Последовательность базовых заданий

1. Скачайте любую песню/мелодию в расширении *.mp3* (например, *song.mp3* с *dl-phnt.spbstu.ru*) и переместите в рабочую директорию. В той же директории создайте скрипт с названием *lab3_ASP_SurnameN.m*.
2. Создайте **первую секцию** с названием *Filters design*. Создайте массив частот, 2 переменные со значением порядка фильтра и частотой дискретизации:

```
freqArray = [31, 62, 125, 250, 500, 1000, 2000, 4000, 8000,16000];
     order = 1024; % должен быть четным
fS = 44100;
```

3. Создайте функцию `CreateFilters` с входными аргументами: массив частот `freqArray`, `order`, `fS`. Выходной аргумент – матрица коэффициентов фильтров `bBank`. Размер выходной матрицы: `[length(freqArray), order + 1]`. Функция `CreateFilters` должна создать *N* (исходя из длины массива `freqArray`) фильтров: 1 ФНЧ (по первой частоте), 1 ФВЧ (по последней частоте), *N-2* полосовых (остальные частоты). Для вычисления коэффициентов использовать функцию `fir2`. При создании функции необходимо использовать цикл `for` и условные операторы `if, elseif, else`. *Пример создания:*

```
freqArrayNorm = freqArray/(fS/2).
mLow = [1, 1, 0, 0];
freqLow = [0, freqArrayNorm(1), 2*freqArrayNorm(1), 1];
bLow = fir2(order, freqLow, mLow);

mBand = [0, 0, 1, 0, 0];
freqBand = [0, freqArrayNorm(1), freqArrayNorm(2), freqArrayNorm(3), 1];
bPass = fir2(order, freqBand, mBand);

mHigh = [0, 0, 1, 1];
freqHigh = [0, freqArrayNorm(end)/2, freqArrayNorm(end), 1];
bHigh = fir2(order, freqBand, mBand);
```

Примечание: необходимо установить *Signal Processing Toolbox*

4. В первой секции скрипта создайте массив фильтров, используя созданную в п.3 функцию.

5. Создайте **вторую секцию** с названием *Filtering of signals*. Прочитайте файл *song.mp3* с помощью функции `audioread` в массив `signal`.
6. Создайте вектор столбец `gain` из единиц, причем `length(gain)= length(freqArray)`
7. Создайте функцию `FilteringBanks`, которая пропускает аудиосигнал через банк фильтров, созданных в п.4. Входные аргументы: отсчеты аудио сигнала, матрица коэффициентов, полученная в результате работы функции `CreateFilters`, тип фильтрации `typeOfFilter` и массив `gain`. Выходной аргумент: `signalOut`.

   Функция должна работать с 3-мя типами фильтрации: `filter, fftfilt`. Используйте оператор `switch` для выполнения команд, соответствующих указанному во входном аргументе типу фильтрации. Типы `filter, fftfilt` соответствуют использованию встроенных функций *Matlab* `filter, fftfilt` соответственно.

   Тело функции включает в себя следующую последовательность действий:

   1) Поэлементное умножение массива `gain` и матрицы коэффициентов `bBank` и последующее сложение элементов внутри каждой строки с помощью функции `sum`
   2) В зависимости от типа фильтрации нужно выполнить:

      ```
      signalOut = filter(b, 1, signal);
      ```

      или

      ```
      signalOut = fftfilt(b, signal);
      ```
8. Пропустите аудио сигнал из п.5 через банк фильтров с помощью функции `FilteringBanks`. Вызовите функцию дважды для двух типов фильтрации.
9. Оцените время работы функции `FilteringBanks` для двух типов фильтрации `filter, fftfilt` с использованием `tic/toc`.

   Вопросы для самоконтроля. Какой тип фильтрации работает быстрее? Чем это можно объяснить?
10. Добавьте в созданную функцию `FilteringBanks` еще один тип фильтрации `trueFilter`, который соответствует выполнению подфункции `trueFilter`:

    ```
    signalOut = trueFilter(b, signal);
    ```

    Входные аргументы подфункции: отсчеты аудио сигнала `signal` и матрица коэффициентов, полученная в результате работы функции `CreateFilters, b`. Выходной аргумент: `signalOut`.

    Подфункция `trueFilter` работает по следующему алгоритму:

```
signalOut(i) = signalOut(i) + b(j)*signal(i - j);
```

Для реализации подфункции Вам понадобятся один основной цикл `for` (счетчик `i`), один вложенный цикл `for` (счетчик `j`), оператор условия `if(i - j >= 1)`

Вопросы для самоконтроля. Если функцию `trueFilter` сделать вложенной, можно ли уменьшить количество входных и выходных аргументов? Если да, то как?

11. Пропустите аудио сигнал из п.5 через банк фильтров с помощью функции `FilteringBanks`. Вызовите функцию с типом фильтрации `trueFilter`.
12. Оцените время работы функции `FilteringBanks` с типом фильтрации `trueFilter` с использованием `tic/toc`. Используйте профайлер для анализа работы скрипта (кнопка *Run and Time*).

    Вопросы для самоконтроля. Быстрее ли работает созданный самостоятельно метод фильтрации? Если да, то почему?
13. Дополните функцию FilteringBanks еще одним входным и выходными аргументами: `initB`. Внутри функции подправить следующим образом:

```
[signalOut, initB] = filter(b, 1, signal, initB);
```

14. Создайте **третью секцию** *Stream sound*. В секции создайте системные объекты для чтения и воспроизведения звуков:

```
deviceWriter = audioDeviceWriter('SampleRate', fS);
fileReader = dsp.AudioFileReader('song.mp3');
```

Используя цикл `while` и условие `isDone(fileReader)` осуществите:

- чтение порции данных `audioData = fileReader();`
- фильтрация порции данных с помощью функции `FilteringBanks`
- воспроизведение отфильтрованной части аудио сигнала с помощью `deviceWriter(audioData)`

Массив `gain` задайте перед циклом while:

```
gain = [10 10 10 0.1*ones(1, 7)]'; % bass boosted, заранее
```

Примечание: необходимо установить *DSP System Toolbox*

15. Меняйте значения массива `gain` в теле цикла, используя функцию rand:

```
gain = rand(size(freqArray))'; % в теле цикла
```

Выполните секцию *Stream sound.*

Вопросы для самоконтроля. Что изменилось по сравнению с п.14?

Дополнительные задания

1. Используя функции `nargin` и `nargout`, измените функцию `FilteringBanks` так, чтобы фильтрация с использованием `filter` была возможна как с `initB` так и без нее.
2. Перепишите функцию `trueFilter` с использование матриц, матричных операций и встроенных функций Matlab.

   Вопросы для самоконтроля. Стало ли работать быстрее?