# **Выполнение индивидуального задания**

# **(описание задания и хода его выполнения)**

## **Получение задания, сбор и описание требований к разработке ПО по заданию предприятия.**

*Формулировка задания:*

Создать отечественный музыкальный плеер. В связи с уходом с рынка множества зарубежных партнёров и их программ, было выдано задание создать отечественный музыкальный плеер для личного пользования сотрудников. Основной целью является написать удобный аудио-проигрыватель. Так же в цель входило научить программу всем стандартным функциям аудио плееров таким как: запуск с указанием стартового аудио файла или списка, открытие и добавление в список папок и фалов. Наименование программного средства – Музыкальный плеер “Zain”. Программа предназначена для воспроизведения аудио файлов различных форматов.

Программа предоставляет графический интерфейс для удобного управления музыкой.

**Выбор средств разработки**

*Описание среды разработки Microsoft Visual Studio*

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

В комплект входят следующие основные компоненты:

1. Visual Basic.NET — для разработки приложений на Visual Basic;

2. Visual C++ — на традиционном языке C++;

3. Visual C# — на языке C# (Microsoft);

4. Visual F# — на F# (Microsoft Developer Division).

Функциональная структура среды включает в себя:

редактор исходного кода, который включает множество дополнительных функций, как авто дополнение IntelliSense, рефракторинг кода и т. д.;

редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

дизайнер схем баз данных.

Visual Studio также позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (Subversion и Visual Source Safe), добавление новых наборов инструментов (для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения).

Коммерческие версии в порядке возрастания цены: Visual Studio Professional, Visual Studio Premium и Visual Studio Ultimate.

*Достоинства и недостатки*

Интегрированная среда разработки (IntegratedDevelopmentEnvironment — IDE) Visual Studio предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio.

* Встроенный Web-сервер. Для обслуживания Web-приложения ASP.NET необходим Web-сервер, который будет ожидать Web-запросы и обрабатывать соответствующие страницы. Наличие в Visual Studio интегрированного Web-сервера позволяет запускать Web-сайт прямо из среды проектирования, а также повышает безопасность, исключая вероятность получения доступа к тестовомуWeb-сайту с какого-нибудь внешнего компьютера, поскольку тестовый сервер может принимать соединения только с локального компьютера.
* Поддержка множества языков при разработке. Visual Studio позволяет писать код на своем языке или любых других предпочитаемых языках, используя все время один и тот же интерфейс (IDE). Более того, Visual Studio также еще позволяет создавать Web-страницы на разных языках, но помещать их все в одно и то же Web-приложение. Единственным ограничением является то, что в каждой Web-странице можно использовать только какой-то один язык (очевидно, что в противном случае проблем при компиляции было бы просто не избежать).
* Меньше кода для написания. Для создания большинства приложений требуется приличное количество стандартного стереотипного кода, и Web-страницы ASP. NET тому не исключение. Например, добавление Web-элемента управления, присоединение обработчиков событий и корректировка форматирования требует установки в разметке страницы ряда деталей. В Visual Studio такие детали устанавливаются автоматически.
* Интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам. Применяемые Visual Studio автоматически параметры форматирования можно даже настраивать, что очень удобно в случаях, когда разработчик предпочитает другой стиль размещения скобок (например, стиль K&R, при котором открывающая скобка размещается на той же строке, что и объявление, которому она предшествует).
* Более высокая скорость разработки. Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense (которая умеет перехватывать ошибки и предлагать правильные варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно.
* Возможности отладки. Предлагаемые в Visual Studio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

Visual Studio также имеет и множество других функций: возможность управления проектом; встроенная функция управления исходным кодом; возможность рефакторизации кода; мощная модель расширяемости. Более того, в случае использования Visual Studio 2008 Team System разработчик получает расширенные возможности для модульного тестирования, совместной работы и управления версиями кода (что значительно больше того, что предлагается в более простых инструментах вроде Visual SourceSafe).

В качестве недостатка можно отметить невозможность отладчика (Microsoft Visual Studio Debugger) отслеживать в коде режима ядра. Отладка в Windows в режиме ядра в общем случае выполняется при использовании WinDbg, KD или SoftICE.

**Требования к музыкальному плееру**

*Определение круга потенциальных пользователей:*

Пользователями «МКУ "Защита и безопасность"» являются сотрудники.

