МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Учебное пособие

Методические указания
по выполнению домашних заданий
по единому комплексному заданию по блоку дисциплины

«Основы цифровой обработки сигналов»

МГТУ имени Н.Э. Баумана

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Методические указания
по выполнению домашних заданий
по единому комплексному заданию по блоку дисциплины

«Основы цифровой обработки сигналов»

Москва
МГТУ имени Н.Э. Баумана

2012

УДК 681.3.06(075.8)
ББК 32.973-018
И201

Методические указания по выполнению домашних заданий по единому комплексному заданию по блоку дисциплины «Основы цифровой обработки сигналов» / Коллектив авторов –
М.: МГТУ им. Н.Э. Баумана, 2012. – 22 с.: ил.

В методических указаниях рассмотрены основные этапы, их последовательность и содержание по выполнению домашних заданий курсовой работы по единому комплексному заданию по блоку дисциплины «Основы цифровой обработки сигналов».

Ил. 39. Табл. 5. Библиогр. 7 назв.

УДК 681.3.06(075.8)

Аннотация

В данной работе рассмотрен процесс создания приложения «Цифровой эквалайзер». Рассмотрено устройство эквалайзера, произведён выбор КИХ-фильтров типа Equiripple. Приведены спектры сигналов до и после фильтрации.

Abstract

This paper describes how to create an application "Digital Equalizer". Considered device equalizer, made choice of FIR filters, such as Equiripple. The spectra of signals before and after filtration.

СОДЕРЖАНИЕ

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ

АЧХ	—	Амплитудно-Частотная Характеристика
КИХ	—	Фильтр с Конечной Импульсной Характеристикой
ФВЧ	—	Фильтр верхних частот
ФНЧ	—	Фильтр нижних частот

1 ОБЩЕЕ ОПИСАНИЕ

Эквала́йзер (англ. equalize — «выравнивать», общее сокращение — «EQ»), темброблок — устройство или компьютерная программа, позволяющая выравнивать амплитудно-частотную характеристику звукового сигнала, то есть корректировать его (сигнала) амплитуду избирательно, в зависимости от частоты [1].

Структурная схема работы разрабатываемого эквалайзера представлена на рисунке 1.1[4].

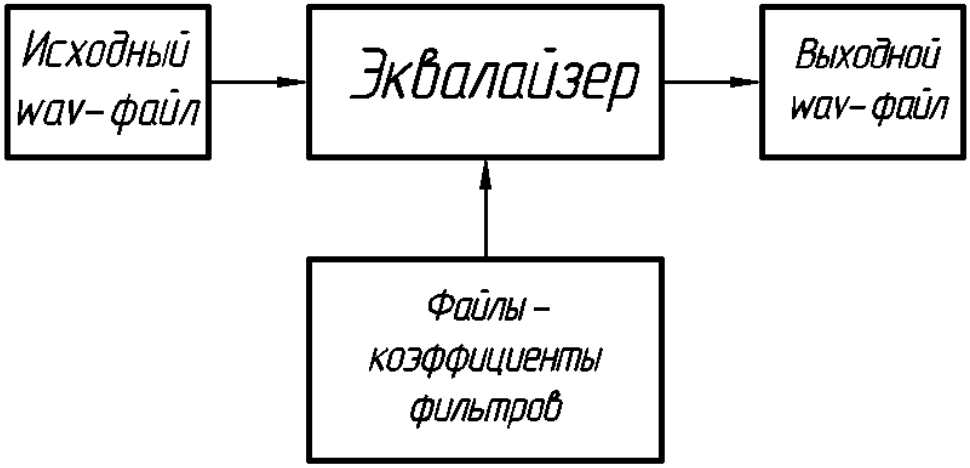

Рисунок 1.1 – Структурная схема работы эквалайзера

Входной информацией является wav-файл, закодированный при помощи импульсно-кодовой модуляции. В теле программы производится свёртка последовательности чисел-кода файла и коэффициентов фильтров. Далее, производится суммирование отфильтрованных значений и результат выводится в wav-файл.

Разрабатываемый цифровой эквалайзер имеет три фильтра – фильтр низких частот, полосовой фильтр и фильтр верхних частот. Функциональная схема эквалайзера представлена на рисунке 1.2.

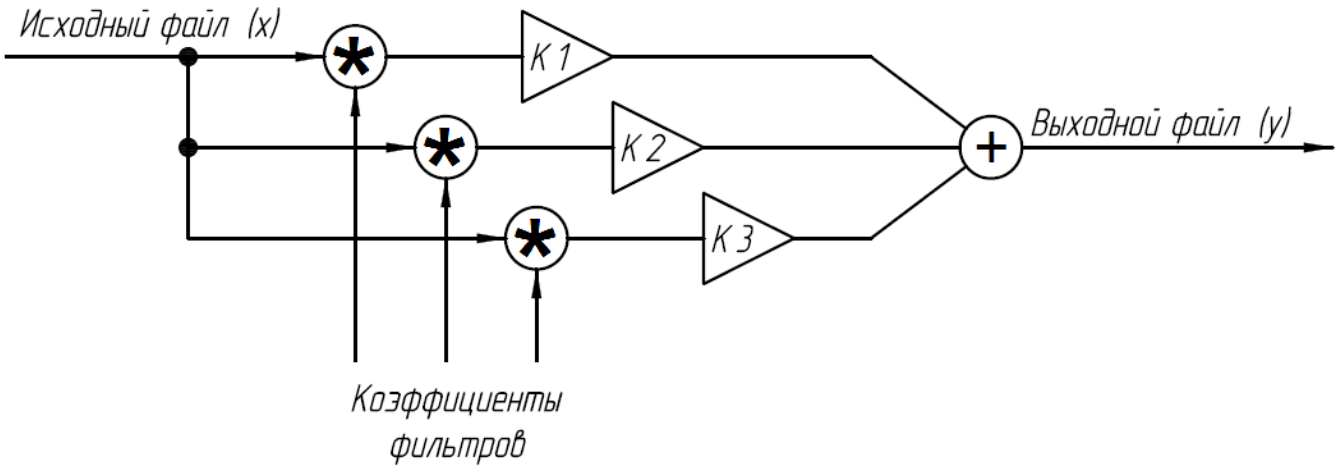

Рисунок 1.1 – Функциональная схема эквалайзера

Свёртка исходного сигнала с коэффициентами каждого из трёх фильтров происходит параллельно. Далее происходит домножение свёрнутого сигнала на коэффициент усиления данной частоты. После свёртки происходит суммирование отфильтрованных последовательностей для получения выходного файла.

