Введение

Microsoft Visual Studio — линия продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Программирование на языке C# представляет собой перспективное направление, которое позволяет реализовать множество современных подходов программирования.

Цель учебной практики по программированию представляет собой освоение новых навыков и закрепление основ программирования на языке программирования C# в среде Microsoft Visual Studio для платформы Microsoft .NET.

Основыми задачами учебной практики по программированию является:

1. освоение объектно-ориентированного програмирования на основе консольных приложений с реализацией системных библиотечных классов, использованием процедур и функций, их перегрузки, обработки исключительных ситуаций, массивов, строк, перечислений, структур, интерфейсов, механизмов наследования, полиморфизма и инкапсуляции, делегатов и событий, многопоточности, обобщенных классов, работы с файлами, создания пользовательских библиотек, использования коллекций и классов-прототипов, указателей и регулярных выражений;
2. разработка Windows-приложений на основе кроссплатформенного GUI, собственных компонентов средствами C#;
3. реализация реляционных СУБД с подключением базы данных и исполнения запросов на примере MySQL.

В заключении будут подведены итоги учебной практики по программированию.

1 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКАХ C++, C# И ПЛАТФОРМА MICROSOFT .NET FRAMEWORK

1.1 Решение задания по использованию библиотечных классов для математических вычислений и ввода-вывода

Для выполнения различных математических операций в библиотеке классов .NET предназначен класс Math. Предоставляет константы и статические методы для тригонометрических, логарифмических и иных общих математических функций. Он является статическим, поэтому все его методы также являются статическими. Также класс Math определяет две константы: Math.E и Math.PI.

Условные конструкции - один из базовых компонентов многих языков программирования, которые направляют работу программы по одному из путей в зависимости от определенных условий.

В языке C# используются следующие условные конструкции: if..else и switch..case.

Конструкция if/else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код.

После ключевого слова if ставится условие. И если это условие выполняется, то срабатывает код, который помещен далее в блоке if после фигурных скобок.

При несоблюдении условия можно добавить блок else.

Но при сравнении чисел мы можем насчитать три состояния: первое число больше второго, первое число меньше второго и числа равны. Используя конструкцию else if, мы можем обрабатывать дополнительные условия: Также мы можем соединить сразу несколько условий, используя логические операторы.

Условие задачи: Дан угол в радианах. Написать программу, переводящую его в градусы, минуты и секунды (градусы и минуты — целые числа). Ответ вывести в виде: «Угол a рад равен d° m′ s′′». Вместо буквенных обозначений должны стоять конкретные числа с точностью до 2-го знака после запятой. Перед запросом ввода с клавиатуры выводить подсказку.

Алгоритм решения: Пользователь ввёл в консоль математическое выражение (величину угла в радианах). Выполнилась проверка на введенное значение угла, выражение было вычислено и запомнилось в переменной вещественного типа. Далее исполнились переводы полученного числа в градусы, минуты и секунды при помощи математических формул. На консоль вывелся полученный результат.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Console.Write("Введите угол a в радианах. Для ввода можно использовать число pi (значения вводятся без пробелов, pi - если есть - указывается как 'p' латинское; пример ввода - p\*3; 4,35/p; 25): ");

back1: string UgolRad = Console.ReadLine();

double MathUgolRad = 0, Seconds = 0;

int UgolGrad = 0, Minutes = 0;

if (UgolRad.Length > 0)

{

if (UgolRad.IndexOf("p") > -1)

{

if (UgolRad.Length > 1 && UgolRad.IndexOf("p") == 0)

{

if (UgolRad.Substring(1, 1) == "\*")

MathUgolRad = Math.PI \* Convert.ToDouble(UgolRad.Substring(2, UgolRad.Length - 2));

else if (UgolRad.Substring(1, 1) == "/")

MathUgolRad = Math.PI / Convert.ToDouble(UgolRad.Substring(2, UgolRad.Length - 2));

else if (UgolRad.Substring(1, 1) == "+")

MathUgolRad = Math.PI + Convert.ToDouble(UgolRad.Substring(2, UgolRad.Length - 2));

else if (UgolRad.Substring(1, 1) == "-")

MathUgolRad = Math.PI - Convert.ToDouble(UgolRad.Substring(2, UgolRad.Length - 2));

}

else if (UgolRad == "-p")

MathUgolRad = -Math.PI;

else if (UgolRad.Length > 1 && UgolRad.IndexOf("p") == 1)

{

if (UgolRad.Substring(2, 1) == "\*")

MathUgolRad = -Math.PI \* Convert.ToDouble(UgolRad.Substring(3, UgolRad.Length - 3));

else if (UgolRad.Substring(2, 1) == "/")

MathUgolRad = -Math.PI / Convert.ToDouble(UgolRad.Substring(3, UgolRad.Length - 3));

else if (UgolRad.Substring(2, 1) == "+")

MathUgolRad = -Math.PI + Convert.ToDouble(UgolRad.Substring(3, UgolRad.Length - 3));

else if (UgolRad.Substring(2, 1) == "-")

MathUgolRad = -Math.PI - Convert.ToDouble(UgolRad.Substring(3, UgolRad.Length - 3));

}

else if (UgolRad.Length > 1 && UgolRad.IndexOf("p") == UgolRad.Length - 1)

{

if (UgolRad.Substring(UgolRad.Length - 2, 1) == "\*")

MathUgolRad = Math.PI \* Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 2));

else if (UgolRad.Substring(UgolRad.Length - 2, 1) == "/")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 2)) / Math.PI;

else if (UgolRad.Substring(UgolRad.Length - 2, 1) == "+")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 2)) + Math.PI;

else if (UgolRad.Substring(UgolRad.Length - 2, 1) == "-")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 2)) - Math.PI;

}

else if (UgolRad.Length > 1 && UgolRad.IndexOf("p") == UgolRad.Length - 2)

{

if (UgolRad.Substring(UgolRad.Length - 3, 1) == "\*")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 4)) \* -Math.PI;

else if (UgolRad.Substring(UgolRad.Length - 3, 1) == "/")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 4)) / -Math.PI;

else if (UgolRad.Substring(UgolRad.Length - 3, 1) == "+")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 4)) + -Math.PI;

else if (UgolRad.Substring(UgolRad.Length - 3, 1) == "-")

MathUgolRad = Convert.ToDouble(UgolRad.Substring(0, UgolRad.Length - 4)) - -Math.PI;

}

else

MathUgolRad = Math.PI;

}

else

MathUgolRad = Convert.ToDouble(UgolRad);

}

else

{

Console.WriteLine("Повторите ввод.");

goto back1;

}

UgolGrad = Convert.ToInt32(MathUgolRad \* (180 / Math.PI));

Minutes = Convert.ToInt32(Math.Round(Math.Abs(MathUgolRad \* (180 / Math.PI) - UgolGrad) \* 60));

Seconds = Math.Abs((MathUgolRad \* (180 / Math.PI) - UgolGrad) \* 3600 - Minutes \* 60);

