Министерство образования и науки, молодежи и спорта Украины
Севастопольский национальный технический университет

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Лабораторная работа №4

«Разработка приложения для определения тонов *DTMF* и *MFC* с помощью преобразования Фурье»

по дисциплине

«**Персональные компьютеры в инженерной практике**»

для студентов дневной и заочной формы обучения

по направлению 6.050901 — Радиотехника.

Севастополь
2013 г.

УДК.519.72

Методические указания «Лабораторная работа №4 Разработка приложения для определения тонов *DTMF* и *MFC* с помощью преобразования Фурье» по дисциплине «Персональные компьютеры в инженерной практике» для студентов дневной и заочной форм обучения по направлению 6.050901 — Радиотехника / Сост. А.В. Лукьянчиков — Севастополь: Изд-во СевНТУ, 2013. — 14 с.

Методические указания рассмотрены и утверждены на заседании кафедры радиотехники и телекоммуникаций (протокол № _ от __ _____ 2013 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент: канд. техн. наук, доцент Савочкин А.А.

Ответственный за выпуск: заведующий кафедрой радиотехники и телекоммуникаций, доктор техн. наук, профессор Гимпилевич Ю.Б.,

4. ЛАБОРАТОРНАЯ РАБОТА № 4
РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ ТОНОВ *DTMF* И *MFC* С ПОМОЩЬЮ ПРЕОБРАЗОВАНИЯ ФУРЬЕ

4.1. Цель работы: Разработать приложение для анализа звуковых файлов с расширением *wav* с помощью прямого преобразования Фурье. Используя разработанное приложения проанализировать выбранный звуковой файл и определить наличие в нем тонов *DTMF* и *MFC*/

4.2. Теоретические сведения

Тональный набор, *DTMF* (англ. *Dual-Tone Multi-Frequency*) — двухтональный многочастотный аналоговый сигнал, используемый для набора телефонного номера, а также для различных интерактивных систем, например голосового автоответа. По используемой полосе частот сигнал соответствует телефонии (300…3400) Гц. Для кодирования символа в *DTMF* сигнал необходимо сложить два синусоидальных сигнала. Частоты синусоид берутся согласно таблице 4.1 из столбца и строки соответствующих передаваемому символу.

Таблица 4.1 — Алфавит кодировки *DTMF*

	1209 Гц	1336 Гц	1477 Гц	1633 Гц
697 Гц	1	2	3	*A*
770 Гц	4	5	6	*B*
852 Гц	7	8	9	*C*
941 Гц	*	0	#	*D*

Технология *DTMF* нашла применение в системе умного дома, охранных и тревожных сигнализациях. Также *DTMF*-метки широко используются в коммерческом радиовещании. Сигнал *DTMF* может быть декодирован на цифровой ЭВМ с использованием алгоритма Гёрцеля или преобразования Фурье.

Кодировка *MFC* (многочастотный челнок) используется в межстанционной сигнализации, а также для передачи номера вызываемому абоненту (АОН). Сведения о номере телефона вызывающего абонента необходимы для выписки счёта на оплату междугороднего разговора. Аппаратура АОН кроме автоматической выдачи номера вызывающего абонента позволяет осуществить выдачу номера категории, присвоенного тому или иному абоненту. Вся информация формируется комбинациями из 2-х частот на основе 6 базовых(табл. 4.2)

Таблица 4.2 — Базовые частоты кодировки *MFC*

Номер частоты	Частота, Гц
*F*0	700
*F*1	900
*F*2	1100
*F*4	1300
*F*7	1500
*F*11	1700

Таблица 4.3 — Алфавит кодировки *MFC*

Цифра	Комбинация	Цифра	Комбинация
0	*F*4,*F*7	6	*F*2,*F*4
1	*F*0,*F*1	7	*F*0,*F*7
2	*F*0,*F*2	8	*F*1,*F*7
3	*F*1,*F*2	9	*F*2,*F*7
4	*F*0,*F*4	Начало/Конец	*F*2,*F*11
5	*F*1,*F*4	Повтор	*F*4,*F*11

Длительность передачи каждой частотной комбинации 40мс. Передача номера может начаться с любой цифры но АО*Н* передает ("дублирует") информацию несколько раз (после передачи последней цифры номера затем следует первая цифра номера, вторая и т.д.; общее время выдачи всего пакета составляет обычно от 0.8 до 1.2 секунды), оканчиваться пакет может также "на любом месте", таким образом, при определении номера надо учитывать появление кода

"начало/конец". Пример: номер 1234567, категория 9, может быть выдан следующими комбинациями:
"97654321_97654321_976";
"4321_97654321_97654";
"4321_97654321_97654";
"1_97654321_97654321_97654321_9";
"4321_97654"
где "_" — код "начало/конец".
При декодировании тонов *MFC* следует учитывать что посылки передаются без интервалов между символами (в отличии от *DTMF*), поэтому одинаковые символы заменяются кодом «Повтор», например телефон 455555 передается в виде «45*p*5*p*5*p*5», где *p* — символ «Повтор». Делается это для того чтобы не было рядом стоящих одинаковых символов и можно было четко определить границы символов.