2 РАЗРАБОТКА ФИЛЬТРОВ ЭКВАЛАЙЗЕРА

В разрабатываемом эквалайзере были использованы фильтры с конечной импульсной характеристикой (КИХ-фильтры). Для проектирования КИХ-фильтров был выбран оконный метод, реализованный в программном пакете MATLAB (fdatool).

Согласно рекомендациям [2] в качестве типа фильтра было выбрано Equiripple, т.к. их лекгче всего было моделировать в matlab.

В данной работе границы частот звукового спектра (слышимый звук – 10Гц..20кГц) обозначил следующим образом:

- Низкие: 10Гц…2,5кГц
- Средние: 2,5кГц…11кГц
- Высокие: свыше 11кГц

В связи с этим, разрабатываемый эквалайзер содержит 3 фильтра:

- Низких частот, пропускающий частоты до 2,5кГц
- Полосовой, пропускающий частоты от 2,5кГц до 11кГц
- Высоких частот, пропускающий частоты свыше 11кГц

Амплитудно-частотная характеристика (АЧХ) фильтра низких частот представлена на рисунке 2.1.

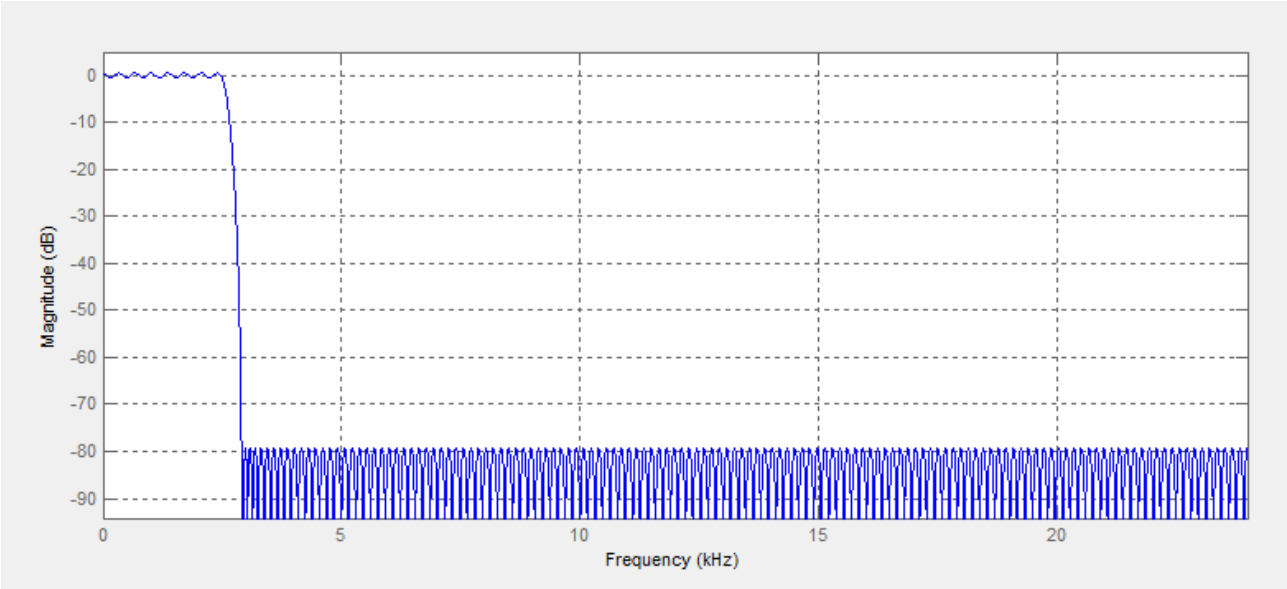

Рисунок 2.1 – АЧХ ФНЧ

Спецификации фильтра низких частот представлены в таблице 2.1.

Таблица 2.1 Спецификация фильтра низких частот

Order	304
Fs	44100
Fpass	2500
Fstop	2900

Амплитудно-частотная характеристика (АЧХ) полосового фильтра представлена на рисунке 2.2.

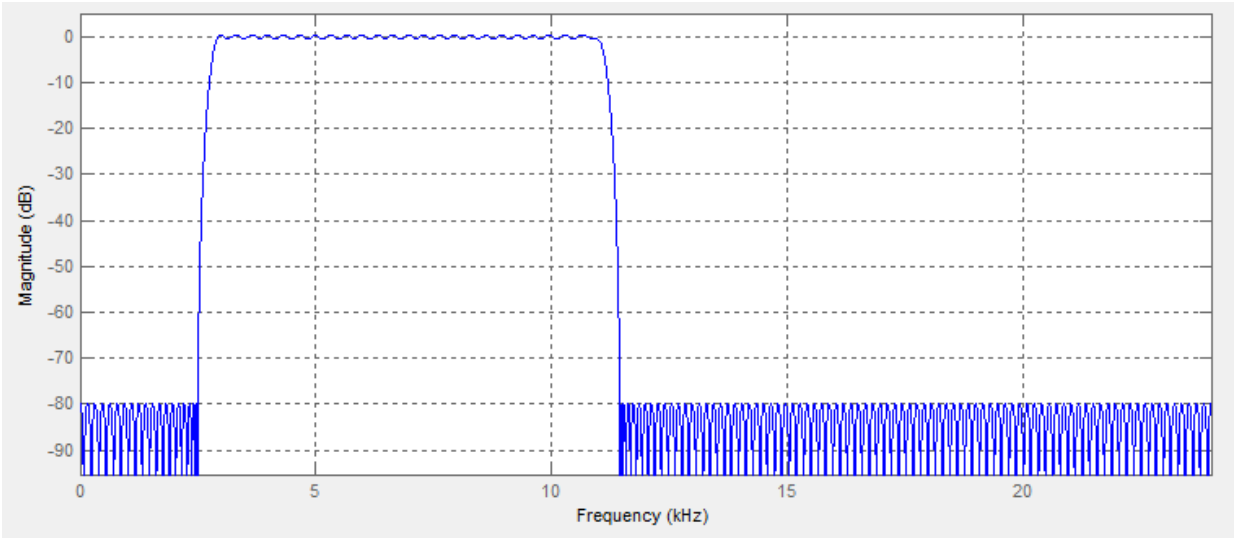

Рисунок 2.2 – АЧХ полосового фильтра

Спецификации полосового фильтра представлены в таблице 2.4.

Таблица 2.2 Спецификация полосового фильтра

Order	304
Fs	44100
Fstop1	2500
Fpass1	2900
Fpass2	11000
Fstop2	11450

Амплитудно-частотная характеристика (АЧХ) фильтра верхних частот представлена на рисунке 2.3.