Console.WriteLine("Угол a рад равен {0:0}°, {1:0} мин, {2:0.00} с", UgolGrad, Minutes, Seconds);

Console.ReadKey();

}

}

}

Тестирование приложения:

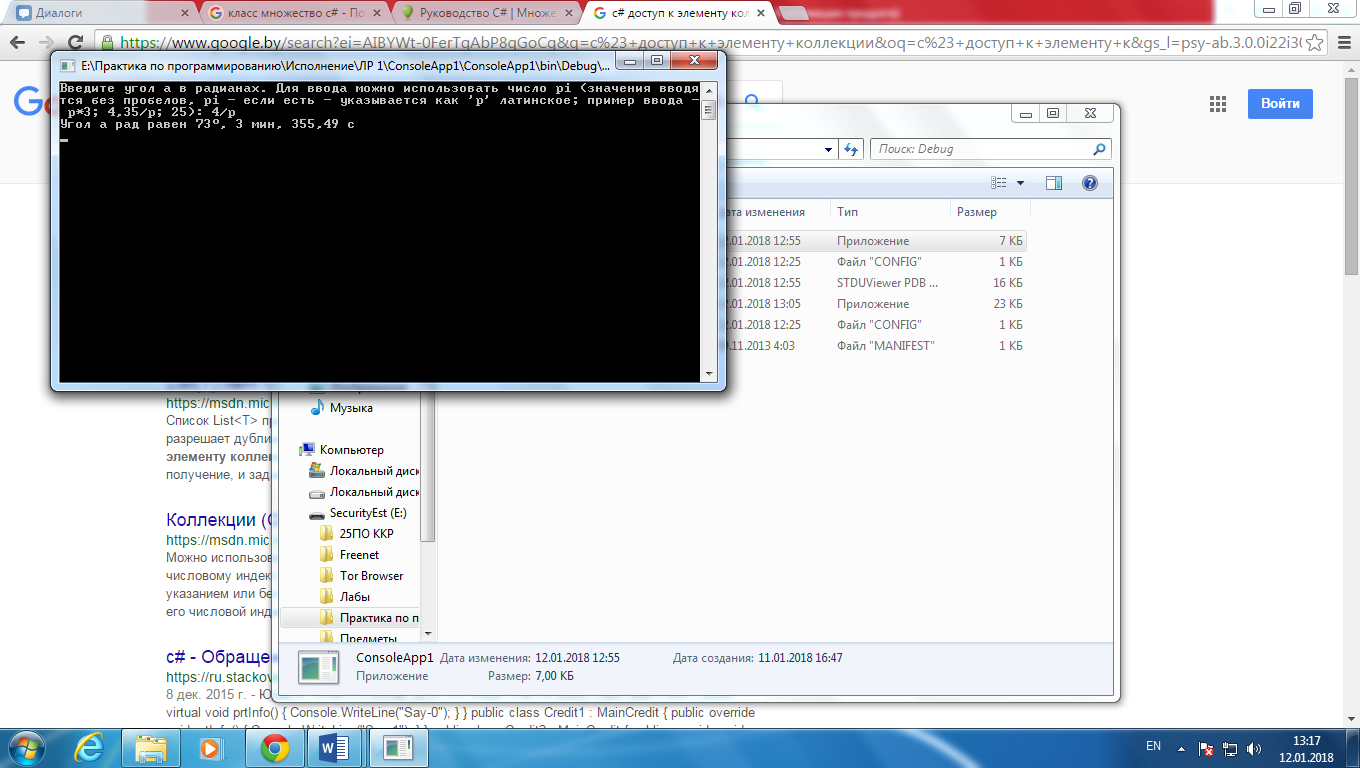


Рисунок 1 – Исполнение программы

1.2 Решение задания с использованием процедур и функций. Перегрузка функций (методов) и операторов

В языке C# существует 2 способа передачи аргументов: передача по значению и передача по ссылке, и результат передачи объекта аргументу по ссылке будет отличаться от передачи значения простого типа.

Передача аргумента по значению — значение фактически копируется в формальный параметр метода. Соответственно, изменения, которые вносятся в параметр метода никак не влияют на аргумент, используемый для вызова. В случае с передачей по ссылке — всё наоборот. Так как передаётся не значение, а ссылка на объект, то изменения, которые вносятся в параметр метода, влияют и на аргумент, используемый для вызова.

Перегрузка функций - это определения нескольких функций с одинаковыми именами, но различными параметрами. Наборы параметров перегруженных функций могут отличаться порядком следования, количеством, типом.

Функции, имеющие одинаковые имена и различные сигнатуры и определенные в одной области видимости, называются перегруженными.

Выбор экземпляра функции при вызове включает сравнение типов и числа фактических аргументов с формальными параметрами в объявлении каждого экземпляра.

При вызове функций с несколькими аргументами правила соответствия применяются для каждого аргумента. При перегрузке такой функции выбирается тот экземпляр, для которого достигнуто либо точное соответствие, либо достигнуто соответствие при помощи преобразований.

Если при перегрузке функций есть один или несколько экземпляров с аргументами по умолчанию, то вызываться будет тот, для которого обеспечены все или некоторое подмножество аргументов, к которым применимы правила соответствия.

Условие задачи: Создать заданный в варианте класс. Определить в классе конструкторы, деструктор, необходимые функции и заданные перегруженные операции. Написать программу тестирования, в которой проверяется использование всех перегруженных операций. Класс – множество Сset. Дополнительно перегрузить следующие операции: () – конструктор множества (в стиле конструктора для множественного типа); + – объединение множеств; <= – сравнение множеств; int()– мощность множества; [] – доступ к элементу в заданной позиции.

Алгоритм решения: Задаётся два множества, к первому добавился новый элемент через функцию, прописанную в классе. На консоль вывелись множества, затем была высчитана их мощность и отобразился результат, в том числе сравнение. Был введен номер элемента, где выполнилась проверка на его существование, и, если номер элемента найден, отображен на консоли.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

public class Cset<T>

{

private List<T> \_list;

Cset()

{

\_list = new List<T>();

}

public Cset(params T[] args)

: this()

{

\_list.AddRange(args);

}

public Cset(IEnumerable<T> mas)

: this()

{

\_list.AddRange(mas);

}

public void Add(T elem)

{

\_list.Add(elem);

}

public void Delete(T elem)

{

\_list.Remove(elem);

}

public static Cset<T> operator +(Cset<T> Source1, Cset<T> Source2)

{

return new Cset<T>(Source2.\_list.Union(Source1.\_list));

}

public int Power()

{

return \_list.Count;

}

public static bool operator <=(Cset<T> Source1, Cset<T> Source2)

{

if (Source1.Power() <= Source2.Power())

{

Console.WriteLine("Мощность множества 1 равна или меньше мощности множества 2");

return true;

}

else

{

Console.WriteLine("Мощность множества 1 больше мощности множества 2");

return false;

}

}

public static bool operator >=(Cset<T> Source1, Cset<T> Source2)

{

if (Source1.Power() >= Source2.Power())