4.3. Методические рекомендации к выполнению работы

Для работы со звуковыми файлами необходимо понимать их структуру. *Wav*-файлы используют структуру *RIFF* файла, заголовок которого начинается со слова "*RIFF*". Дальнейшая информация дана в таблице 4.4.

Таблица 4.4 — Структура *Wav* файла

Название поля	Размер и формат	Содержание
Заголовок файла	4 байта (*DWord*)	*ASCII* текст "*RIFF*" Процедура, переводящая строку в *DWord*, включена в модуль *Wave*
Размер файла	4 байта (*DWord*)	Размер файла без строки "*RIFF*" (4 байта) и собственно числа размера (4 байта). Это размер_файла - 8.
WAV заголовок	4 байта (*DWord*)	*ASCII* текст "*WAVE*"
Описание формата	4 байта (*DWord*)	*ASCII* текст "*fmt* " (включая пробел)
Размер описания формата	4 байта (*DWord*)	Тип *WAVE* файла (2 байта) + *mono*/*stereo* (2 байта) + частота дискретизации (4 байта) + байт_в_секунду (4 байта) + выравнивание (2 байта) + разрядность (2 байта). В сумме 16.
Тип *WAVE* файла	2 байта (*Word*)	Заголовок *PCM* = $01 (линейная дискретизация). Другие значения подразумевают некоторые формы сжатия.
Количество каналов	2 байта (*Word*)	*mono* ($01) или *stereo* ($02). Можно попробовать больше...
Частота дискретизации	4 байта (*DWord*)	обычно 44100 *Hz*, 22050 *Hz*, ...
байт в секунду	4 байта (*DWord*)	Скорость потока данных = Число_каналов*Частота_дискретизации*Разрядность/8
Выравнивание	2 байта (*Word*)	Число байт в сэмпле = Число_каналов*Разрядность/8
Разрядность	2 байта (*Word*)	Двоичные разряды дискретизации (обычно 32, 24, 16, 8). Интересно, что будет, если установить это равным, например, 5 ..?
Заголовок данных	4 байта (*DWord*)	*ASCII* текст "*data*".
Размер данных	4 байта (*DWord*)	Число байта собственно звуковых данных
Данные	-?-	Собственно данные. Будьте внимательны при записи стерео

После слова "*data*" в файле идет собственно информация о звуке в виде байтов (или слов), записанных с периодичностью, соответствующей частоте дискретизации. Например, если частота дискретизации 10 кГц, то пауза между двумя соседними байтами, полученными с АЦП, будет 0,1 мс. Если мы имеем 16 - битный АЦП, то в файле вместо байтов будут слова (2 байта). Таким образом, информация о звуке в *wav* файле представлена в несжатом виде, ее легко использовать, но такой файл имеет большой размер. Если мы исследуем медленные процессы, то таким способом

будет записано много избыточной информации, т.к. стандартные устройства не могут иметь малую (менее 4 кГц) частоту дискретизации.

Для работы со звуковыми файлами используется готовый класс *TWave* написанный на паскале и расположенный в файле *Wave.pas* (который скачивается с методического сервера).

Поскольку класс написан на паскале его требуется подключить к *Borland C++ Builder*. Как известно, *C++Builder* вырос из *Delphi*. Большая часть того, что есть в *C++Builder*, пришла напрямую из *Delphi*. Иногда это может быть разочаровывающим, но, тем не менее, есть некоторые преимущества. Имеется большое количество доступного кода на *Delphi*, который может быть серьезным подспорьем в разработке приложений на *C++Builder*. В некоторых случаях этот код может быть использован непосредственно. В других случаях код может быть преобразован для использования в *C++Builder*. Более того, существуют много компонентов *Delphi*, для которых не существует их аналогов в *C++Builder*.

В *C++Builder* есть встроенный компилятор паскаля. Компилятор паскаля позволяет вам использовать код *Delphi* в *C++Builder*'е. Он может также помочь в конвертации кода из *Delphi* в *C++Builder*. Компилятор паскаля доступен как из *IDE C++Builder*, так и из командной строки.