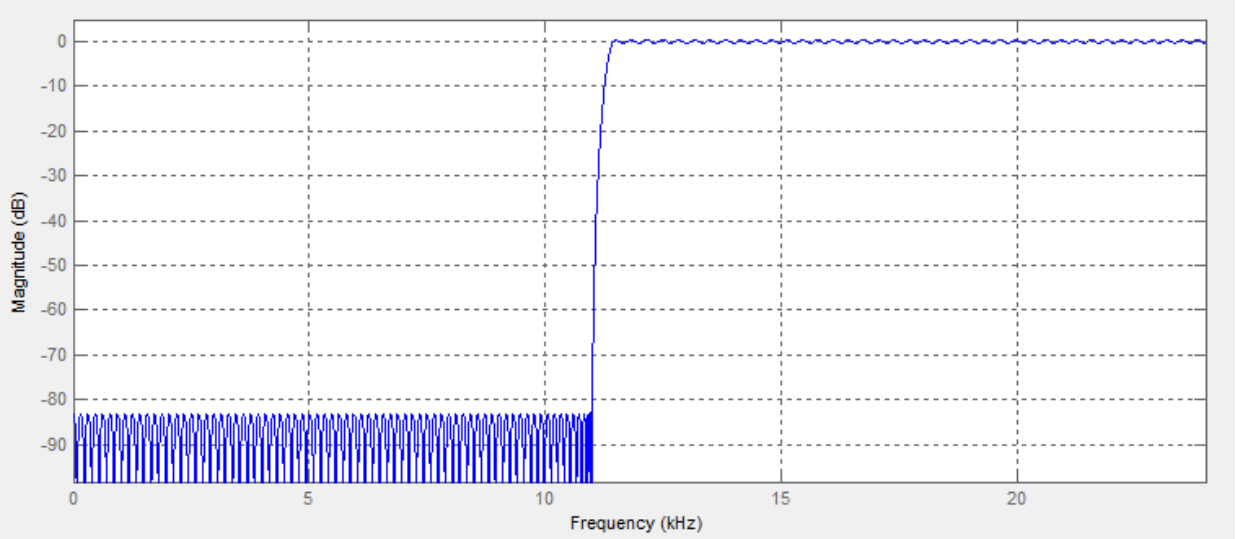

Рисунок 2.3 – АЧХ ФВЧ

Спецификации фильтра верхних частот представлены в таблице 2.3.

Таблица 2.3 Спецификация фильтра верхних частот

Order	304
Fs	200
Fs	44100
Fc	5000

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Интерфейс программы выполнен при помощи консольного режима. Внешний вид интерфейса представлен на рисунке 3.1[4].

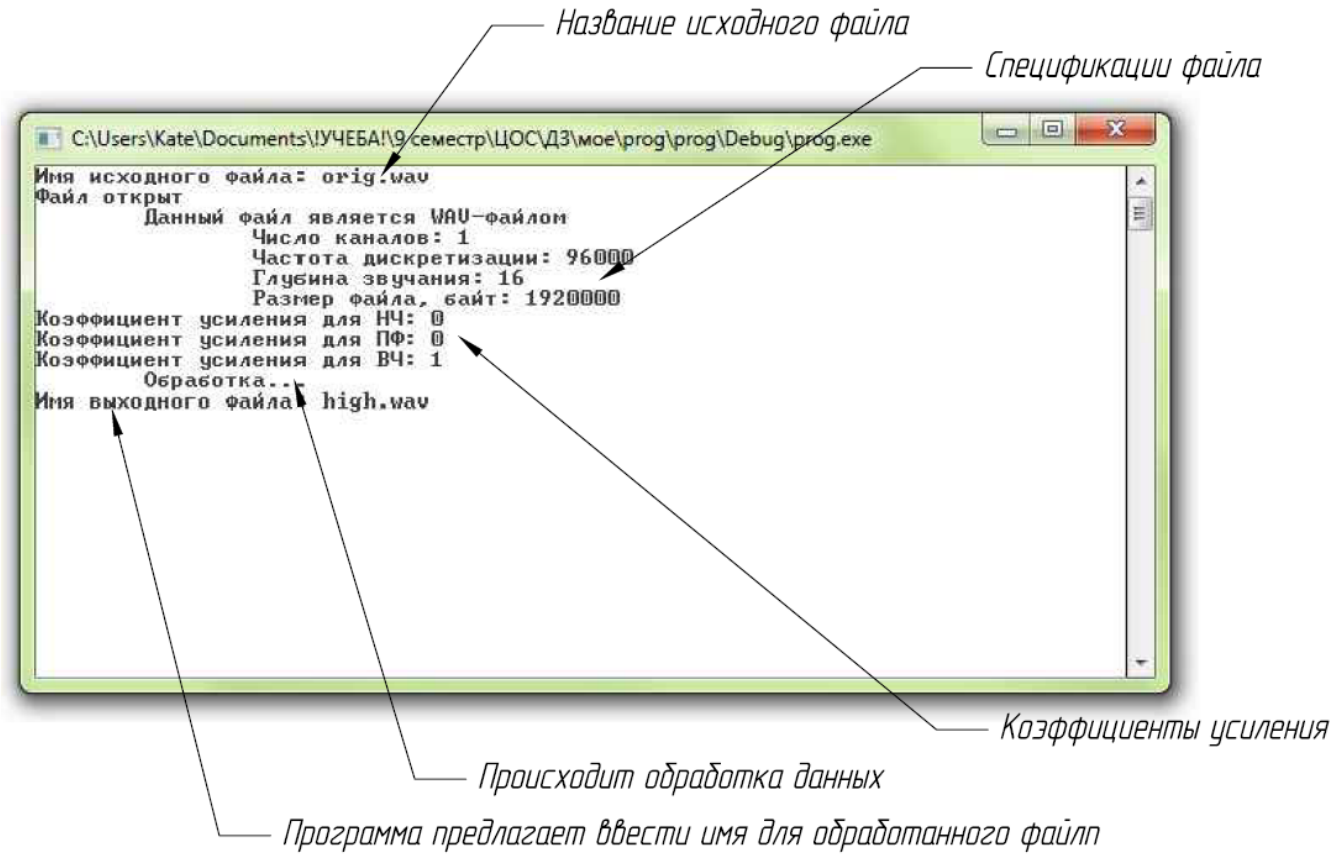

Рисунок 3.1 – Консольный интерфейс эквалайзера

4 ИССЛЕДОВАНИЕ РАБОТЫ ЭКВАЛАЙЗЕРА

Для проверки работы эквалайзера были составлены скрипты для пакета MATLAB, строящие спектры исходного wav-файла и выходных файлов, пропущенных через различные фильтры. В качестве тестового файла первоначально был выбран трек Paul van Dyk - Time Of Our Lives, но при просмотре спектра как обработанноко, так и исходного wav-файла наблюдался резкий пик, который не давал мне покоя. Поэтому по рекомендации @intoleranz и @Omega(его пока там нет) был выбран белый шум. Спектр неизменённого файла представлен на рисунке 4.1.

