using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Threading;

namespace Training

{

public delegate void Del(int a);

public partial class Form1 : Form

{

public string str="Rammstein";

Thread t1;

Del deleg;

public Form1()

{

InitializeComponent();

deleg = new Del(M1);

}

private void button1\_Click(object sender, EventArgs e)

{

ParameterizedThreadStart prmt = new ParameterizedThreadStart(M2);

t1 = new Thread(M2);

t1.Start(4);

}

void M1(int a)

{

textBox1.Text += str[a];

}

void M2(object a)

{

int len = (int)a;

Thread.Sleep(2000);

for (int i = 0; i < len; i++)

{

Invoke(deleg, i);

}

}

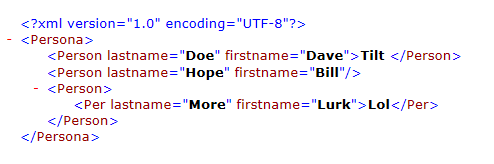
}

}

Проверка на клавишу

if (Console.ReadKey(true).Key = ConsoleKey.Escape)

Чтение с XML-файла



var reader = new XmlTextReader(@"D:/Azaza.xml");

while (reader.Read())

{

if (reader.NodeType == XmlNodeType.Element)

{

if(reader.Name == "Person")

Console.WriteLine(reader.Name + " " + reader.GetAttribute("firstname") + " " + reader.GetAttribute("lastname"));

}

}

XmlReader reader = XmlReader.Create(filestream);

while (reader.Read())

{

if (reader.NodeType == XmlNodeType.Element)

{

if (reader.HasAttributes)

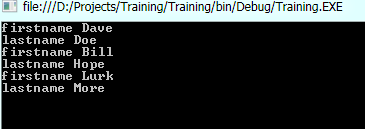
{

while (reader.MoveToNextAttribute()) {

Console.WriteLine(reader.Name + " " + reader.Value);

}

}



Делегаты и лямбда выражения

Del my = new Del((x)=>Console.WriteLine(x));

my(20);

Del me = x => Console.WriteLine(x);

me(21);

var ma = new Del((x) => Console.WriteLine(x));

Del mi = delegate(int x) {Console.WriteLine(x);};

myDelegate = delegate(int x) { return x \* 2; }; // Лямбда-Метод

myDelegate = (x) => { return x \* 2; }; // Лямбда-Оператор.

myDelegate = x => x \* 2; // Лямбда-Выражение.

Func<bool> methodCall = delegate() { return output.SendToFile(); };

if (methodCall())

Console.WriteLine("Success!");

Func<bool> methodCall = () => output.SendToFile();

if (methodCall())

Console.WriteLine("Success!");

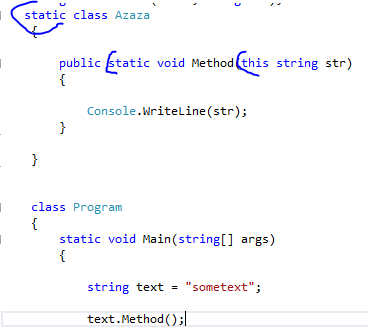
Func<bool> methodCall = () => false;

bool b = methodCall();

Del my = (x, str) => x == str.Length;

bool f = my(5, "pyatt");

Расширяющий метод



Оператор **->** объединяет разыменование указателя и доступ к члену.

Используется для проверки на значения NULL перед выполнением операции доступа к элементу (**?.**) или операции индексирования (**?[**)  Эти операторы позволяют писать меньше кода для проверок значений null, особенно если речь идет о внедрении в структуры данных.

int? length = customers?.Length; // null if customers is null

Customer first = customers?[0]; // null if customers is null

int? count = customers?[0]?.Orders?.Count();