{

Console.WriteLine("Мощность множества 1 равна или больше мощности множества 2");

return true;

}

else

{

Console.WriteLine("Мощность множества 1 меньше мощности множества 2");

return false;

}

}

public void Elem(int a)

{

Console.WriteLine("Элемент {0} равен {1}", a, \_list[a]);

}

public void Elem(double a)

{

Console.WriteLine("Элемент {0} равен {1}", a, \_list[Convert.ToInt32(a)]);

}

public override string ToString()

{

return string.Join(",", \_list);

}

}

class Program

{

static void Main(string[] args)

{

Cset<int> \_mt1 = new Cset<int>(1, 2, 3, 4, 5);

\_mt1.Add(6);

Console.WriteLine("Множество 1: {0}", \_mt1);

Cset<int> \_mt2 = new Cset<int>(4, 5, 6, 7, 8);

Console.WriteLine("Множество 2: {0}", \_mt2);

Console.WriteLine("Мощность множества 1: {0}", \_mt1.Power());

Console.WriteLine("Мощность множества 2: {0}", \_mt2.Power());

if (\_mt1 <= \_mt2)

Console.WriteLine("<=");

else

Console.WriteLine(">");

Console.WriteLine("Объединение множеств: {0}", \_mt1 += \_mt2);

Console.Write("Введите номер элемента множества 2: ");

back1: int a = int.Parse(Console.ReadLine());

if (a > \_mt2.Power() - 1)

{

Console.WriteLine("Превышает количество элементов множества, повторите ввод");

goto back1;

}

\_mt2.Elem(a);

Console.ReadKey();

}

}

}

Тестирование приложения:

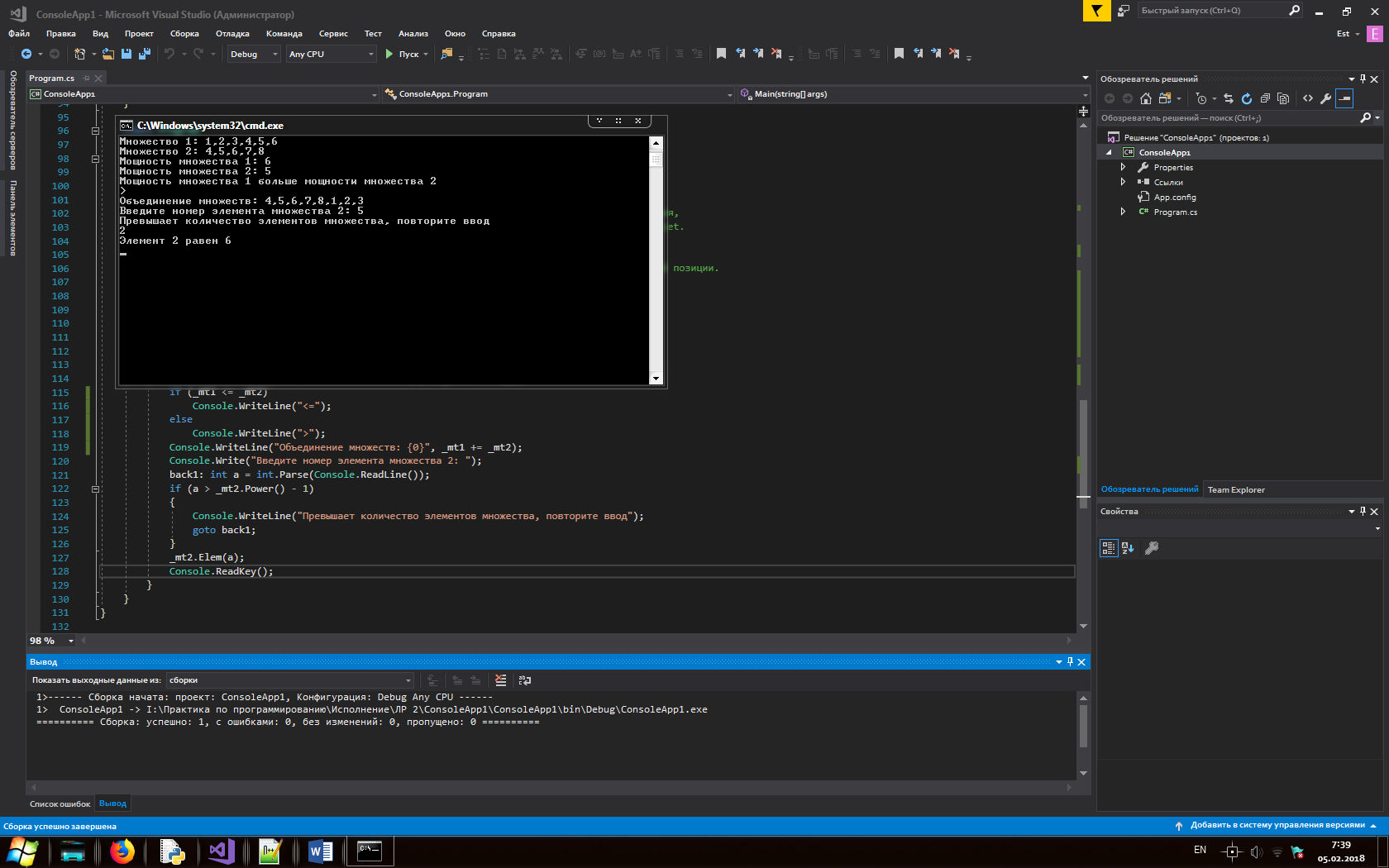


Рисунок 2 – Исполнение программы

1.3 Обработка исключительных ситуаций

В языках программирования для обработки исключительных ситуаций предлагались самые разные подходы.

Всюду в тексте модуля, где синтаксически допускается использование блока, этот блок можно сделать охраняемым, добавив ключевое слово try. Вслед за try-блоком могут следовать catch-блоки, называемые блоками-обработчиками исключительных ситуаций, их может быть несколько, они могут и отсутствовать. Завершает эту последовательность finally-блок - блок финализации, который также может отсутствовать. Вся эта конструкция может быть вложенной - в состав try-блока может входить конструкция try-catch-finally.

Условие задачи: Вычислить и вывести на экран значение функции F(x) на отрезке [a; b] c шагом h = 0.1 с точностью e. Результат работы программы представить в виде следующей таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Значение x | Значение функции F(x) | Количество просуммированных слагаемых n |
| 1 |  |  |  |

Замечание. При решении задачи использовать вспомогательную функцию.

F(x) = \frac{x-1}{x}+\frac{(x-1)^2}{2x^2}+\frac{(x-1)^3}{3x^3}+…, x\in [1;2].