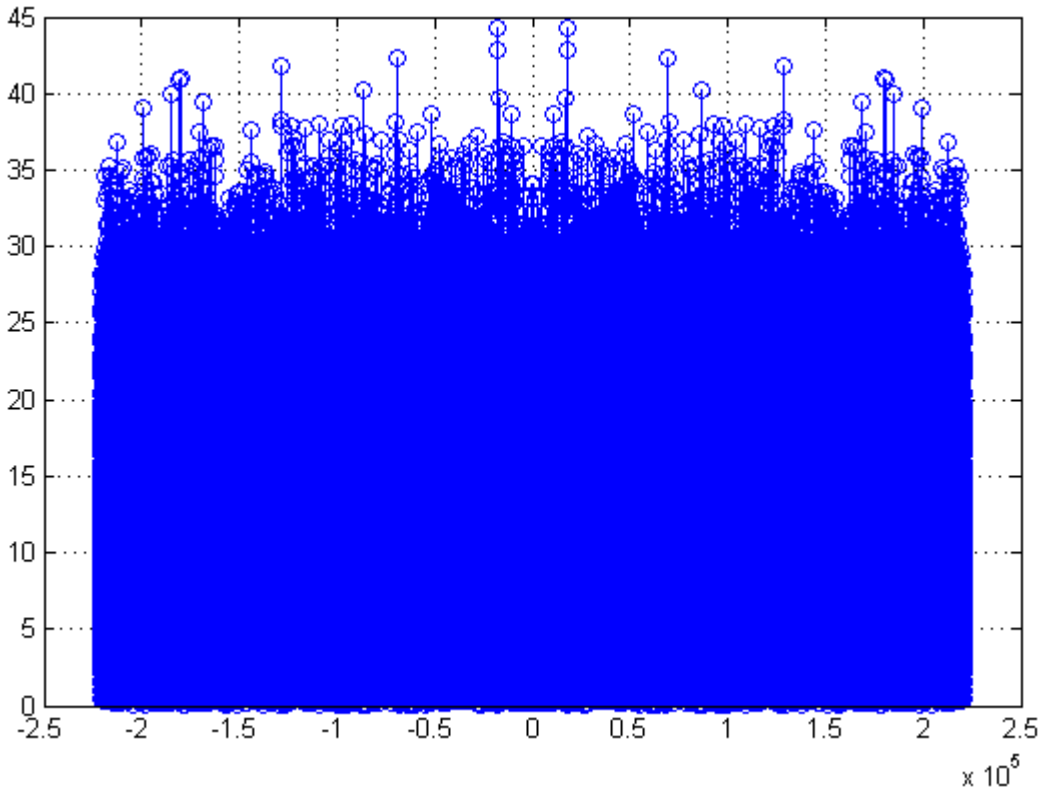

Рисунок4.1 – Спектр неизменённого файла

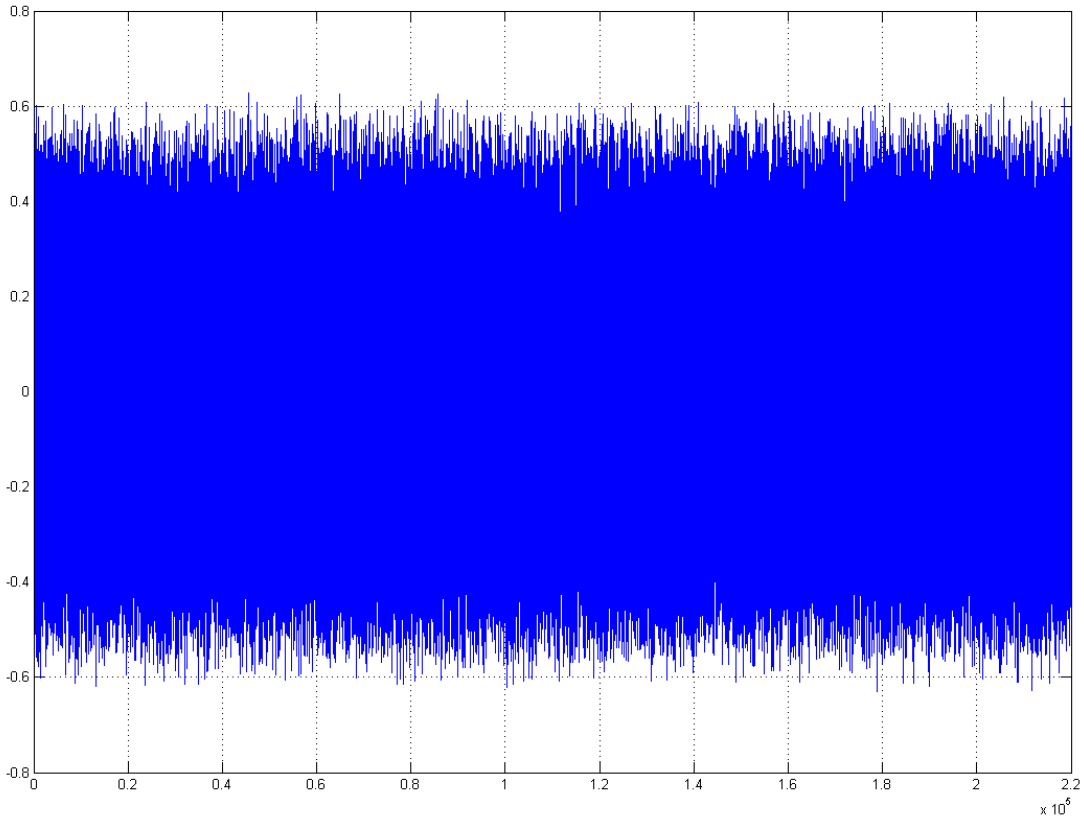

Рисунок 4.2 – Данные из wav-файла в графическом виде

Спектр файла, пропущенного через фильтр нижних частот представлен на рисунке 4.3. Данные wav-файла в графическом виде на рисунке 4.4