Часто вы будет обнаруживать проекты *Delphi*, содержащие модуль, который бы вы хотели использовать в своих приложениях. Простейшим путем использования модуля *Delphi* является его добавление в проект. Ниже приведены шаги, необходимые для добавления модуля *Delphi* в проект *C++Builder*'а:

- Создайте в *C++Builder*'е свой проект.
- Выберите "*Add to Project*" в панели *C ++ Builder* 'а или в меню.
- Выберите "*Pascal unit*" в типах файлов выпадающего списка диалогового окна открытия файлов.
- Выберите модуль *Delphi* для добавления в свой проект и нажмите *OK*.
- Перестройте свое приложение перед написанием кода, ссылающегося на модуль *Delphi*. Перестройка проекта создаст из модуля заголовок, который вы сможете включить в свое приложение.
- Выберите пункт "*File | Include Unit Hdr…*" в главном меню *C++Builder* 'а и добавьте форму *Delphi* в ваше приложение.
- Напишите код, который ссылается на модуль *Delphi*.

Когда вы перестраиваете приложение, *C++Builder* использует встроенный компилятор паскаля для создания *obj* -файла, который приложение сможет использовать. Компилятор паскаля также создает заголовочный файл из исходного текста. Использование этого способа подключения модулей *Delphi* является простым.

Однако предпочтительным является перевод код паскаля в *C++*. Тем не менее, вы можете получить преимущество, используя компилятор паскаля *C++Builder*'а для создания заголовочного файла для этого модуля. Для этого используем компилятор из командной строки. Вот последовательность действий:

Поместить файл в папку проекта *C++Builder*'а

Открыть окно командной строки и перейти к папке, содержащей модуль *Delphi*(папка проекта *C++Builder*'а).

В командной строке наберите: "*dcc*32 *-jphn Wave.pas*" (без кавычек).

*DCC*32.*EXE* — это компилятор паскаля. Ключ *-jphn* сообщает компилятору о необходимости создать заголовочный и объектный файлы, совместимые с *C++Builder*. По завершению исполнения данной команды будет откомпилирован исходный файл на паскале и будут созданы заголовочный и объектный файлы. Далее необходимо подключить этот файл в проект (Листиниг 4.1 и 4.2).

Листиниг 4.1 — Фрагмент файла *Unit*1.*cpp* для подключения *Wave.pas*

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma link "Wave"
…..
```

Листиниг 4.2 — Фрагмент файла *Unit*1*.hpp* для подключения *Wave.pas*

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "Wave.hpp"
…..
```

Для использования преобразования Фурье необходимо звуковой файл привести к однотипным параметрам:

Частота дискретизации — 8 кГц

Количество каналов — 1

Динамический диапазон — 16 бит.

Делается это с помощью кода в Листинг 4.3

Листинг 4.3— Пример конвертации звукового файла с использоанием класса *TWave*

```
// Указываем имя файла. Название файла хранится в Edit1->Text при присваении Wave1->FileName файл считывается в память
Wave1->FileName=Edit1->Text; // Загружаем файл в память
if(Wave1->NumChannels==2 )Wave1->ConvertToMono() ; // Преобразуем количество каналов в файле в один канал
Wave1->ConvertTo16Bits() ; // Динамический диапазон — 16 бит
.// Преобразуем частоту выборки в 8кц
while(Wave1->SampleRate>8000)
Wave1->HalveSampleRate() ;
while(Wave1->SampleRate<8000)
Wave1->DoubleSampleRate() ;
…
```

Прямое преобразование Фурье осуществляется с помощью готового класса *TFFT,* распложенного в файле *FFT.CPP* (который скачивается с методического сервера). Этот файл подключается к проекту с помощью листинга 4.4.

Листинг 4.4 — Подключение файла *FFT.CPP*

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "FFT.CPP"
#include "Wave.hpp"
…
```

Непосредственно преобразование осуществляется следующей строкой