Алгоритм решения: Вводились значения отрезка [a; b]. В цикле с шагом 0.1 выполнялась функция на каждой итерации и выдавались значения в табличном представлении.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static double Func(double x, int count)

{

return Math.Pow(x - 1, count) / (count \* Math.Pow(x, count));

}

static void Main(string[] args)

{

int count = 1;

Console.WriteLine("Введите значения a и b: ");

double a = double.Parse(Console.ReadLine());

double b = double.Parse(Console.ReadLine());

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| № | Значение x | Значение функции F(x)| Кол-во просуммированных слагаемых n |");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

for (double i = a; i <= b; i += 0.1)

{

double F = 0;

F = Func(i, count);

count++;

Console.WriteLine("| {0} | {1:0.0} | {2:0.000000000} | {3} |", count - 1, i, F, count - 1);

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.ReadKey();

}

}

}

Тестирование приложения:

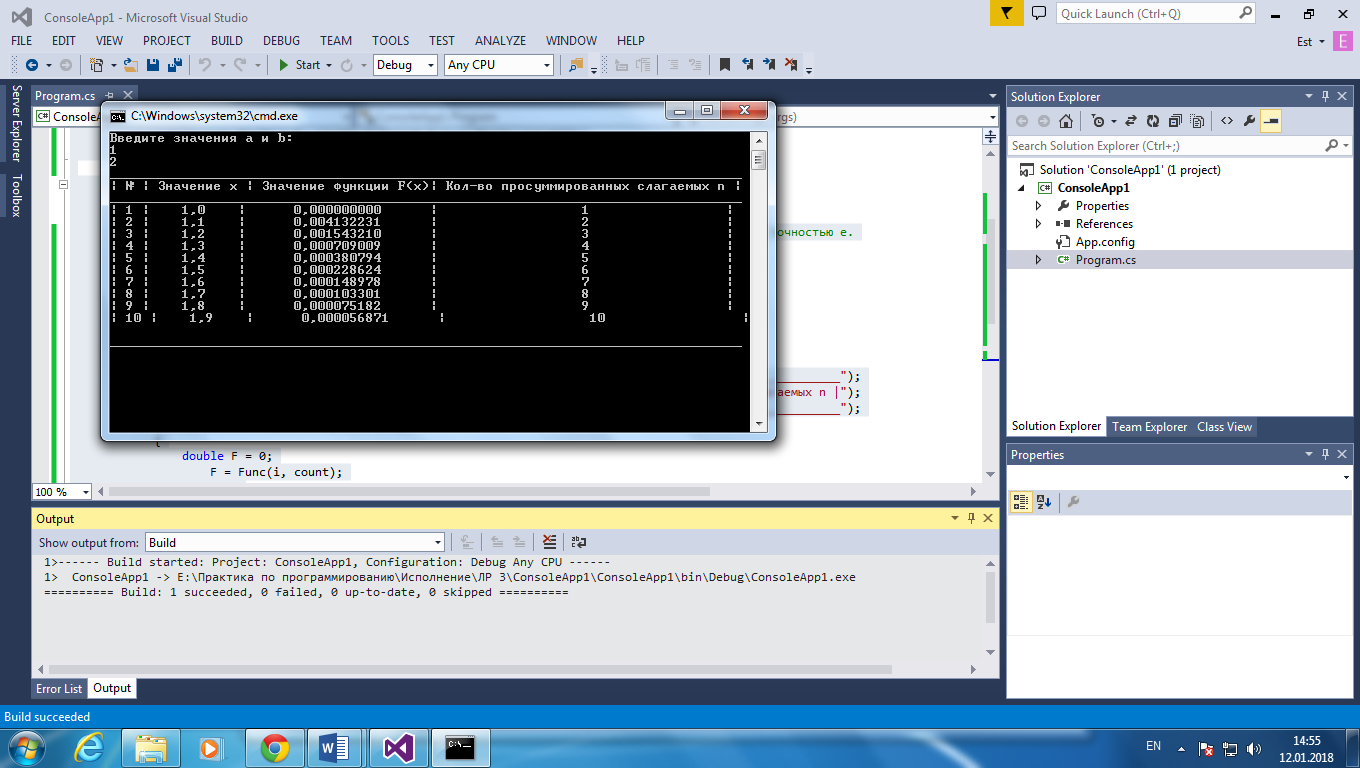


Рисунок 3 – Исполнение программы

1.4 Решение задания с использованием массивов

В языке C#, соблюдая преемственность, сохранены одномерные массивы и массивы массивов. В дополнение к ним в язык добавлены многомерные массивы. Динамические многомерные массивы языка C# являются весьма мощной, надежной, понятной и удобной структурой данных, которую смело можно рекомендовать к применению не только профессионалам, но и новичкам, программирующим на C#. После этого краткого обзора давайте перейдем к более систематическому изучению деталей работы с массивами в C#.

Объявление массивов.

Рассмотрим, как объявляются одномерные массивы, массивы массивов и многомерные массивы.

Объявление одномерных массивов

Напомню общую структуру объявления:

[<атрибуты>] [<модификаторы>] <тип> <объявители>;

Забудем пока об атрибутах и модификаторах. Объявление одномерного массива выглядит следующим образом:

<тип>[] <объявители>;

Заметьте, в отличие от языка C++ квадратные скобки приписаны не к имени переменной, а к типу. Они являются неотъемлемой частью определения типа, так что запись T[] следует понимать как тип, задающий одномерный массив с элементами типа T.

Условие задачи: Для заданного одномерного массива A из N элементов найти количество элементов массива, для которых выполняется условие *sin Ai* < *cos Ai*. В рекурсивной функции каждый раз отделять последнюю треть от первых двух третей рассматриваемой части массива и применять эту же функцию к обеим частям. Рекурсивные вызовы заканчивать, когда останется только один элемент в рассматриваемой части массива. Например, для N=12:



Алгоритм решения: Ввелась размерность массива, произошло его заполнение. Были вычислены первые две трети от размера и оставшаяся часть. Произошел вызов рекуррентной функции для первой части, затем для второй и сложили полученные счётчики.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static int sum = 0; //Счётчик

static int[] A; //Массив

static int Rec(int c, int n)

{

if (Math.Sin(A[c]) < Math.Cos(A[c]))

sum++;

if (c < n - 1)

Rec(c + 1, n);

return sum;

}

static void Main(string[] args)

{

Random rand = new Random();

Console.Write("Введите кол-во элементов: ");

int N = int.Parse(Console.ReadLine());

A = new int[N];

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

{

A[i] = rand.Next(-9, 10);

Console.Write("{0} ", A[i]);

}

Console.WriteLine();

int tret = (N / 3) \* 2;

Console.WriteLine("Первые две трети равны {0}.", tret);

int count1 = 0, count2 = tret;

int Count = Rec(count1, tret);

sum = 0;

Count += Rec(count2, N);

Console.WriteLine("Количество элементов, где его синус меньше косинуса = {0}", Count);

}

}

}

Тестирование приложения:

