use any format that suits the main content of your documentation.