```
RealFFT(Form1->a,Form1->N,1);
```

где *Form*1*->a* — указатель на массив выборок; *Form*1*->N* — количество выборок; 1 — направление преобразования (в данном случае — прямое (получение спектра сигнала из выборок)).

После выполнения функция *RealFFT* в массив на который указывает *Form*1->*a* сохраняет действительную мнимую часть спектра сигнала (причем четные члены массива — действительные значения, а нечетные — мнимые значения). Чтобы рассчитать спектр амплитуд необходимо выполнить действие

*Form*1->*d*[*i*]=(2/*Form*1->*N*)*sqrt(*Form*1->*a*[2**i*]**Form*1->*a*[2**i*]+
*Form*1->*a*[2**i*+1]**Form*1->*a*[2**i*+1]);.

То есть вычислить корень из суммы квадратов действительной и мнимой части спектра. Далее необходимо вывести спектр амплитуд на форму приложения. Для отображения графической информации в *C++Builder* используется компонент *Chart*. Компонент *Chart* позволяет строить различные диаграммы и графики.

Является контейнером объектов *Series* типа *TChartSeries* – серий данных, характеризующихся различными стилями отображения. Каждая серия будет соответствовать одной кривой на графике. Характерные свойства компонента представлены в таблице 4.5.

Таблица 4.5 — Свойства компонента *Chart*

Свойство	Описание
AllowPanning	Определяет возможность пользователя прокручивать наблюдаемую часть графика во время выполнения, нажимая правую кнопку мыши: *pmNone* – прокрутка запрещена; *pmHorizontal* – разрешена прокрутка только в горизонтальном направлении; *pmVertical* – только в вертикальном *pmBoth* – в обоих напрвлениях
AllowZoom	Позволяет пользователю изменять во время выполнения масштаб изображения, вырезая фрагменты диаграммы или графика курсором мыши.
Title	Определяет заголовок диаграммы
Foot	Определяет подпись под диаграммой. По умолчанию отсутствует. Текст подписи определяется подсвойством *Text*
Frame	Определяет рамку вокруг диаграммы
Legend	Легенда диаграммы – список обозначений
MarginLeft, *MarginRight* *MarginTop* *MarginButtom*	Значения левого, правого, верхнего и нижнего полей
ButtomAxis *LeftAxis* *RightAxis*	Эти свойства определяют характеристики соответственно нижней, левой и правой осей. Задание этих свойств имеет смысл для графиков и некоторых других типов диаграмм
LeftWall *ButtomWall* *BackWall*	Эти свойства определяют характеристики соответственно левой, нижней и задней граней области трехмерного отображения графика
SeriesList	Список серий данных, отображаемых в компоненте
View3d	Разрешает или запрещает трехмерное отображение диаграммы