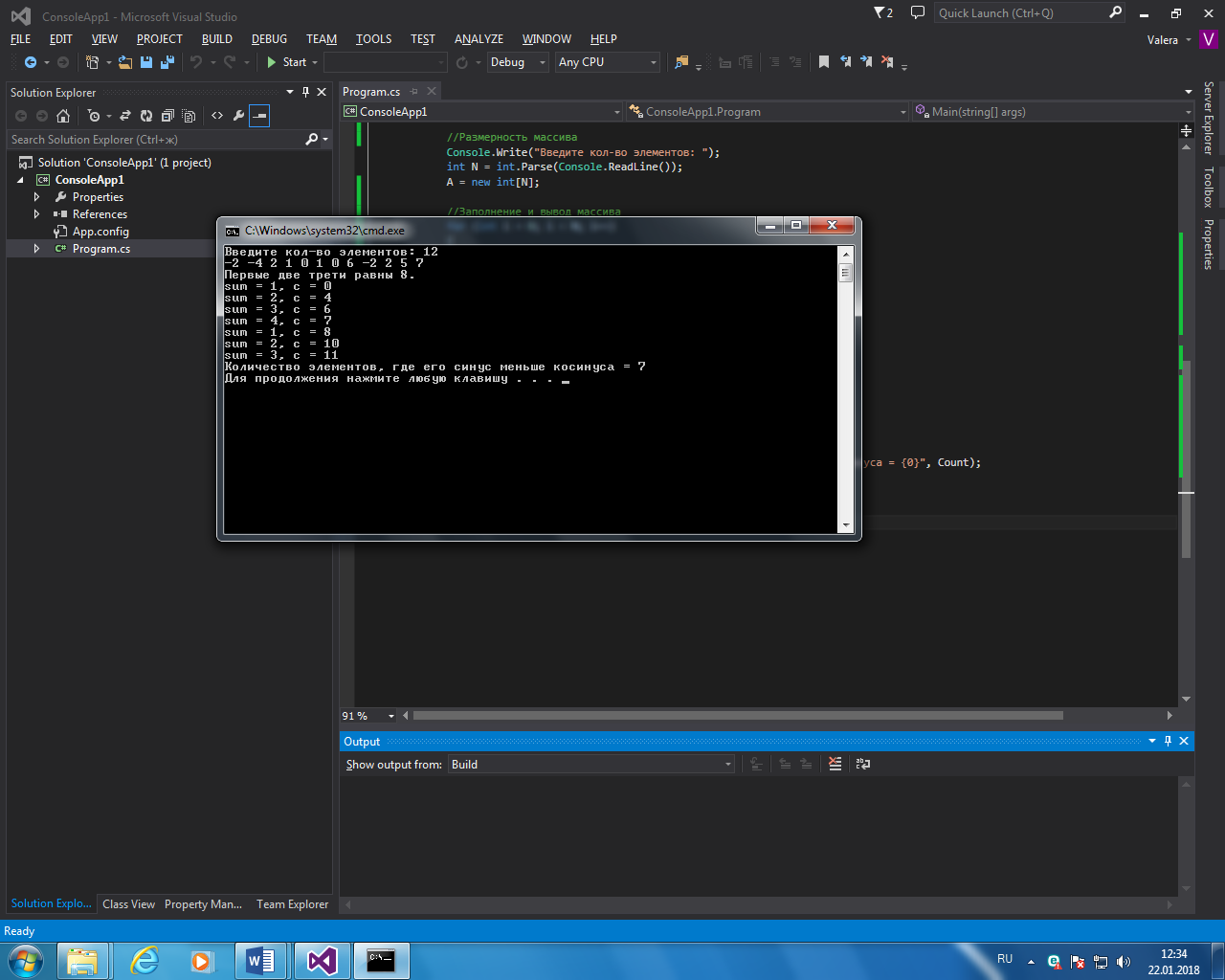


Рисунок 4 – Исполнение программы

1.5 Решение задания с использованием строк

Отличительной особенностью обратной польской нотации является то, что все аргументы (или операнды) расположены перед знаком операции. В общем виде запись выглядит следующим образом:

Запись набора операций состоит из последовательности операндов и знаков операций. Операнды в выражении при письменной записи разделяются пробелами.

Выражение читается слева направо. Когда в выражении встречается знак операции, выполняется соответствующая операция над двумя последними встретившимися перед ним операндами в порядке их записи. Результат операции заменяет в выражении последовательность её операндов и её знак, после чего выражение вычисляется дальше по тому же правилу.

Результатом вычисления выражения становится результат последней вычисленной операции.

Например, рассмотрим вычисление выражения 7 2 3 \* - (эквивалентное выражение в инфиксной нотации: 7-2\*3).

Первый по порядку знак операции — «\*», поэтому первой выполняется операция умножения над операндами 2 и 3 (они стоят последними перед знаком). Выражение при этом преобразуется к виду 7 6 - (результат умножения — 6, — заменяет тройку «2 3 \*»).

Второй знак операции — «-». Выполняется операция вычитания над операндами 7 и 6.

Вычисление закончено. Результат последней операции равен 1, это и есть результат вычисления выражения.

Очевидное расширение обратной польской записи на унарные, тернарные и операции с любым другим количеством операндов: при использовании знаков таких операций в вычислении выражения операция применяется к соответствующему числу последних встретившихся операндов

Особенности обратной польской записи следующие:

Порядок выполнения операций однозначно задаётся порядком следования знаков операций в выражении, поэтому отпадает необходимость использования скобок и введения приоритетов и ассоциативности операций.

В отличие от инфиксной записи, невозможно использовать одни и те же знаки для записи унарных и бинарных операций. Так, в инфиксной записи выражение 5 \* (-3 + 8) использует знак «минус» как символ унарной операции (изменение знака числа), а выражение (10 - 15) \* 3 применяет этот же знак для обозначения бинарной операции (вычитание). Конкретная операция определяется тем, в какой позиции находится знак. Обратная польская запись не позволяет этого: запись 5 3 - 8 + \* (условный аналог первого выражения) будет интерпретирована как ошибочная, поскольку невозможно определить, что «минус» после 5 и 3 обозначает не вычитание; в результате будет сделана попытка вычислить сначала 5 - 3, затем 2 + 8, после чего выяснится, что для операции умножения не хватает операндов. Чтобы всё же записать это выражение, придётся либо переформулировать его (например, записав вместо выражения - 3 выражение 0 - 3), либо ввести для операции изменения знака отдельное обозначение, например, «±»: 5 3 ± 8 + \*.

Так же, как и в инфиксной нотации, в ОПН одно и то же вычисление может быть записано в нескольких разных вариантах. Например, выражение (10 - 15) \* 3 в ОПН можно записать как 10 15 - 3 \*, а можно — как 3 10 15 - \*

Из-за отсутствия скобок обратная польская запись короче инфиксной. За этот счёт при вычислениях на калькуляторах повышается скорость работы оператора (уменьшается количество нажимаемых клавиш), а в программируемых устройствах сокращается объём тех частей программы, которые описывают вычисления. Последнее может быть немаловажно для портативных и встроенных вычислительных устройств, имеющих жёсткие ограничения на объём памяти.

