Тимофеева Наталья

М8О-202Б-19

ООП C#

Лабораторная работа № 9

**Описание работы**

ДЕЛЕГАТЫ, АНОНИМНЫЕ ДЕЛЕГАТЫ

Изучите способы реализации и применяя в C# делегатов и анонимных делегатов.

Создайте отдельный проект, в котором:

реализуйте примеры работы делегатов для функций с возвращаемым и невозвращаемым параметром

создание и вызов множественных делегатов для таких функций

создание и вызов анонимных делегатов для разного типа функций

продемонстрируйте работу обобщённых делегатов типа Action<> и Funct<>

ЛЯМБДА-ВЫРАЖЕНИЯ, СОБЫТИЯ

Изучите способы реализации и применяя в C# лямбда-выражений и событий.

Создайте отдельный проект, в котором:

реализуйте различные примеры работы лямбда-выражений для записи анонимных функций с возвращаемым и невозвращаемым параметром

создайте класс типа publisher и несколько классов типа subscriber

приведите пример связывания объекта класса типа publisher с объектами классов типа subscriber через механизма событий

продемонстрируйте работу событий, возникающих при изменении состояния класса publisher и вызывающих реакцию в объектах классов subscriber

**Программа**

**9.1 Делегаты**

using System;

namespace Lab9\_1\_delegate {

class Program {

public delegate void TypeD1(int a, double b);

public delegate int TypeD2(string s, int a);

public delegate void Anon();

public delegate int Fact(int a);

static void Main() {

TypeD1 deleg1 = Max;

TypeD1 dlg1 = new TypeD1(Max);

if (deleg1.Equals(dlg1)) { Console.WriteLine("Equals"); }

else { Console.WriteLine("Not equals"); };

Console.WriteLine(dlg1.Method.ToString());

deleg1(3, 5.833);

deleg1.Invoke(44, 39.5);

deleg1 = Same;

if (deleg1.Equals(dlg1)) { Console.WriteLine("Equals"); }

else { Console.WriteLine("Not equals"); };

deleg1(3, 3.98);

deleg1(74, 12.2);

Console.WriteLine();

TypeD2 deleg2 = Len;

TypeD2 dlg2 = new TypeD2(Len);

if (deleg2.Equals(dlg2)) { Console.WriteLine("Equals"); }

else { Console.WriteLine("Not equals"); };

Console.WriteLine(dlg2.Method.ToString());

int print = deleg2("Sfwgfsa", 9);

Console.WriteLine(print);

deleg2 = Word;

if (deleg2.Equals(dlg2)) { Console.WriteLine("Equals"); }

else { Console.WriteLine("Not equals"); };

Console.WriteLine(deleg2("I am tired", 5));

Console.WriteLine(deleg2("I am tired", 12));

Console.WriteLine(deleg2("I am tired", 4));

Console.WriteLine(deleg2("I am tired", 6));

Console.WriteLine();

deleg1 += Max;

deleg1(2, 2.02);

Delegate[] dlg\_list = deleg1.GetInvocationList();

for (int i = 0; i < dlg\_list.Length; ++i){

Console.WriteLine(dlg\_list[i].Method.ToString());

}

Console.WriteLine();

deleg2 += Len;

Console.WriteLine(deleg2("I am tired", 7));

dlg\_list = deleg2.GetInvocationList();

for (int i = 0; i < dlg\_list.Length; ++i)

{

Console.WriteLine(dlg\_list[i].Method.ToString());

}

Console.WriteLine();

Anon a = delegate () {

Console.WriteLine("Anonymous method");

};

a();

Fact fa = delegate (int n) {

if (n < 0) return -1;

else {

int f = 1;

for (int i = 2; i <= n; ++i) {

f = f \* i;

}

return f;

}

};

Console.WriteLine(fa(11));

Console.WriteLine();

Func<int, int> f = delegate (int a) {

int res = 0;

while (a > 0) {

res = res \* 10 + a % 10;

a /= 10;

}

return res;

};

Console.WriteLine(f(35681));

}

public static void Max(int a, double b) {

if (a > b) { Console.WriteLine("{0} больше {1}", a, b); }

else { Console.WriteLine("{0} больше {1}", b, a); }

}

public static void Same(int a, double b) {

if (a == (int)b) { Console.WriteLine("Идентичны"); }

else { Console.WriteLine("Не идентичны"); }

}

public static int Len(string s, int a) {

if (s.Length == a) { return 0; }

else { return s.Length - a; }

}

public static int Word(string s, int a) {

if (a > s.Length) { return -1; }

else if (s[a-1] == ' ') { return 0; }

else {

int k = 1;

for (int i = 0; i < a; ++i) {

if (s[i] == ' ') {

++k;