View3dOptions	Характеристики трехмерного отображения
Chart3DPersent	Масштаб трехмерности (толщина диаграммы, ширина лент графика)

Редактор диаграмм вызывается:

- кнопкой с многоточием рядом с названием свойства в инспекторе объектов;
- двойным щелчком на компоненте *Chart* при проектировании формы;
- выбором команды *Edit Chart* в контекстном меню компонента *Chart* при проектировании формы.

Для задания отображаемых значений надо использовать методы серий *Series*:

- *Clear* – очищает серию от занесенных ранее данных.
- *Add* – позволяет добавить в диаграмму новую точку:

Add(*Const Double AValue*; *Const AnsiString ALabel*; *TColor AColor*)
Параметр *AValue* соответствует добавляемому значению, параметр*ALabel* – название, которое буде отображаться на диаграмме и в легенде, параметр *AColor* – цвет. Параметр *ALabel* необязательный, его можно задавать пустым.

- *AddXY* – позволяет добавить новую точку в график функции:

AddXY(*Const Double AXValue, AYValue*; *Const AnsiString ALabel*; *TColor AColor*);
Параметры *AXValue* и*AYValue* соответствуют аргументу и функции, параметры *ALabel* и *AColor* – те же, что и в методе *Add*.

Для настройки компонента *Chart* необходимо:

Перейти в редактор диаграмм *Chart*1.

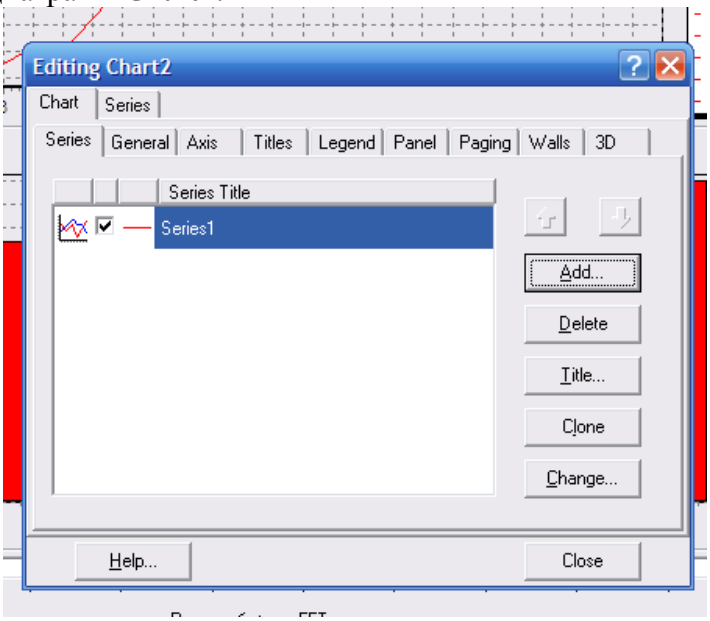


Рисунок 4.1 — Окно редактора диаграмм *Chart*

На закладке *Chart*, на закладке *Series* щелкнуть на кнопке *Add* – добавить серию.

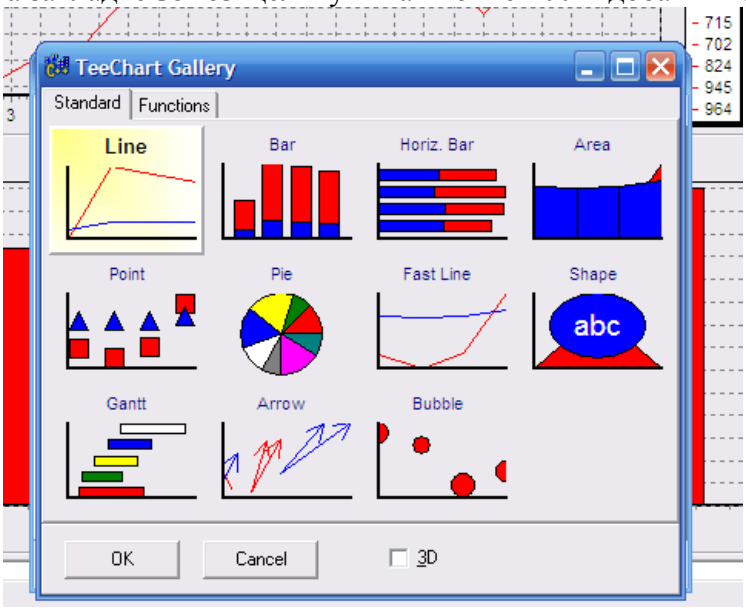


Рисунок 4.2 — Окно добавления новой серии в диаграмму *Chart*

Вы попадаете в окно, в котором можно выбрать тип диаграммы или графика. В данном случае выберем *Line* – кусочно ломана аппроксимация.

Закладка *Titles* – позволяет задавать заголовок диаграммы (Диаграмма продукции подразделений)

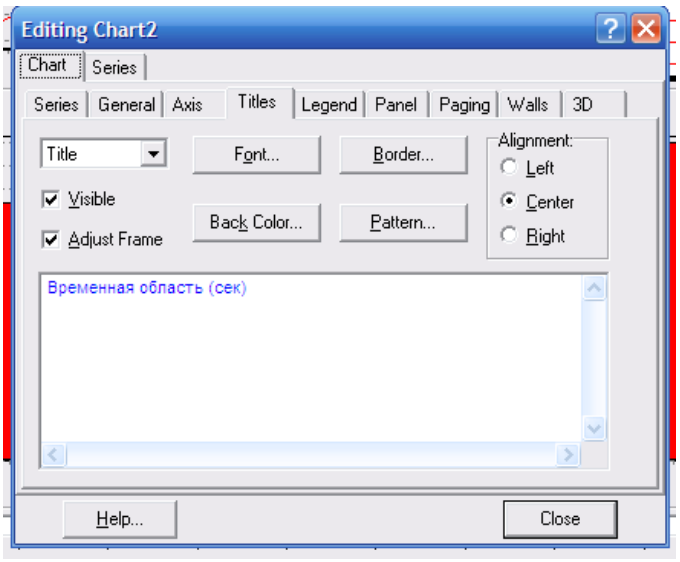


Рисунок 4.3 — Окно настройки заголовка диаграмм *Chart*

Закладка *Legend* – позволяет задавать позволяет задавать параметры отображения легенды легенды диаграммы (списка обозначений) или вообще убирать ее с экрана.

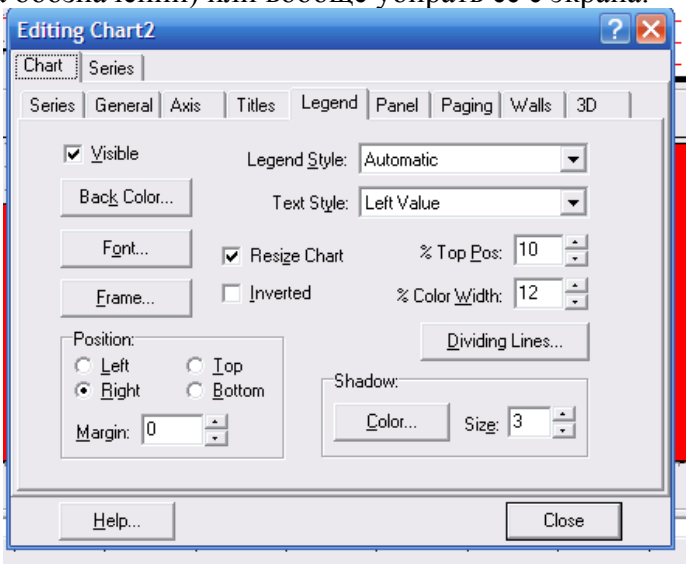


Рисунок 4.4 — Окно настройки параметров отображения легенды диаграммы *Chart*

Закладка *Panel* – определяет вид панели, на которой отображается диаграмма.

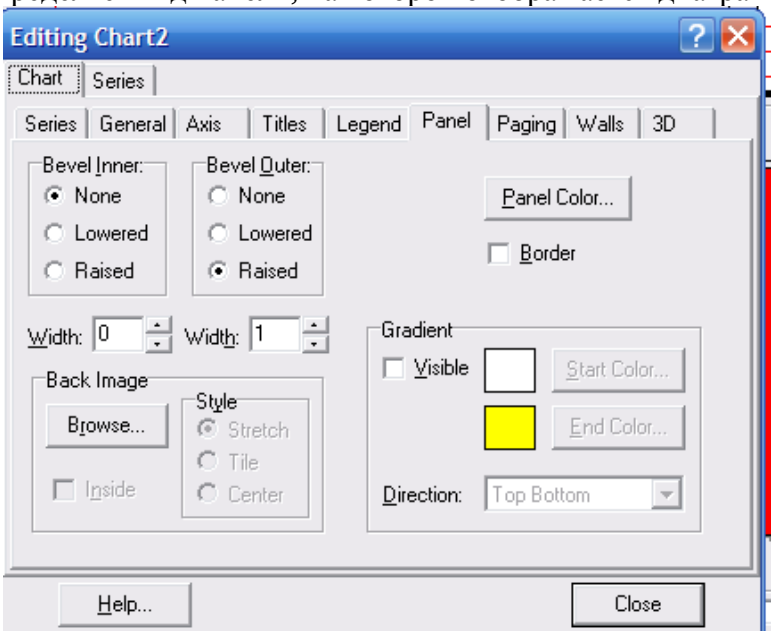


Рисунок 4.5 — Окно настройки отображения вида панели диаграммы *Chart*

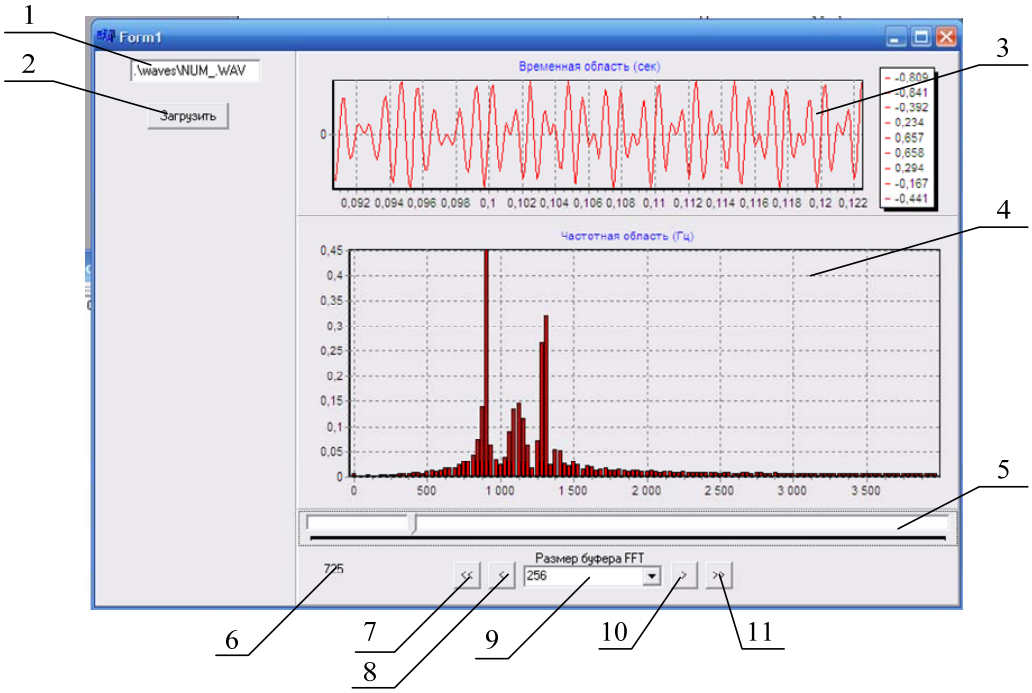
Закладка 3*D* – позволяет определить внешний вид диаграммы: сдвиг, наклон, толщину и т.д.