*Для сотрудников программный продукт должен:*

-Проигрывать, останавливать, выбирать музыкальные треки

-Предоставлять информацию о песне, что играет

-Изменять громкость и скорость воспроизведения песен

-Сохранять, удалять избранные треки

-Иметь удобный интуитивно понятный интерфейс для взаимодействия пользователя и системы

*Требования к функциональным характеристикам*

Было принято решение о написании плеера на языке C# в связи с множеством преимуществ данного языка.

Воспроизведение аудио файлов. Воспроизведение файлов в формате \*.mp3, \*.aac, \*.wav, \*.mid, \*.ogg. Возможность поиска по Play листу. Динамический поиск (поиск осуществляется, начиная с набора первого символа, после набора второго символа поиск продолжается с более жесткими условиями и т. д. до нахождения определенно файла или уведомления о том, что файла, соответствующего таким условиям поиска, не существует). Настройка громкости воспроизведения. Воспроизведение всех песен по кругу “Зацикленное воспроизведение”.

*Графический интерфейс*

Возможность открытия файлов через графическое меню. Возможность управления воспроизведением через графические инструменты управления (клавиши «Play», «Pause», «Stop» и т. д.).

*Требования к надежности*

Надёжность программного средства должна соответствовать следующим требованиям:

- защита от ошибочных действий пользователей на уровне пользовательского приложения;

- обеспечение обнаружения и диагностики ошибок с выдачей соответствующих сообщений пользователю на уровне пользовательского приложения.

*Требования к аппаратному обеспечению*

Требования к аппаратному обеспечению отражены в требованиях для работы в «Visual Studio 2019». Необходимы компьютеры с характеристиками не хуже:

* Процессор с тактовой частотой не ниже 1,8 ГГц. Рекомендуется использовать как минимум двухъядерный процессор.
* 2 ГБ ОЗУ; рекомендуется 8 ГБ ОЗУ (минимум 2,5 ГБ при выполнении на виртуальной машине)
* Место на жестком диске: до 1 ГБ (минимум 50 МБ)
* Скорость жесткого диска: для повышения производительности установите Zain (Музыкальный плеер) на твердотельный накопитель (SSD)
* Видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей); для оптимальной работы рекомендуется разрешение WXGA (1366 на 768 пикселей) или более высокое.

**Разработка музыкального плеера**

*Разработка интерфейса*

Основной задачей при разработке интерфейса является:

* Максимальное использование пустого пространства на форме.
* Максимально интуитивное управление.
* Минимальное количество дополнительных окон.

После запуска программы открывается основное окно программы с пустой формой (Приложение А, рисунок 1). После этого был изменён цвет формы, её размеры и некоторые свойства приводя форму к нынешнему виду.

(Приложение А, рисунок). Затем были добавлены списки для композиций, поисковые строки, PictureBox для отображения обложки песни и конечно же, сам плеер, воспроизводящий композиции. (Приложение А, рисунок 3, 4).

Добавлены кнопки для взаимодействия с проигрываемыми треками. (Приложение А, рисунок 5). Добавлена временная шкала для отслеживания и контроля хода воспроизведения композиции. (Приложение А, рисунок 6). И в конце были добавлены “ползунок” для изменения громкости композиции и меню выбора темпа. (Приложение А, рисунок 7)

Конечный вид дизайна приложения (Приложение А, рисунок 8).

*Разработка функционала*

Сначала были запрограммированы кнопки Play, Pause, Next, Previous выполнять функции остановки, возобновления проигрывания композиции, а также переключения между композициями в списке. (Приложение Б, рисунок 9).

Добавлена возможность открытия музыкальных композиций из файловой системы устройства, на котором установлен плеер. (Приложение Б, рисунок 10).

Позже была запрограммирована возможность поиска композиций по названию. (Приложение Б, рисунок 11). В связи с необходимостью в перемотке и настройке звука при прослушивании композиций, соответствующие функции были добавлены и не только. (Приложение Б, рисунки 12 и 13).

Также для удобства пользователей была добавлена система <<Избранных песен>>, включающая сохранение понравившихся песен и удаление ненужных. Все <<Избранные песни>> хранятся в папке внутри приложения. (Приложение В, рисунок 14,15).