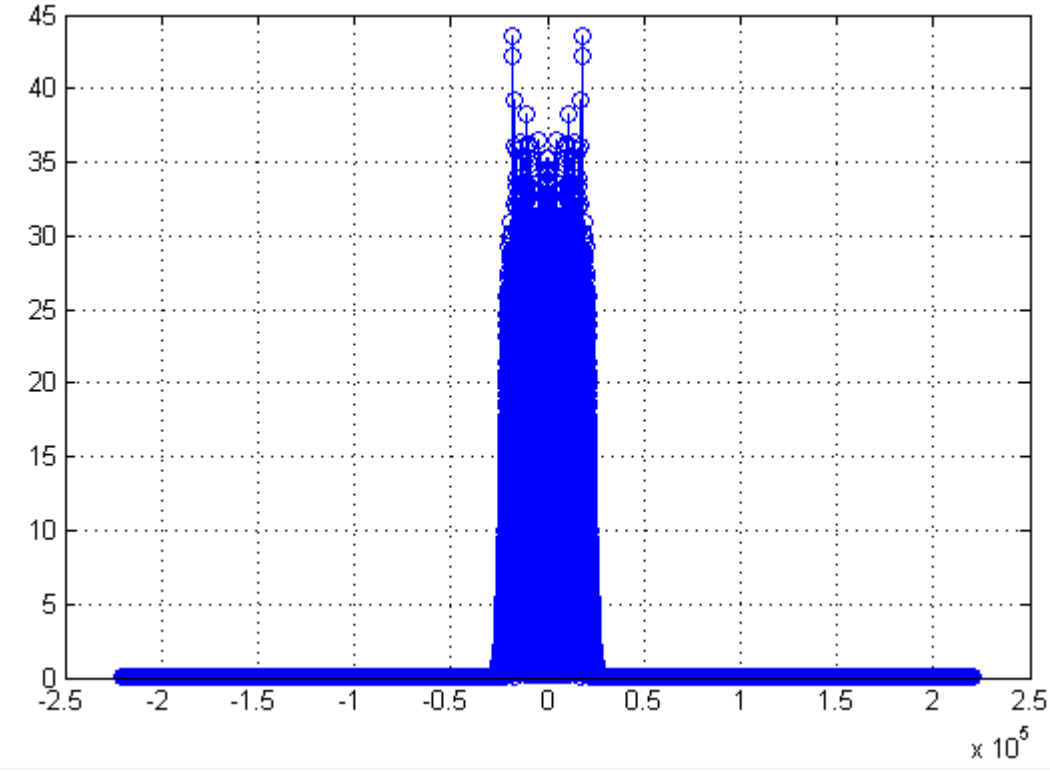

Рисунок 4.3 – Спектр файла, пропущенного через фильтр нижних частот

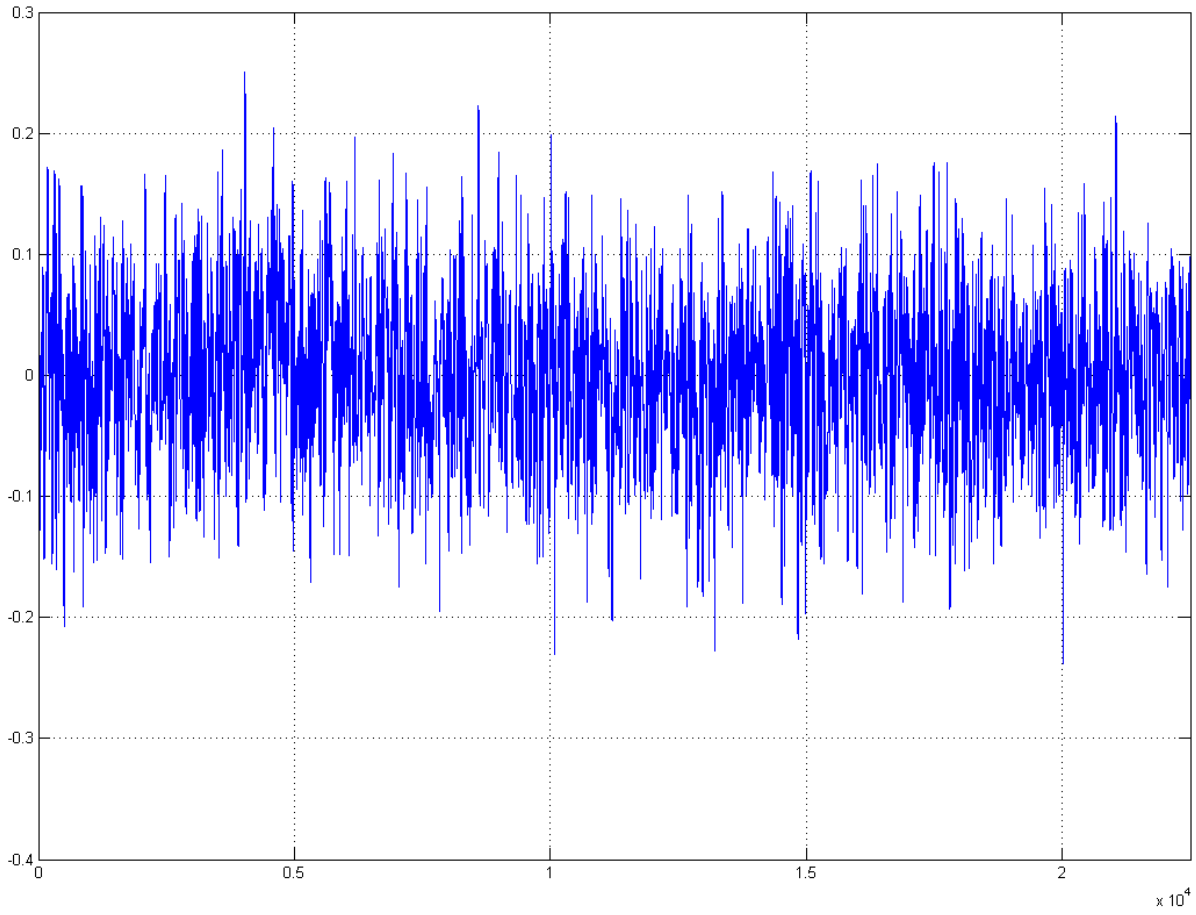

Рисунок 4.4 – Данные из wav-файла в графическом виде после ФНЧ

Спектр файла, пропущенного через полосовой фильтр представлен на рисунке 4.5.