Условие задачи: Написать программу формирования ОПЗ и расчета полученного выражения. Разработать удобный интерфейс ввода исходных данных и вывода результатов. Работу программы проверить на конкретном примере.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | Выражение | a | b | c | d | e | Результат |
| 11 | a– (b/c\*(d+e)) | 5.6 | 3.2 | 0.9 | 1.7 | 4.8 | – 17.51 |

Алгоритм решения: Пользователем ввелось выражение. Данные были записаны в новую строку: изначально числа, затем, исходя из приоритетных значений, символы – при помощи вспомогательной коллекции. В консоли вывелась конечная строка в обратной польской записи, исходя из которой при помощи чисел во вспомогательной коллекции использовались арифметические операции для получения новых величин до конечного результата, который был выведен на консоль.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

private static byte Prioritet(char v)

{

switch (v)

{

case '(':

return 0;

case ')':

return 1;

case '+':

return 2;

case '-':

return 2;

case '\*':

return 3;

case '/':

return 3;

default:

return 3;

}

}

private static bool operat(char v)

{

if (("+-/\*^()".IndexOf(v) != -1))

return true;

return false;

}

static void Main(string[] args)

{

Console.Write("Введите выражение: ");

string input = Console.ReadLine();

string output = string.Empty;

Stack<char> Operacii = new Stack<char>();

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

{

if (Char.IsDigit(input[i]))

{

while (!operat(input[i]))

{

output += input[i];

i++;

if (i == input.Length)

break;

}

output = output + " ";

i--;

}

if (operat(input[i]))

{

if (input[i] == '(')

Operacii.Push(input[i]);

else if (input[i] == ')')

{

char s = Operacii.Pop();

while (s != '(')

{

output += s.ToString() + ' ';

s = Operacii.Pop();

}

}

else

{

if (Operacii.Count > 0)

if (Prioritet(input[i]) <= Prioritet(Operacii.Peek()))

output = output + Operacii.Pop().ToString() + " ";

Operacii.Push(char.Parse(input[i].ToString()));

}

}

}

while (Operacii.Count > 0)

output += Operacii.Pop() + " ";

Console.WriteLine("Выражение в обратной польской записи: " + output);

double result = 0;

Stack<double> StackReshenie = new Stack<double>();

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

{

if (Char.IsDigit(output[i]))

{

string a = string.Empty;

while (output[i] != ' ' && !operat(output[i]))

{

a = a + output[i];

i++;

if (i == output.Length)

break;

}

StackReshenie.Push(double.Parse(a));

i--;

}

else if (operat(output[i]))

{

double a = StackReshenie.Pop();

double b = StackReshenie.Pop();

switch (output[i])

{

case '+':

result = b + a;

break;

case '-':

result = b - a;

break;

case '\*':

result = b \* a;

break;

case '/':

result = b / a;

break;

}

StackReshenie.Push(result);

}

}

Console.WriteLine("Ответ: " + StackReshenie.Peek());

Console.ReadKey();

}

}

}

Тестирование приложения:

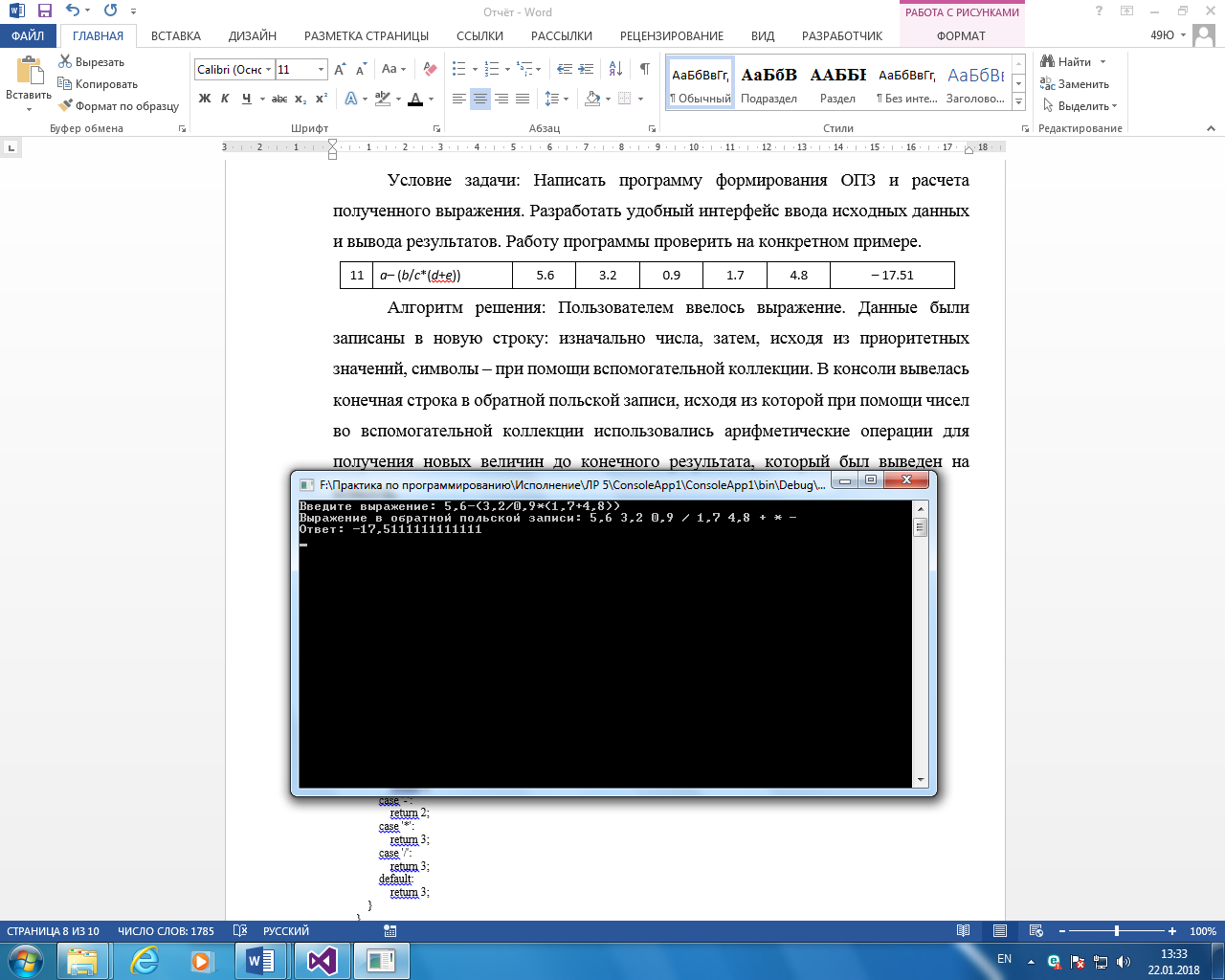


Рисунок 5 – Исполнение программы

1.6 Решение задания с использованием перечислений, структур, интерфейсов

Структура – это более простая версия классов. Все структуры наследуются от базового класса System.ValueType и являются типами значений, тогда как классы - ссылочные типы.

Структуры подходят для создания несложных типов, таких как точка, цвет, окружность. Если необходимо создать множество экземпляров подобного типа, используя структуры, мы экономим память, которая могла бы выделяться под ссылки в случае с классами.

