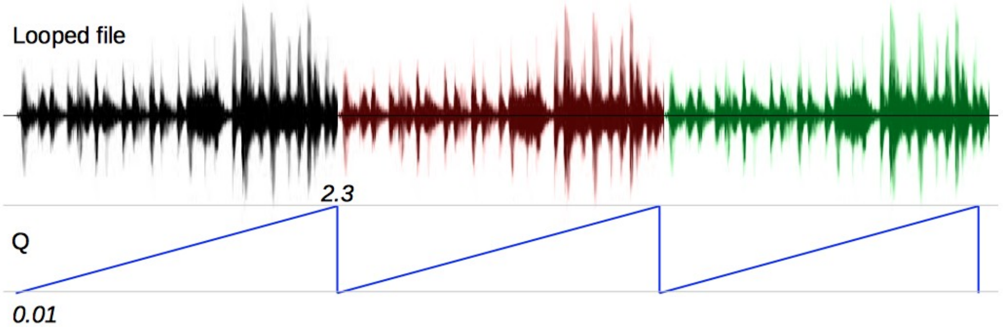
Assignment 3b

Starting from the provided template, use the same filter implemented for Assignment 3a, but make the following modifications to the application.

Input:
use as input the samples of the audio file “progression.wav” (included in the template). Put it in loop. There is no need to visualize the result on the Scope.

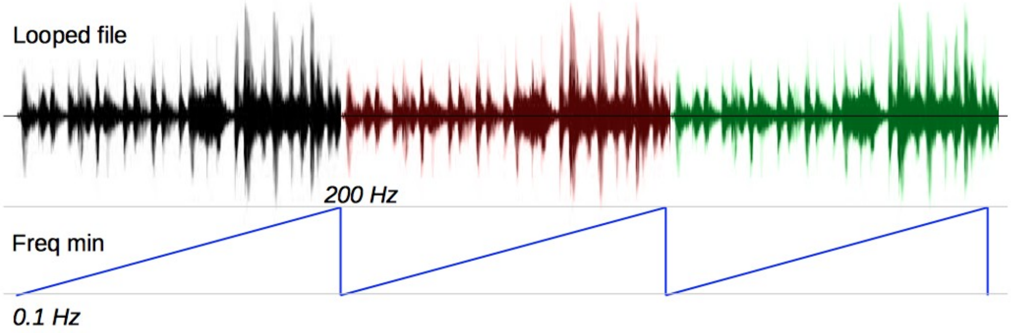
Step1 – filter modulation:
create a Low Frequency Oscillator (LFO, with sinusoidal waveshape) that modulates the cut-off frequency of the filter (f_c).
Set the modulation rate to 2 Hz (frequency of the LFO).
Make f_c oscillate between a minimum value of 200 Hz and a maximum value of 2000 Hz.
You will need to update the filter’s coefficients at each iteration within the *render()* function.

Step 2 – *Q* automation:
set *Q* to a starting value of 0.01. Then make it linearly increase throughout the duration of the audio file, so that when the file reaches its end *Q* is equal to 2.3. As the file loops back to its beginning, set *Q* back to 0.01 a let the automation start again.



The result is a ramp that repeats itself in sync with the file, which can be obtained by means of a sawtooth oscillator (already provided in the template) with a proper frequency.

Step 3 – filter automation and stereo output:
use a similar ramp to automate the frequency modulation. In particular, linearly increase the lower boundary of the modulation, so that it goes from 0.1 Hz to 200 Hz throughout the duration of the file. Keep the range of the modulation (i.e., difference between max and min freq) fixed.



Then send the output of the filter (second stage output) to the left audio out channel and the intermediate output (first stage output) to the right audio out channel.

When finished, download the render.cpp file on your laptop and rename it as “assignment3b_YOURNAME.cpp”.
Make a zip file that includes both “assignment3a_YOURNAME.cpp” and “assignment3b_YOURNAME.cpp”, call it “assignment3_YOURNAME.zip” and upload it on Blackboard.