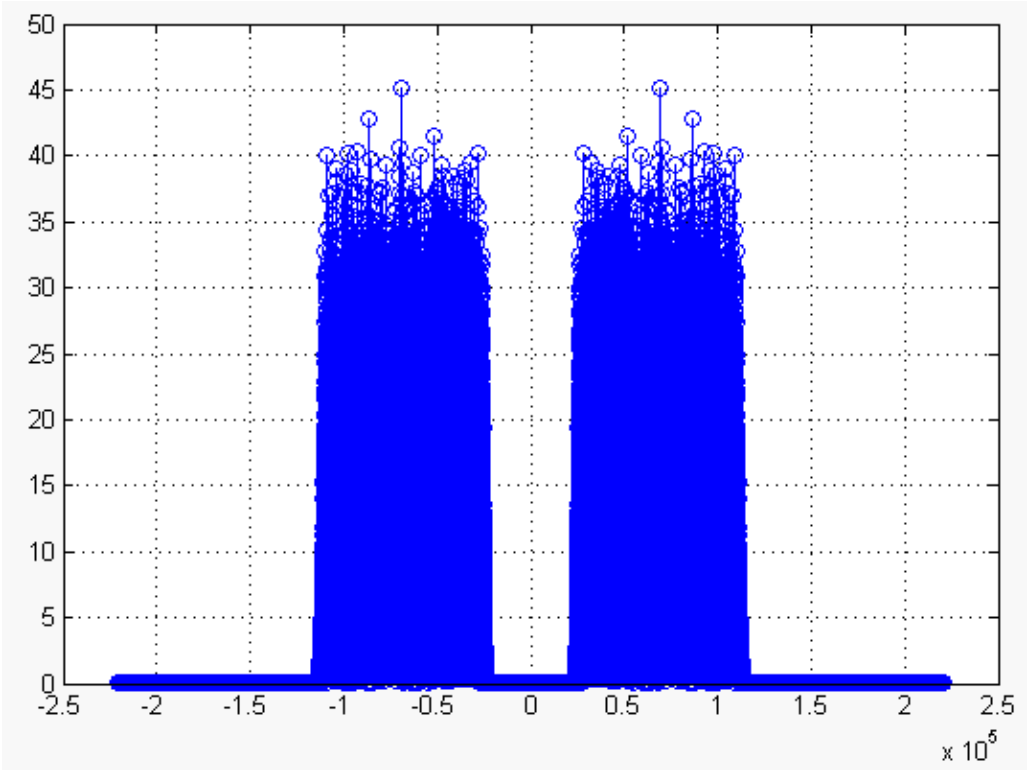

Рисунок 4.5 – Спектр файла, пропущенного через полосовой фильтр

Данные wav-файла после попущения через полосовой фильтр графическом виде на рисунке 4.6

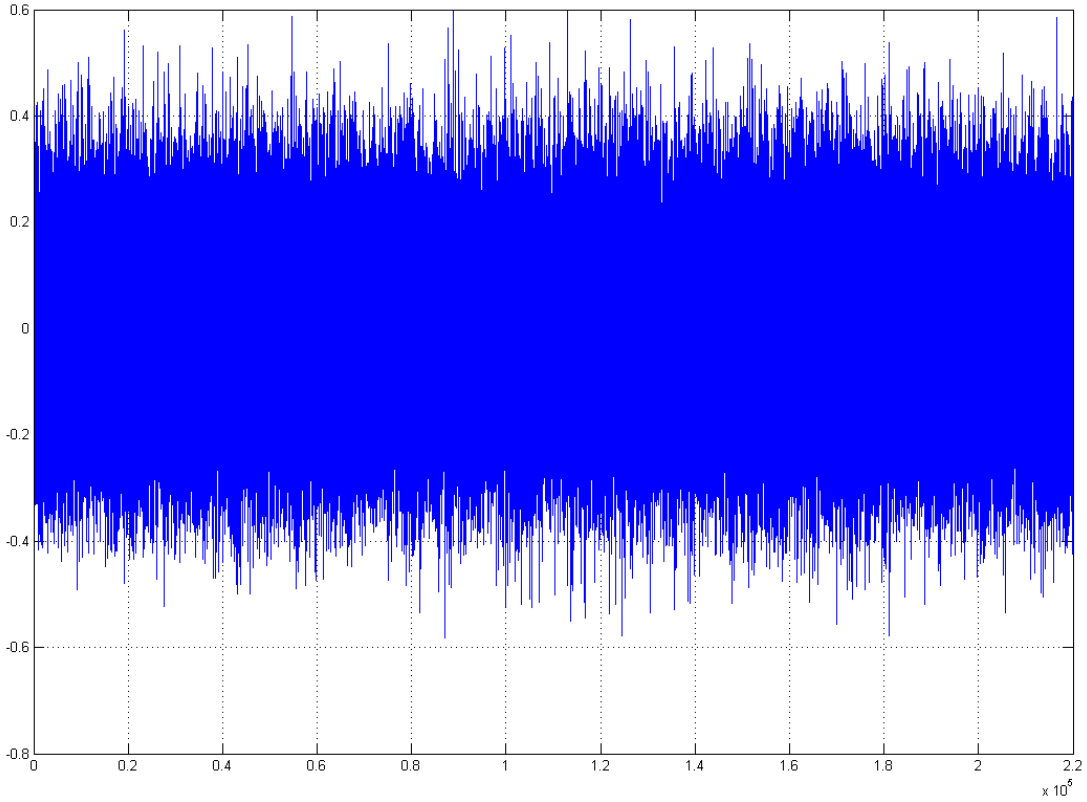

Рисунок 4.6 – Данные из wav-файла в графическом виде после полосового фильтра

Спектр файла, пропущенного через фильтр верхних частот представлен на рисунке 4.7