## **Приложения**

Приложение А

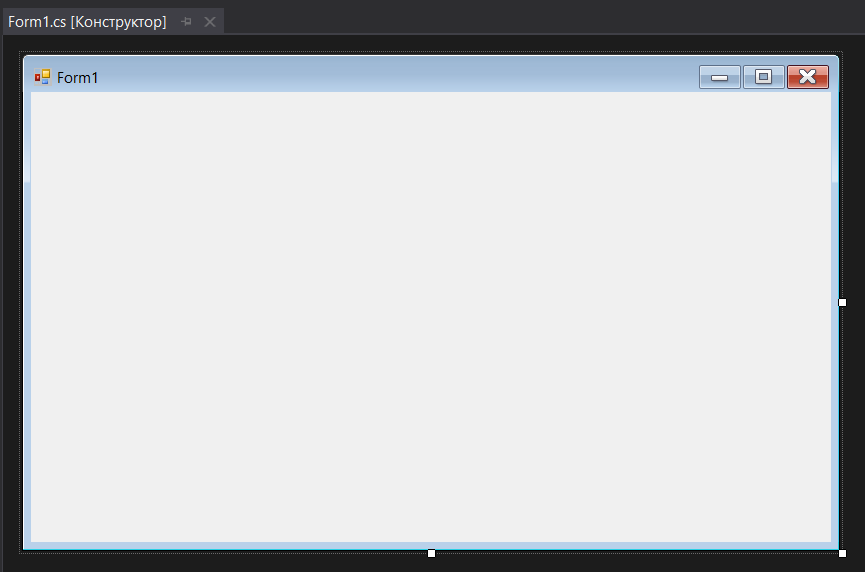


Рисунок 1 – Пустая форма

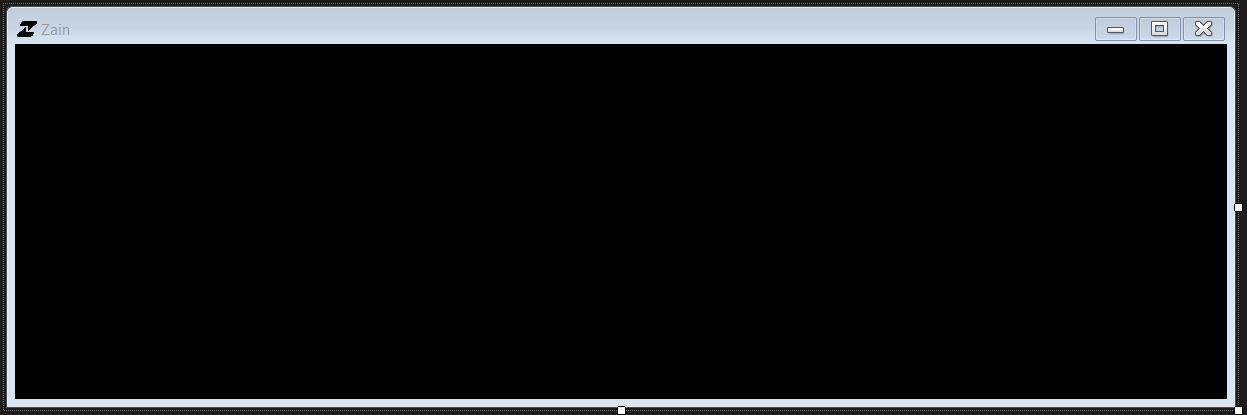


Рисунок 2 – Форма после редактирования

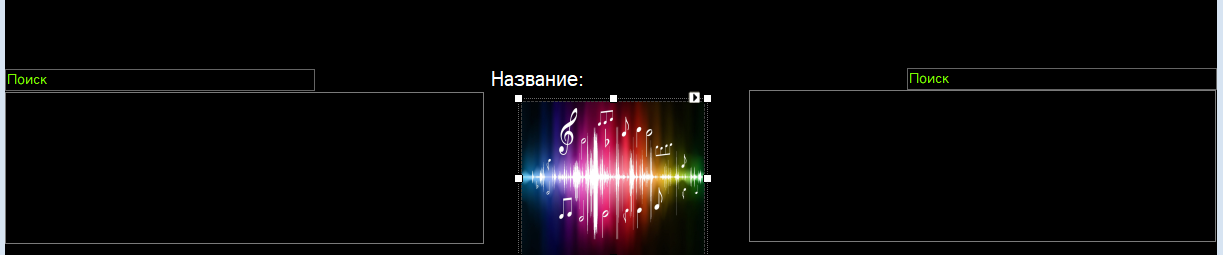


Рисунок 3 – Списки, поисковые строки, плеер



Рисунок 4 – VU-метр



Рисунок 5 - Кнопки



Рисунок 6 – Шкала времени



Рисунок 7 –“Ползунок” громкости и меню темпа воспроизведения



Рисунок 8 – Дизайн приложения

Приложение Б

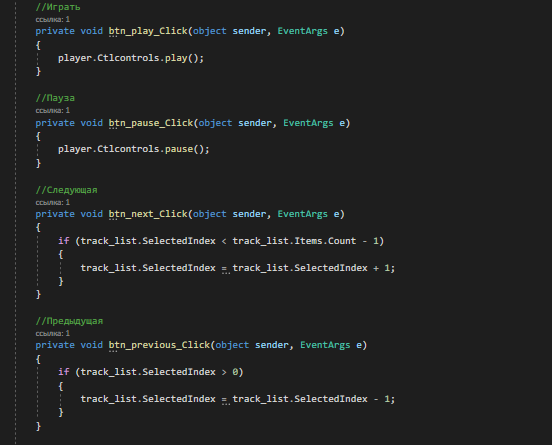


Рисунок 9 – Play, Pause, Next, Previous

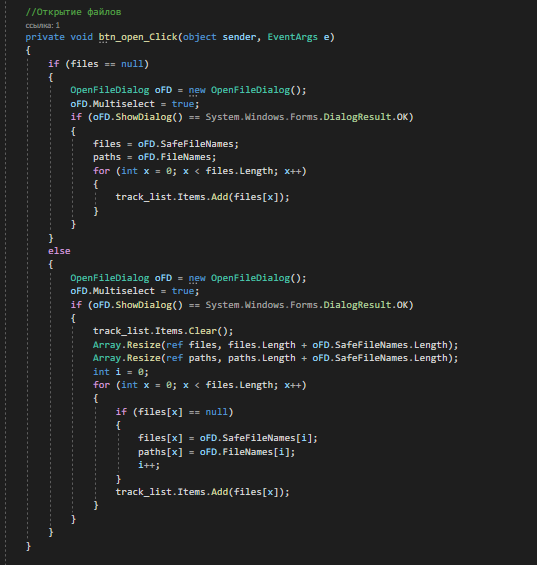


Рисунок 10 – Открытие файлов



Рисунок 11 – Поиск

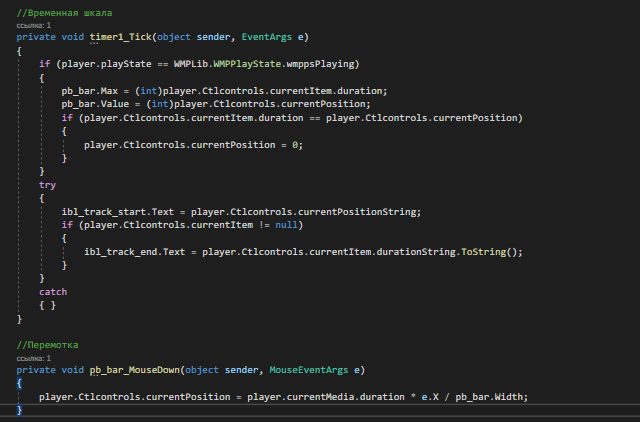


Рисунок 12 – Контроль времени композиции

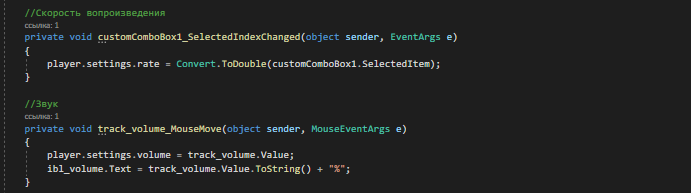


Рисунок 13 – Звук и скорость воспроизведения

Приложение В

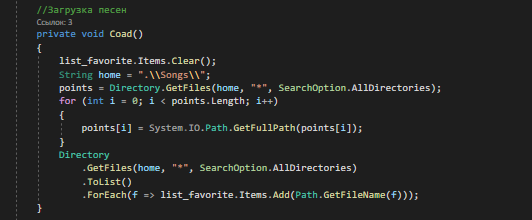


Рисунок 14 – Обновление списка

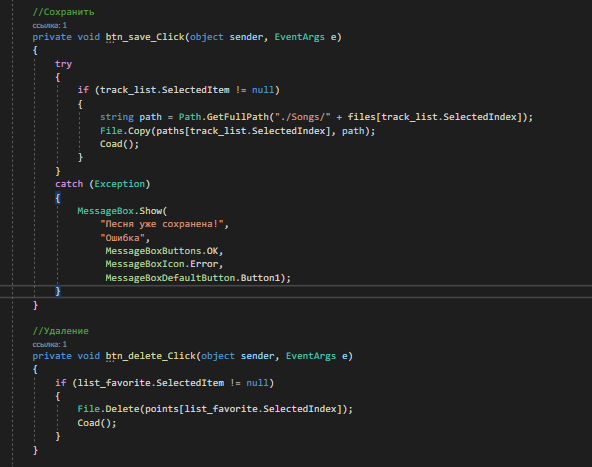


Рисунок 15 – Сохранение и Удаление

Приложение Г

Функциональный код программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace AudioPlayer

{

public partial class Form1 : Form

{

//

public Form1()

{

InitializeComponent();

track\_volume.Value = 50;

ibl\_volume.Text = "50%";

Coad();

}

//Загрузка песен

private void Coad()

{

list\_favorite.Items.Clear();

String home = ".\\Songs\\";

points = Directory.GetFiles(home, "\*", SearchOption.AllDirectories);