Условие задачи: Описать структуру BOOK (автор, название книги, издательство, год издания, количество страниц). Введите данные в массив, состоящий из шести элементов типа BOOK. Выведите на экран информацию о книгах определенного автора. Если таких книг нет, то программа должна выдать соответствующее сообщение на экран.

Алгоритм решения: Задан массив структур из шести элементов. Пользователь заполнил необходимые поля, затем ввёл искомого автора. Если автор был найден, списком на консоли вывелись сведения о каждой его книге, иначе уведомление об отсутствии его произведений.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

struct BOOK

{

public string author, name, pub;

public int year, pages;

}

public class Program

{

static void Main(string[] args)

{

BOOK[] book = new BOOK[6];

Console.WriteLine("Заполнение данных:");

for (int i = 0; i < 6; i++)

{

Console.WriteLine("Книга №{0}:", i + 1);

Console.Write("Введите автора: ");

book[i].author = Console.ReadLine();

Console.Write("Введите название книги: ");

book[i].name = Console.ReadLine();

Console.Write("Введите издательство: ");

book[i].pub = Console.ReadLine();

Console.Write("Введите год издания: ");

book[i].year = int.Parse(Console.ReadLine());

Console.Write("Введите количество страниц: ");

book[i].pages = int.Parse(Console.ReadLine());

Console.WriteLine();

}

Console.Write("Введите автора: ");

string vvod = Console.ReadLine();

int count = 0;

bool b = false;

for (int i = 0; i < 6; i++)

{

if (book[i].author == vvod)

{

count++;

Console.WriteLine("{0}) название книги '{1}', издательство '{2}', год издания {3} и количество страниц {4}", count, book[i].name, book[i].pub, book[i].year, book[i].pages);

b = true;

}

}

if (b == false)

{

Console.WriteLine("Нет книг данного автора.");

}

Console.ReadKey();

}

}

}

Тестирование приложения:

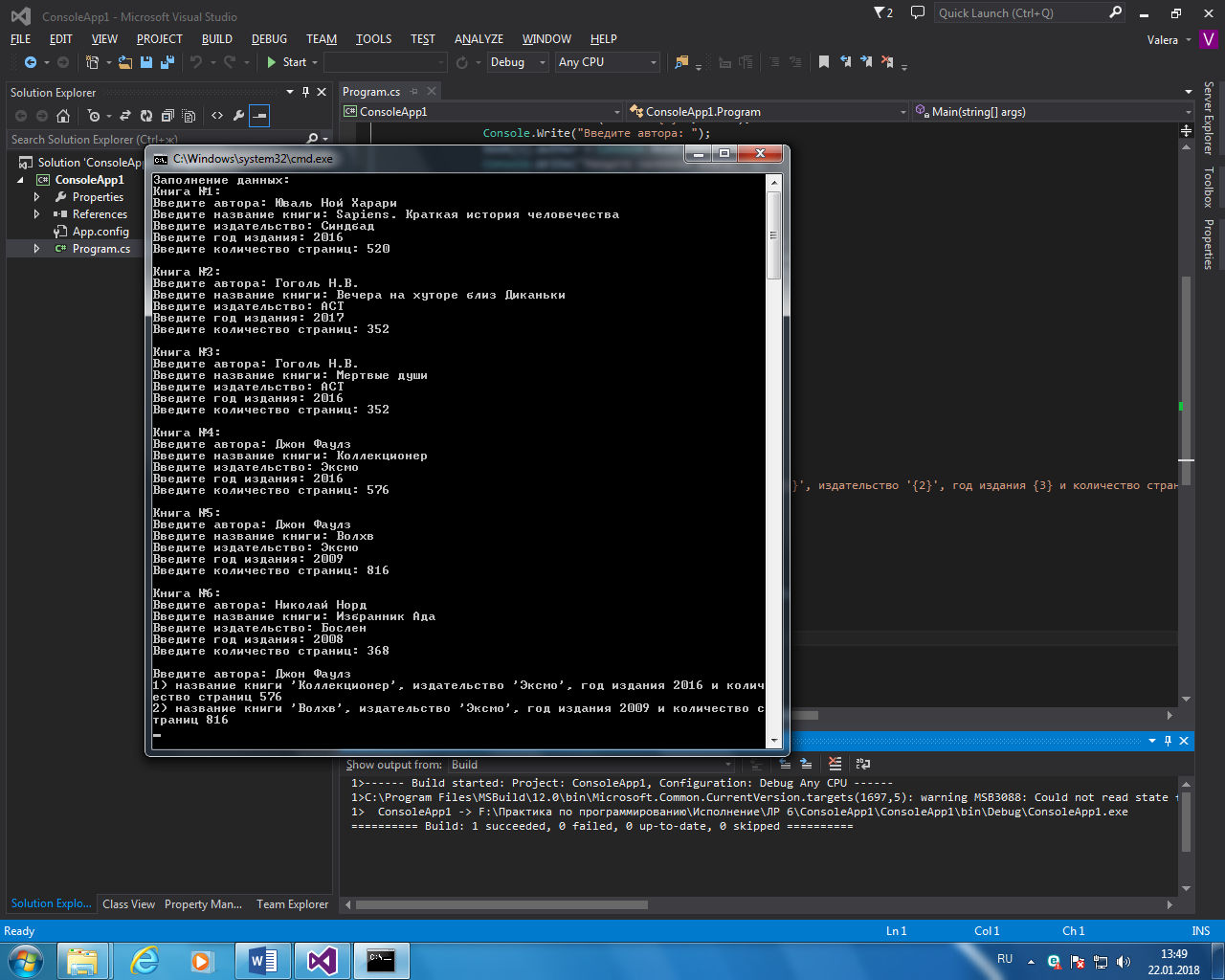


Рисунок 6 – Исполнение программы

1.7 Реализация механизма наследования

Наследование – это тот аспект ООП, который способствует многократному использованию кода. Основная идея наследования состоит в том, что новые классы могут использовать и/или расширять функциональность других классов. При этом поддерживается концепция иерархической классификации, имеющей направление сверху вниз. Используя наследование, объект должен определить только те качества, которые делают его уникальным в пределах своего класса.

Синтаксис:

class имя\_класса : имя\_родительского\_класса

{тело\_класса}

В семействе классов базовый класс часто является обобщенным (в нашем примере таков класс сотрудника Employee). Его задача состоит в определении общих полей, свойств и методов для классов-наследников, а создание экземпляров обобщенного класса не имеет смыла. В этом случае в C# используют ключевое слово abstract.

Назначение абстрактного класса заключается в предоставлении общего определения для базового класса, которое могут совместно использовать несколько производных классов. Создавать экземпляры абстрактного класса нельзя. Абстрактные классы могут определять абстрактные методы. Для этого перед типом возвращаемого значения метода необходимо поместить ключевое слово abstract. Абстрактные методы не имеют реализации, поэтому определение такого метода заканчивается точкой с запятой вместо обычного блока метода. Классы, производные от абстрактного класса, должны реализовывать все абстрактные методы. Если абстрактный класс наследует виртуальный метод из базового класса, абстрактный класс может переопределить виртуальный метод с помощью абстрактного метода.

