Median C++ Functional

MICROELECTRONIC SYSTEMS DESIGN RESEARCH GROUP

FPGAHS Lab - Median C++ Functional
Dr.-Ing. Christian De Schryver

Goals

- Create a functional reference of a median filter implementation.
- Work with loops and 2D-arrays.
- Run the C++ binary and see the output.

Setup

Setup

A template code is provided on the GitHub system in the repository:

```
median.cpp
```

Please clone this repository to a working directory. You will find template code for this task and a project file for the Qt Creator there. The files contain the following:

- `EasyBMP` contains a helper library for reading and writing .bmp images.
- `image.h` contains an example input image as a 2D-array.
- `main.cpp` reads the input image, calls the median filter, and saves the output in a .bmp file.
- `median.cpp` implements the median filter.
- `median.h` defines the interface for the median filter.
- `median.pro` is the project file for the QT Creator tool.
- `profiler.h` implements a helper class for measuring the run time of code (used by main.cpp).

Task Description

1) Clone the `median.cpp` repository and move into it.
2) Open the QT Creator environment by executing `qtcreator median.pro &` in the task directory. You will be prompted to configure your project. Leave the settings as they are and click "Configure Project". A `median.pro.user` file will be created with your personal settings.
3) Explore the Qt Creator tool and locate the file tree separating Headers and Sources in the left column. Locate the green arrows and the hammer symbols on the lower left for run, debug, and build.

4) Open the `median.cpp` source file. You will find a `TODO` tag where you need to place your implementation of the median filter. Implement a median filter with kernel size 3x3 and neglect all possible border handling (ignore the borders and set them to 0 in the result). For a description of how median filtering works refer to https://en.wikipedia.org/wiki/Median_filter.

5) Code your implementation and execute it by pressing the green arrow "run" button. A successful run should display something like the following:

```
Profile: Programm took ~0.002753sec
Press <RETURN> to close this window...
```

An `Output.bmp` image has been created in the directory that you've specified in step 2). Open it and investigate the content.

Questions

- How are 2D-arrays passed to and received from functions?