for (int i = 0; i < points.Length; i++)

{

points[i] = System.IO.Path.GetFullPath(points[i]);

}

Directory

.GetFiles(home, "\*", SearchOption.AllDirectories)

.ToList()

.ForEach(f => list\_favorite.Items.Add(Path.GetFileName(f)));

}

//Переменные

String[] paths, files, points;

Color b\_color;

Color h\_color = Color.Crimson;

//Открытие файлов

private void btn\_open\_Click(object sender, EventArgs e)

{

if (files == null)

{

OpenFileDialog oFD = new OpenFileDialog();

oFD.Multiselect = true;

if (oFD.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

files = oFD.SafeFileNames;

paths = oFD.FileNames;

for (int x = 0; x < files.Length; x++)

{

track\_list.Items.Add(files[x]);

}

}

}

else

{

OpenFileDialog oFD = new OpenFileDialog();

oFD.Multiselect = true;

if (oFD.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

track\_list.Items.Clear();

Array.Resize(ref files, files.Length + oFD.SafeFileNames.Length);

Array.Resize(ref paths, paths.Length + oFD.SafeFileNames.Length);

int i = 0;

for (int x = 0; x < files.Length; x++)

{

if (files[x] == null)

{

files[x] = oFD.SafeFileNames[i];

paths[x] = oFD.FileNames[i];

i++;

}

track\_list.Items.Add(files[x]);

}

}

}

}

//Играть

private void btn\_play\_Click(object sender, EventArgs e)

{

player.Ctlcontrols.play();

}

//Пауза

private void btn\_pause\_Click(object sender, EventArgs e)