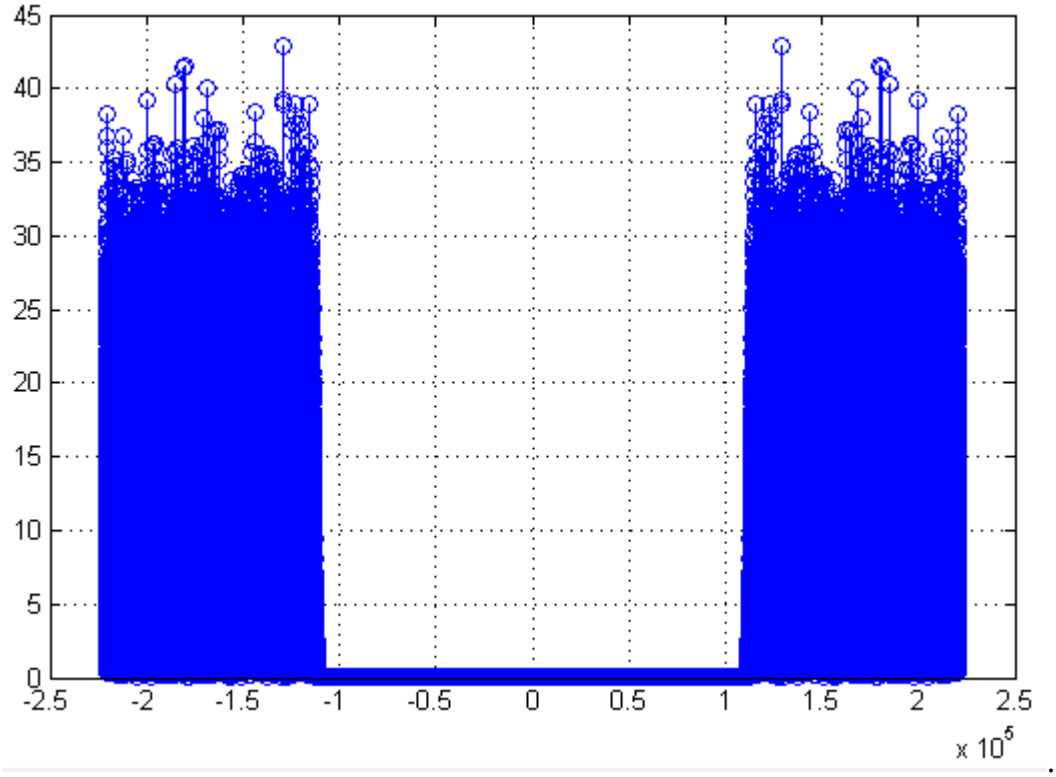

Рисунок 4.7 – Спектр файла, пропущенного через ФВЧ

На рисунке 4.8 изображено данные wav-файла после попущения через ФВЧ в графическом виде.

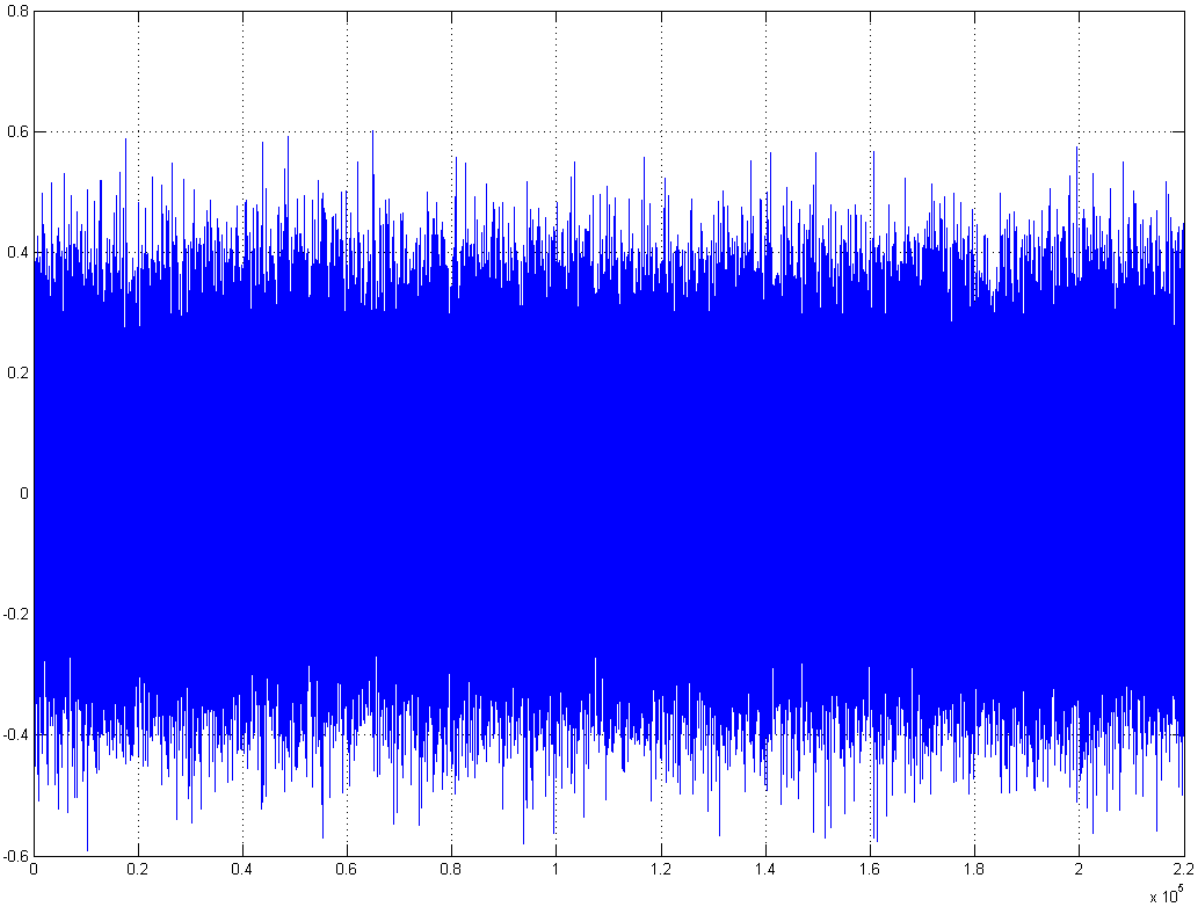

Рисунок 4.8 – Данные из wav-файла в графическом виде после ФВЧ

ЗАКЛЮЧЕНИЕ

В данном домашнем задании рассмотрен процесс создания приложения «Цифровой эквалайзер». Рассмотрено устройство эквалайзера, произведён выбор фильтров, подбор характеристик фильтров. Рассмотрен консольный интерфейс эквалайзера и приведены спектры сигналов до и после фильтрации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Эквалайзер (http://ru.wikipedia.org/wiki/Эквалайзер). [Электронный ресурс]. Проверено 17.12.2011.

2 Лайонс Р., Цифровая обработка сигналов: Второе издание. Пер. с англ. – М.: ООО «Бином-Пресс», 2006 г. – 626 с.:ил.

3 Памятка музыканту: частоты (http://fdstar.com/2008/09/01/pamyatka_muzykantu_chastoty.html). [Электронный ресурс]. Проверено 17.12.2011.