Результат применения различных опций сразу отображается на условном примере.

В редакторе диаграмм *Chart*1, на закладке *Series* можно выбрать дополнительные характеристики отображения серии. На закладке *Format* для круговой диаграммы можно включить опцию *Circled Pie*, которая обеспечит при любом размере компонента *Chart* отображение диаграммы в виде круга. На закладке *Marks* кнопки группы *Style* определяют, что будет написано на ярлычках, относящихся к отдельным сегментам диаграммы: *Value* – значение, *Percent* – проценты, *Label* – названия данных и т.д. В приведенном примере включена кнопка *Percent*, а на закладке *General* установлен шаблон процентов, обеспечивающий отображение значений с точностью два десятичных знака. Есть возможность отображать одни и те же данные с помощью разных типов диаграмм. Для этого в редакторе диаграмм *Chart*1, на закладке *Chart*, нажав на закладке *Series* кнопку *Clone*, а затем для этой новой серии нажать кнопку *Change* (изменить) и выбрать другой тип диаграммы, например, *Bar*. Чтобы эти два разных типа диаграммы не появлялись на одном рисунке одновременно, нужно выключить индикатор этой новой серии на закладке *Series*, а потом предосватить пользователю выбрать тот или иной вид отображения диаграммы, например с помоью событи *OnClick* для диаграммы.

Операторы *Clear* нужны, если в процессе работы приложения нужно обновлять данные. без этих операторов повторное выполнение методов *Add* и *AddXY* только добавит новые точки, не удалив прежние.

Рекомендованное окно программы изображено на рисунке 4.6



1 — компонент *Edit*1 для ввода имени файла;
2 — компонент *Button*5 для загрузки файла;
3 — компонент *Chart*2 для вывода выборок файла, количество которых задается в компоненте 9;
4 — компонент *Chart*1 для вывода спектрограммы амплитуд на основе выборок файла, количество которых задается в компоненте 9;
5 — компонент *TrackBar*1 для точной навигации по файлу;
6 — компонент *Label*1 указывается номер первой выборки ;
7, 11 — компонент *Button*2, *Button*4 для грубой навигации по файлу с шагом указанным в компоненте 9;
8, 10 — компонент *Button*1, *Button*3 для грубой навигации по файлу с шагом указанным в компоненте 9 и деленным на два;
9 — компонент *ComboBox*1 для задания количества выборок, которое может принимать фиксированные значения: 8, 16, 32, 64, 128, 256, 512, 1024, 2048,

Рисунок 4.6 — Окно программы для анализа звуковых файлов

Листинг 4.6 — Функция расчета и вывода спектрограммы амплитуд на график