{

player.Ctlcontrols.pause();

}

//Следующая

private void btn\_next\_Click(object sender, EventArgs e)

{

if (track\_list.SelectedIndex < track\_list.Items.Count - 1)

{

track\_list.SelectedIndex = track\_list.SelectedIndex + 1;

}

}

//Предыдущая

private void btn\_previous\_Click(object sender, EventArgs e)

{

if (track\_list.SelectedIndex > 0)

{

track\_list.SelectedIndex = track\_list.SelectedIndex - 1;

}

}

//Временная шкала

private void timer1\_Tick(object sender, EventArgs e)

{

if (player.playState == WMPLib.WMPPlayState.wmppsPlaying)

{

pb\_bar.Max = (int)player.Ctlcontrols.currentItem.duration;

pb\_bar.Value = (int)player.Ctlcontrols.currentPosition;

if (player.Ctlcontrols.currentItem.duration == player.Ctlcontrols.currentPosition)

{

player.Ctlcontrols.currentPosition = 0;

}

}

try

{

ibl\_track\_start.Text = player.Ctlcontrols.currentPositionString;

if (player.Ctlcontrols.currentItem != null)

{

ibl\_track\_end.Text = player.Ctlcontrols.currentItem.durationString.ToString();

}

}

catch

{ }

}

//Перемотка

private void pb\_bar\_MouseDown(object sender, MouseEventArgs e)

{

player.Ctlcontrols.currentPosition = player.currentMedia.duration \* e.X / pb\_bar.Width;

}

//Скорость вопроизведения

private void customComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

player.settings.rate = Convert.ToDouble(customComboBox1.SelectedItem);