}

}

return k;

}

}

}

}

**9.2**

using System;

using System.Collections.Generic;

namespace Lab9\_2\_Lambda {

class Program {

delegate bool Bi(int x);

public delegate void Lambda(int x, int y);

static void Main() {

List<int> numbers = new List<int>() { -10, 0, -46, 19, 1024 };

Bi positive = x => x > 0;

int index = numbers.FindIndex(x => x > 0);

Console.WriteLine(numbers[index]);

Lambda lambda = (x, y) => Console.WriteLine(x \* y);

lambda(5, -11);

Func<double, double> f = x => x \* -1;

Console.WriteLine(f(31.345));

Action a = () => Console.WriteLine("Action!");

a();

}

}

}

**9.3**

using System;

namespace Lab9\_3\_Event {

class Publisher {

public delegate void Account();

public event Account BankAccount;

int sum { get; set; }

public Publisher(int s) { sum = s; }

public void Put(int s) {

sum += s;

if (BankAccount != null) BankAccount();

Console.WriteLine($"На счёт положили {s}. На карте {sum}");

}

public void Take(int s) {

if (sum > s) {

sum -= s;

if (BankAccount != null) BankAccount();

Console.WriteLine($"Со счёта списали {s}. На карте {sum}");

}

else {

if (BankAccount != null) BankAccount();

Console.WriteLine($"Недостаточно средств. На карте {sum}");

}

}

}

class Subscriber {

private Publisher publisher = null;

private int value { set; get; }

public Subscriber(Publisher p, int s) {

publisher = p;

value = s;

publisher.BankAccount += Print;

}

void Print() {

Console.WriteLine("Изменение баланса");

}

public void Put() { publisher.Put(value); }

public void Take() { publisher.Take(value); }

}

class Program {

static void Main() {

Publisher publisher = new Publisher(1000);

Subscriber subscriber1 = new Subscriber(publisher, 700);

Subscriber subscriber2 = new Subscriber(publisher, 5000);

Subscriber subscriber3 = new Subscriber(publisher, 1300);

subscriber1.Put();

subscriber2.Take();

subscriber3.Take();