```
//---------------------------------------------------------------------------
void drawFFT(int position)// аргумент функции указывает на номер первой выборки в файле
{
int j;
double maxw;
 int i;
 int jj=position;
Form1->Label2->Caption=IntToStr(jj); // Вывести номер первой выборки
//Определение максимальной выборки в файле
maxw=fabs(Form1->Wave1->GetValue(0,0));
for(j=0;j<(int)Form1->Wave1->Size;j++ )
if(maxw<fabs(Form1->Wave1->GetValue(j,0)))maxw=fabs(Form1->Wave1-
>GetValue(j,(TSelectedChannel)0));

Form1->Chart2->Series[0]->Clear(); // Очистка графика выборок
// Вывод выборок на график Chart2
for(i=0;i<Form1->N;i++)
{
Form1->a[i]=Form1->Wave1->GetValue(jj+i,(TSelectedChannel)0)/maxw;
Form1->Chart2->Series[0]->AddXY((jj+i)/Form1->freq,Form1->a[i]);
}

Form1->Chart1->Series[0]->Clear(); // Очистка графика спектрограммы амплитуд
RealFFT(Form1->a,Form1->N,1); // Выполнение прямого преобразования Фурье результат
находится в массиве a
// Вывод спектрограммы амплитуд в Chart1
for(i=0;i<Form1->N/2;i++)
{  double x;
Form1->d[i]=(2/Form1->N)*sqrt(Form1->a[2*i]*Form1->a[2*i]+Form1->a[2*i+1]*Form1-
>a[2*i+1]);
x=Form1->freq*i/Form1->N;
Form1->Chart1->Series[0]->AddXY(x,Form1->d[i]);
}
// Выход из функции
}
```

При нажатии на элемент 2 (см. рис. 4.6) должна выполнятся следующая последовательность действий:

Листинг 4.7 — Обработчик нажатия кнопки «Загрузить»

```
N= ComboBox1->Text.ToIntDef(96); // задания количества выборок согласно элемнта 9 (см.
рис. 4.6)
 Panel2->Enabled=true; // Разрешить использовать панель навигации
//------Преобразование wav файла в одноканальный, 16 битный, с частотой выборки 8 кГц

Wave1->FileName=Edit1->Text;
if(Wave1->NumChannels==2 )Wave1->ConvertToMono() ;
Wave1->ConvertTo16Bits() ;
while(Wave1->SampleRate>8000)
Wave1->HalveSampleRate() ;
while(Wave1->SampleRate<8000)
```