}

//Звук

private void track\_volume\_MouseMove(object sender, MouseEventArgs e)

{

player.settings.volume = track\_volume.Value;

ibl\_volume.Text = track\_volume.Value.ToString() + "%";

}

//Дизайн списков

private void track\_list\_DrawItem(object sender, DrawItemEventArgs e)

{

b\_color = e.BackColor;

Color clr = Color.FromArgb(0, b\_color);

if (e.Index >= 0)

{

SolidBrush sb = new SolidBrush(((e.State & DrawItemState.Selected) == DrawItemState.Selected) ? h\_color : b\_color);

e.Graphics.FillRectangle(sb, e.Bounds);

string txt = track\_list.Items[e.Index].ToString();

SolidBrush tb = new SolidBrush(e.ForeColor);

e.Graphics.DrawString(txt, e.Font, tb, track\_list.GetItemRectangle(e.Index).Location);

}

}

private void list\_favorite\_DrawItem(object sender, DrawItemEventArgs e)

{

b\_color = e.BackColor;

Color clr = Color.FromArgb(0, b\_color);

if (e.Index >= 0)

{

SolidBrush sb = new SolidBrush(((e.State & DrawItemState.Selected) == DrawItemState.Selected) ? h\_color : b\_color);

e.Graphics.FillRectangle(sb, e.Bounds);

string txt = list\_favorite.Items[e.Index].ToString();

SolidBrush tb = new SolidBrush(e.ForeColor);

e.Graphics.DrawString(txt, e.Font, tb, list\_favorite.GetItemRectangle(e.Index).Location);

}

}

//Очищение

private void textBox1\_MouseClick(object sender, MouseEventArgs e)

{

txt\_search.Text = "";

}

private void txt\_search\_MouseLeave(object sender, EventArgs e)

{

txt\_search.Text = "Поиск";

}

private void txt\_searches\_MouseClick(object sender, MouseEventArgs e)

{

txt\_searches.Text = "";

}

private void txt\_searches\_MouseLeave(object sender, EventArgs e)

{

txt\_searches.Text = "Поиск";

}

//Поиск

private void txt\_search\_KeyUp(object sender, KeyEventArgs e)

{

int index = track\_list.FindString(txt\_search.Text);

if (0 <= index)

{

track\_list.SelectedIndex = index;

}

}

private void txt\_searches\_KeyUp(object sender, KeyEventArgs e)

{

int index = list\_favorite.FindString(txt\_searches.Text);

if (0 <= index)

{

list\_favorite.SelectedIndex = index;

}

}

//Сохранить

private void btn\_save\_Click(object sender, EventArgs e)

{

try

{

if (track\_list.SelectedItem != null)

{

string path = Path.GetFullPath("./Songs/" + files[track\_list.SelectedIndex]);

File.Copy(paths[track\_list.SelectedIndex], path);

Coad();

}

}

catch (Exception)

{

MessageBox.Show(

"Песня уже сохранена!",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1);

}

}

//Удаление

private void btn\_delete\_Click(object sender, EventArgs e)

{

if (list\_favorite.SelectedItem != null)

{

File.Delete(points[list\_favorite.SelectedIndex]);

Coad();

}

}

//Список

private void track\_list\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

if (track\_list.SelectedItem != null)

{

player.URL = paths[track\_list.SelectedIndex];

}

var file = TagLib.File.Create(paths[track\_list.SelectedIndex]);

byte[] bin = (byte[])(file.Tag.Pictures[0].Data.Data);

pic\_art.Image = Image.FromStream(new MemoryStream(bin));

txt\_title.Text = "Название: " + file.Tag.Title;

}

catch { }

}

private void list\_favorite\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

if (list\_favorite.SelectedItem != null)

{

player.URL = points[list\_favorite.SelectedIndex];

}

var file = TagLib.File.Create(points[list\_favorite.SelectedIndex]);

byte[] bin = (byte[])(file.Tag.Pictures[0].Data.Data);

pic\_art.Image = Image.FromStream(new MemoryStream(bin));

txt\_title.Text = "Название: " + file.Tag.Title;

}

catch { }

}

//Зацикливание

private void player\_PlayStateChange(object sender, AxWMPLib.\_WMPOCXEvents\_PlayStateChangeEvent e)

{

if (e.newState == 1)

{

player.Ctlcontrols.play();

}

}

}

}