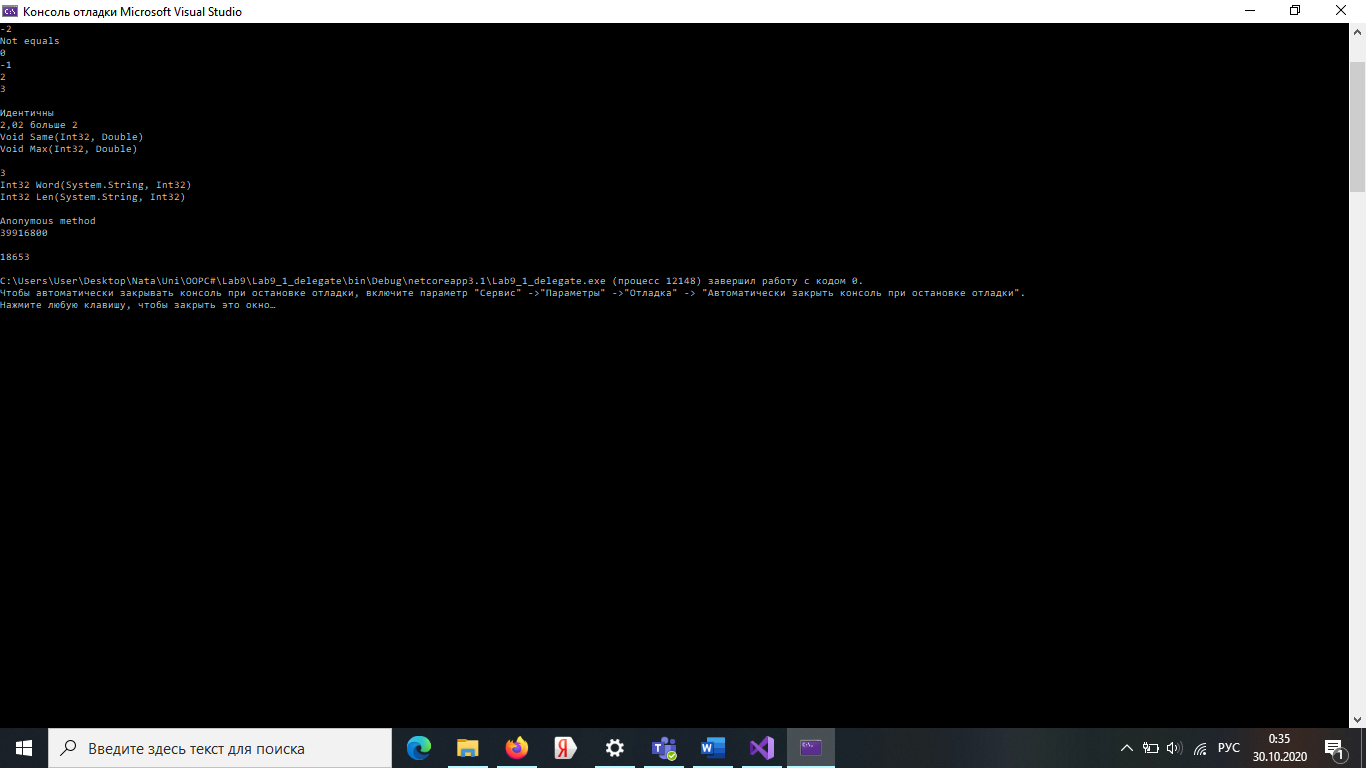
}

}

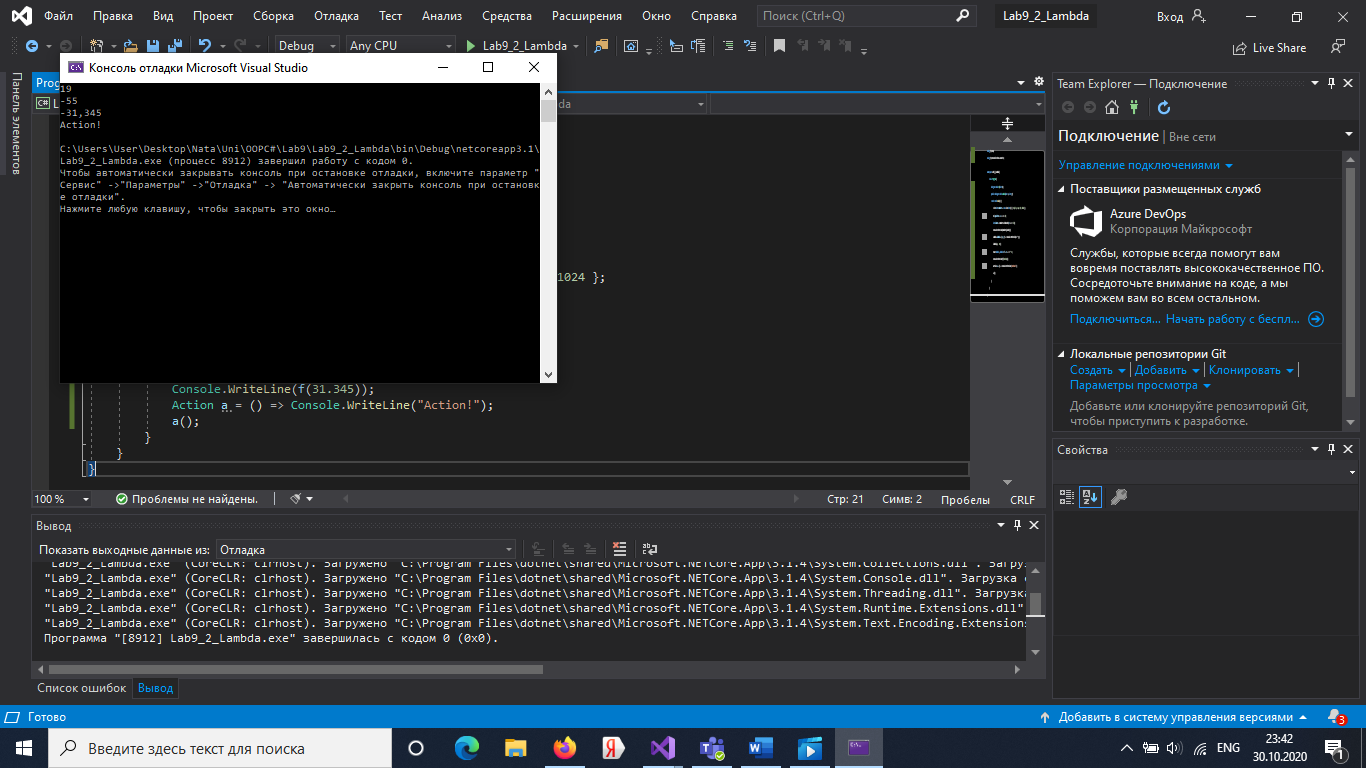
}

**Вывод с консоли**

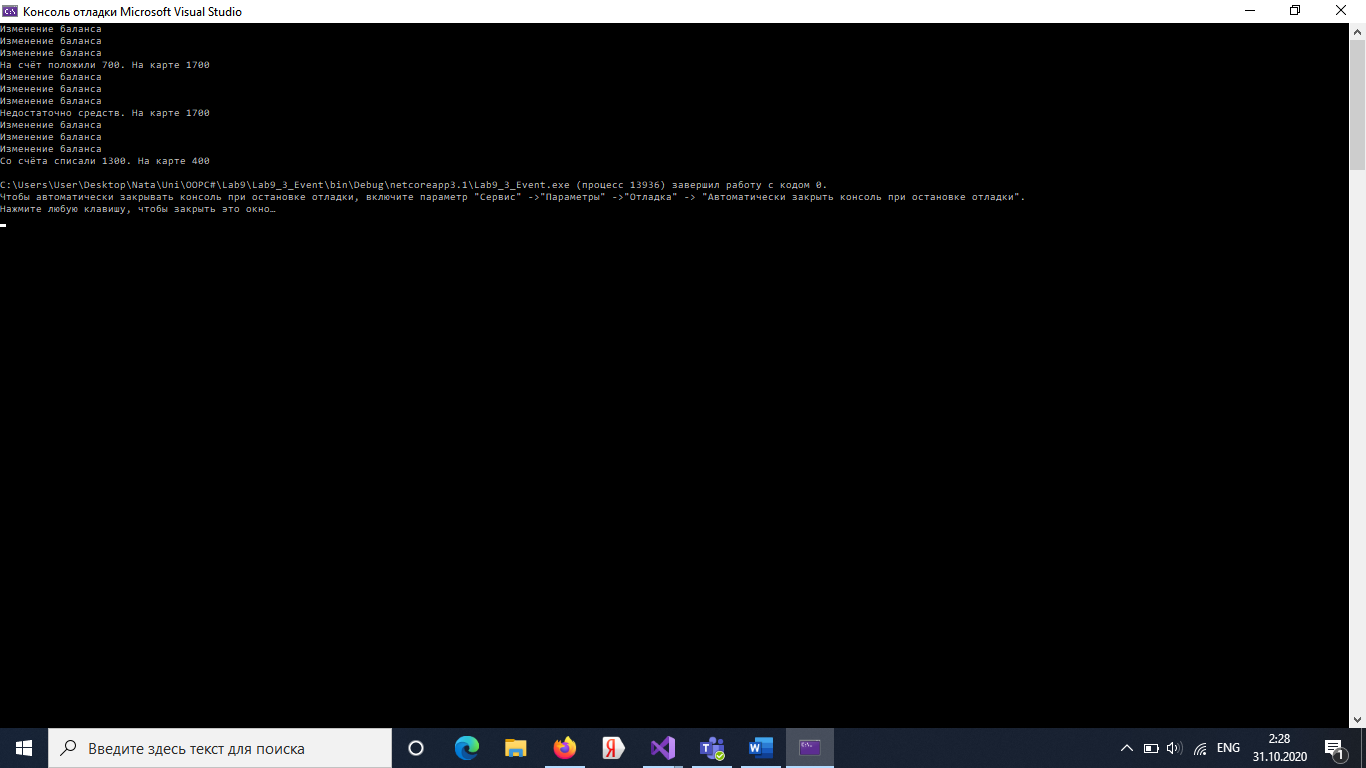
**9.1**



**9.2**



**9.3**



**Вывод**

Делегаты представляют такие объекты, которые указывают на методы. То есть делегаты — это указатели на методы и с помощью делегатов мы можем вызвать данные методы.

MulticastDelegate представляет групповой делегат, то есть делегат, имеющий в своем списке вызовов более одного элемента.

С делегатами тесно связаны анонимные методы. Анонимные методы используются для создания экземпляров делегатов. Определение анонимных методов начинается с ключевого слова delegate, после которого идет в скобках список параметров и тело метода в фигурных скобках.

Еще одним распространенным делегатом является Func. Он возвращает результат действия и может принимать параметры. Он также имеет различные формы: от Func<out T>(), где T - тип возвращаемого значения, до Func<in T1, in T2,...in T16, out TResult>(), то есть может принимать до 16 параметров.

Лямбда-выражения представляют упрощенную запись анонимных методов. Лямбда-выражения позволяют создать емкие лаконичные методы, которые могут возвращать некоторое значение и которые можно передать в качестве параметров в другие методы.

События сигнализируют системе о том, что произошло определенное действие. И если нам надо отследить эти действия, то как раз мы можем применять события.