```
Wave1->DoubleSampleRate() ;


freq=8000 ; // Установить переменную для корректного отображения
TrackBar1->Max=(int)Wave1->Size-N; // Максимальное значение в TrackBar1 равно количеству выборок в файле минус количество выборок для расчета
TrackBar1->Min=0; // Минимальное равно нулю
drawFFT(0); // Рисуем спектр используя функцию из листинга 4.6.
```

Для учета изменения количества анализируемых выборок, необходимо добавить обработчик для элемента *ComboBox*1 (элемент 9 см. рис. 4.6) реагирующий на изменение значения *ComboBox*1.

Листинг 4.8 — Обработчик *ComboBox*1*Change*

```
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
N= ComboBox1->Text.ToIntDef(96); // изменить кол-во выборок
TrackBar1->Max=(int)Wave1->Size-N; // изменить максимальное значение в TrackBar1 равно количеству выборок в файле минус количество выборок для расчета
int jj= TrackBar1->Position; //считать текущую позицию TrackBar1
drawFFT(jj); // Рисуем спектр используя функцию из листинга 4.6. начиная с текущей позиции

}
```

Чтобы спектр изменялся при перемещении ползунка *TrackBar*1 или при нажатии кнопок 7,8,10,11 (см. рис. 4.6) необходимо добавить соответствующие обработчики событий которые будут вызвать обработчик *ComboBox*1*Change* так как он является универсальным

Листинг 4.9 — Обработчики *TrackBar*1*Change, Button*2*Click, Button*4*Click, Button*1*Click,* :*Button*3*Click*

```
//---------------------------------------------------------------------------

void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
 ComboBox1Change(Sender);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
if( TrackBar1->Position-N>0) TrackBar1->Position-=N;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
if( TrackBar1->Position+N<Wave1->Size-N) TrackBar1->Position+=N;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
if( TrackBar1->Position-N/4>0) TrackBar1->Position-=N/4;
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
 if( TrackBar1->Position+N/4<Wave1->Size-N/4) TrackBar1->Position+=N/4;
}
//---------------------------------------------------------------------------
```

При анализе звуковых файлов следует особое внимание уделать размеру выборки так как он не должен превышать длительность одного символа (*MFC* или *DTMF*) с одной стороны, а с другой не должен быть слишком маленьким чтобы не возрастало время анализа файла.

4.4. Индивидуальные задания

Согласно варианту полученному у преподавателя выбрать звуковые файлы для анализа из табл. 4.6.

Таблица 4.6 — Индивидуальное задание

Вариант	Имя файла
0	*Palixa_t*54321_1.*wav*
1	5542451_*pk*54321_1.*wav*
2	*Palixa_t*54321_3.*wav*
3	*NUM.WAV*
4	3212.*WAV*
5	*NUM*2.*WAV*
6	*NUM_.WAV*
7	*MFC.WAV*
8	*CLEAR*1.*WAV*
9	9300_*p*54321.*wav*180605_094943.*w*

После написания программы необходимо провести анализ заданных файлов заполнить таблицу 4.7.

Таблица 4.7 — Детектированные *DTMF* и *MFC* символы

№	Время мсек	*DTMF* символ	Время мсек	*MFC* символ
1				
2				
…				

4.5. Содержание отчета

Отчет должен содержать:

1. Титульный лист;
2. Цель работы;
3. Описание и алгоритм работы программы;
4. Текст программы и результаты разработки программы;
5. Результаты работы программы (таблица 4.7);
6. Выводы.

Перед защитой отчета студент должен продемонстрировать работу программы.

4.6. Контрольные вопросы

4.6.1. Как установить режимы рисования на канве? Какие режимы есть у пера?
4.6.2. Как очистить поверхность графического компонента Image?
4.6.3. Как установить цвет и ширину линии, которой рисуется график функции?

4.6.4. Назовите основные методы для рисования класса Canvas.

4.6.5. Назовите основные события графических компонентов Image и PaintBox.

4.6.6. Как вывести текст в графические компоненты?

4.6.7. Как вывести на графическом экране:

а) точку в позицию, указанную курсором мыши;

б) свою фотографию, загруженную из графического файла;

в) установить толщину линии, ее цвет и провести линию указанной длины и под заданным углом.

4.6.8. Как установить цвет рисунка при использовании графики и нарисовать указанным цветом две касающиеся внутренним образом окружности?

4.6.9. Опишите методику анимации изображения.

4.6.10. Какой режим пера Pen используется для удаления рисунка при повторном его рисовании?

4.6.11. Какие свойства имеет компонент Timer?

4.6.12. Назовите основные события мыши.

4.6.13. Как в программе распознать координаты курсора мыши?

4.6.14. Как распознать нажатую кнопку мыши?

4.6.15. Когда наступают события мыши OnStartDrag, OnDragOver, OnDragDrop, OnEndDrag? Что можно распознать при обработке этих событий?

4.6.16. С помощью какой клавиши клавиатуры можно перемещать фокус с элемента на элемент?

4.6.17. Какие способы вывода графической информации Вы знаете?