4. Маруныч К.В., Домашнее задание по Основам цифровой обработки сигналов. (пример оформления)

ПРИЛОЖЕНИЕ А

Листинг программы «Эквалайзер»

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdint.h>
#include "wavheader.h"
#include "filter_coefs.h"

#define N 301                       //Порядок фильтра

struct Item {                       //Объявляем объект, в котором будет записан отсчет сигнала
       float Lc_data;               //(отдельно для левого и правого каналов).
       float Rc_data;               //Данные объекты будут составлять динамический список,
       Item* next;                         //число элементов которого <= N
       Item* prev;
};

int main(int argc, char* argv[]) {

       if(argc != 6)                //Проверяем количество аргументов.
       {
              printf("using: %s <input_file.wav> <output_file.wav> <Low> <Middle> <High>\n", argv[0]);
              return 0;
       }

       FILE *file, *file2;                 //Открываем файлы для чтения и записи.
       file = fopen(argv[1],"rb");
       file2 = fopen(argv[2],"wb");

       if (!file)
       {
              printf("Невозможно открыть файл для чтения!\n");
          return 1;
       }

       if (!file2)
       {
              printf("Невозможно открыть файл для записи!\n");
              return 1;
       }

       int32_t data_size=0;

       short int Lc_value;
       short int Rc_value;

       float Lc_low, Lc_band, Lc_high, Lc_result;
       float Rc_low, Rc_band, Rc_high, Rc_result;

       float L_input = atof(argv[3]);
       float B_input = atof(argv[4]);
       float H_input = atof(argv[5]);

       printf("%f %f %f\n",L_input,B_input,H_input);

       float Low_coef = 0.8;
       float Band_coef = 0.8;
       float High_coef = 0.8;

       //Производим нормирование коэффициентов:
       if ((L_input>=B_input) && (L_input>=H_input)) {
              Low_coef = 0.8;
              Band_coef = 0.8*B_input/L_input;
              High_coef = 0.8*H_input/L_input;
```

```
	}

	if ((B_input>=L_input) && (B_input>=H_input)) {
		Low_coef = 0.8*L_input/B_input;
		Band_coef = 0.8;
		High_coef = 0.8*H_input/B_input;
	}

	if ((H_input>=L_input) && (H_input>=B_input)) {
		Low_coef = 0.8*L_input/H_input;
		Band_coef = 0.8*B_input/H_input;
		High_coef = 0.8;
	}

	//Объявляем и читаем заголовок wav файла:
	WAVHEADER header;
	fread(&header, sizeof(WAVHEADER), 1, file);

	//Проверяем данные wav файла:
	if (header.audioFormat != 1) {
		printf("Ошибка! На входе должен быть несжатый wav файл!\n");
		return 1;
	}

	if (header.sampleRate != 44100) {
		printf("Данная версия программы умеет работать только с данными, частота дискретизации
которых 44100Гц!\n");
		return 1;
	}

	if (header.bitsPerSample != 16) {
		printf("Данная версия программы умеет работать только с 16-битными данными!\n");
		return 1;
	}

	//Производим поиск начала массива отсчетов по ключевому слову 'data':
	char tmp_char;
	bool flag = false;

	while (!feof(file))
	{
		fread(&tmp_char, sizeof(tmp_char), 1, file);
			if (tmp_char == 'd') {
				fread(&tmp_char, sizeof(tmp_char), 1, file);
				if (tmp_char == 'a') {
					fread(&tmp_char, sizeof(tmp_char), 1, file);
					if (tmp_char == 't') {
						fread(&tmp_char, sizeof(tmp_char), 1, file);
							if (tmp_char == 'a'){
								flag = true;
								break;
							}
					}
				}
			}
	}

	//Ругаемся, если не нашли данные:
	if (!(flag)) {
		printf("Ошибка! Не найден блок данных!\n");
		return 1;
	}

	//Читаем из исходного файла число, обозначающее размер данных...
	//И пишем это число и заголовок в новый файл.
	fread(&data_size, sizeof(data_size), 1, file);
	fwrite(&header, sizeof(WAVHEADER), 1, file2);
	int32_t start_data = 0x61746164;
```

```
	fwrite(&start_data, sizeof(start_data), 1, file2);
	fwrite(&data_size, sizeof(data_size), 1, file2);

	//Инициализируем динамический список:
	Item *first = 0;	//Указатель на начало списка
	Item *last = 0;		//Указатель на конец списка
	Item *seek = 0;			//Указатель на обрабатываемый в данный момент элемент (нечто вроде курсора)

	int i=0;
	int j=0;

	printf("Идет обработка...\n");

	//Начинаем обработку данных...
	while (!feof(file))
	{

		if (i==N) {				//Если в динамическом списке N элементов, то удаляем самый первый
			first=first->next;
			delete first->prev;
			first->prev=0;
		}

		//Считываем данные отсчетов:
		fread(&Lc_value, sizeof(short int), 1, file);
		fread(&Rc_value, sizeof(short int), 1, file);

		Item *p = new Item;		//Создаем новый элемент списка.
		p->Lc_data = Lc_value;		//Присваиваем ему значения.
		p->Rc_data = Rc_value;

		if (!i) {				//Если элемент-первый в списке - присваиваем его
			first = p;			//адрес указателю first
		}
		else {
			p->prev = last;			//Иначе присваиваем указателю next предыдущего элемента
			last->next = p;			//адрес текущего.
		}
		last = p;

		if (i<N)
			i++;

		seek = last;

		Lc_low = 0;
		Lc_band = 0;
		Lc_high = 0;
		Lc_result = 0;

		Rc_low = 0;
		Rc_band = 0;
		Rc_high = 0;
		Rc_result = 0;

		//Считаем значение отсчета сигнала для каждой полосы частот:
		for (j=0; j<=i; j++) {
			Lc_low += lp_coefs[j]*seek->Lc_data;
			Rc_low += lp_coefs[j]*seek->Rc_data;

			Lc_band += bp_coefs[j]*seek->Lc_data;
			Rc_band += bp_coefs[j]*seek->Rc_data;

			Lc_high += hp_coefs[j]*seek->Lc_data;
```

```
                Rc_high += hp_coefs[j]*seek->Rc_data;

                if (seek != first)
                        seek=seek->prev;
            }

            //Складываем эти значения и применяем коэффициенты:
            Lc_result = (Low_coef*Lc_low + Band_coef*Lc_band + High_coef*Lc_high);
            Rc_result = (Low_coef*Rc_low + Band_coef*Rc_band + High_coef*Rc_high);

            Lc_value = (short int) Lc_result;
            Rc_value = (short int) Rc_result;

            //Записываем полученные данные в новый файл:
            fwrite(&Lc_value, sizeof(short int), 1, file2);
            fwrite(&Rc_value, sizeof(short int), 1, file2);

            }

            printf("Обработка завершена!\n");

            //Закрываем файлы:
        fclose(file);
        fclose(file2);
        return 0;
}
```

PS: Интерфейс, указанный выше, взят из [4]

ПРИЛОЖЕНИЕ Б

Листинг m-файла для получения спектров в Matlab

```
[yf,Fs,Nf]=wavread('имя_файла');
Af=yf(:,1); %Берем 1-ый канал
figure(1);
plot(yf(:,1)), grid %Графическое представление wav-файла
Af1=fft(Af);
Af2=fftshift(Af1);
f=-length(yf)/2:1:(length(yf)/2-1);
a=2*abs(Af2)/Nf;
figure(2);
stem(f,a), grid on %Спектры
```