Условие задачи 1: в соответствии с вариантом задания самостоятельно разработать класс и программу, иллюстрирующую его возможности – «печатное издание».

Алгоритм решения: Задан объекта класса без использования параметров, вызвана его функция вывода данных на консоль. Пользователем введены переменные и создался объект с параметрами, где использовались его функции.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ConsoleApp5

{

class pub\_print

{

public string name\_print, type\_print, name\_pub, author\_print;

public int pages\_print;

private int year\_print;

public pub\_print()

{

name\_print = "Цветы для Элджернона";

type\_print = "Книга";

name\_pub = "Эксмо";

pages\_print = 382;

year\_print = 2016;

author\_print = "Дэниел Киз";

}

public pub\_print(string name\_print, string type\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

{

this.name\_print = name\_print;

this.type\_print = type\_print;

this.name\_pub = name\_pub;

this.author\_print = author\_print;

this.pages\_print = pages\_print;

this.year\_print = year\_print;

}

public string Name\_print

{

get { return name\_print; }

set { name\_print = value; }

}

public int Pages\_print

{

get { return pages\_print; }

set

{

if ((value >= 150) && (value <= 500))

{

Console.WriteLine("Усредненный размер типа 'книга'.");

}

else if ((value >= 10) && (value <= 20))

{

Console.WriteLine("Усредненный размер типа 'газета'.");

}

}

}

public void print()

{

Console.WriteLine("Печатное издание:\nИздатель: {0}\nТип издания: {2}\nНазвание издания: {1}\nАвтор издания: {5}\nКоличество страниц: {3}\nГод издания: {4}\n", name\_pub, name\_print, type\_print, pages\_print, year\_print, author\_print);

}

public static pub\_print operator ++(pub\_print obj1)

{

obj1.pages\_print += 1;

return obj1;

}

}

public class Program

{

static void Main(string[] args)

{

pub\_print pp = new pub\_print();

pp.print();

string name\_print, type\_print, name\_pub, author\_print;

int pages\_print, year\_print;

Console.Write("Введите издательство: ");

name\_pub = Console.ReadLine();

Console.Write("Введите тип издания: ");

type\_print = Console.ReadLine();

Console.Write("Введите название издания: ");

name\_print = Console.ReadLine();

Console.Write("Введите автора издания: ");

author\_print = Console.ReadLine();

Console.Write("Введите количество страниц: ");

pages\_print = int.Parse(Console.ReadLine());

Console.Write("Введите год издания: ");

year\_print = int.Parse(Console.ReadLine());

pub\_print p\_p = new pub\_print(name\_print, type\_print, name\_pub, author\_print, pages\_print, year\_print);

Console.WriteLine();

p\_p.print();

p\_p.Pages\_print = pages\_print;

p\_p.pages\_print++;

Console.WriteLine("-----------------------------------");

p\_p.print();

Console.ReadKey();

}

}

}

Тестирование приложения:

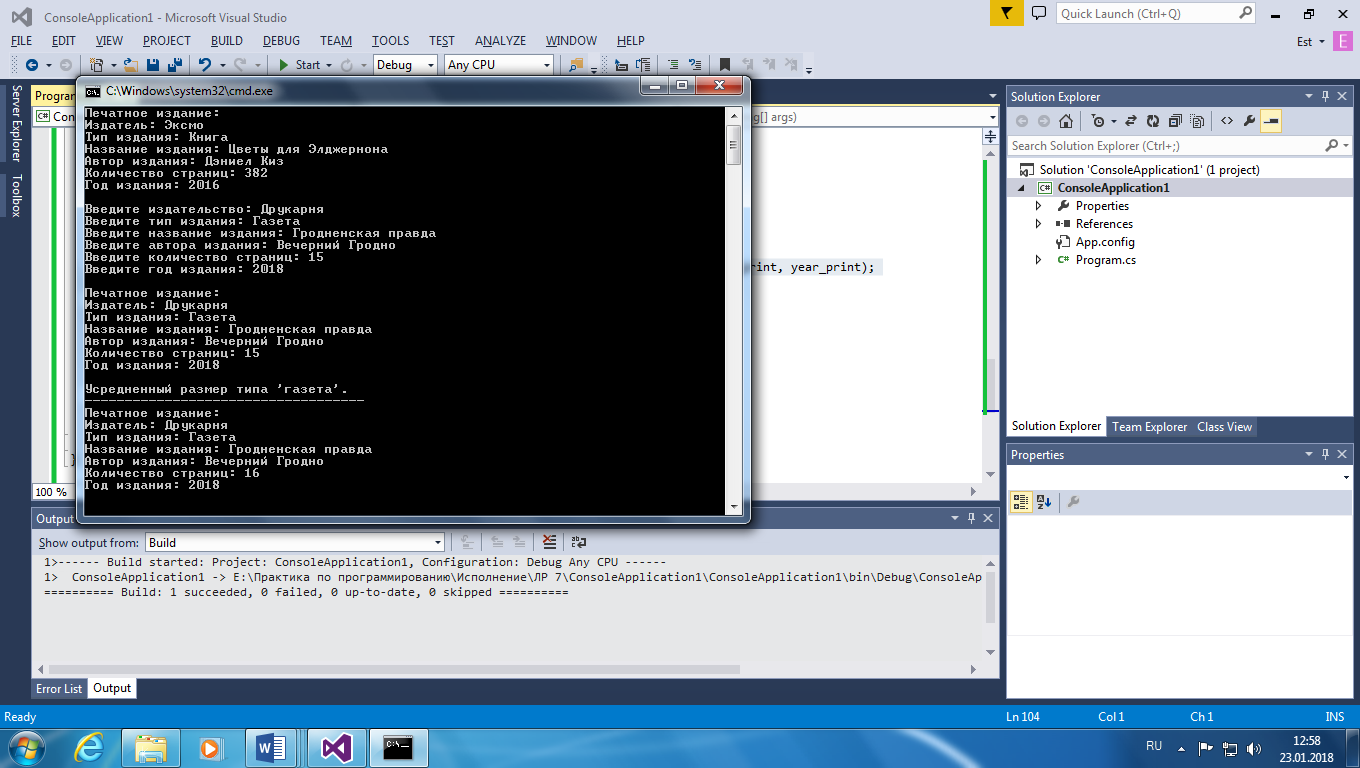


Рисунок 7 – Исполнение программы

Условие задачи 2: построить иерархию классов в соответствии с вариантом задания – «печатное издание, журнал, книга, учебник».

Алгоритм решения: Из файла считано количество объектов, которые затем были инициализированы в массив класса и информация о них вывелась на консоль. После пользователь вручную ввел количество объектов другого класса и сведения о них, после чего результат вывелся на рабочее окно.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.IO;

namespace ConsoleApp5

{

abstract class pub\_print

{

public string name\_print, name\_pub, author\_print;

public int pages\_print, year\_print;

public pub\_print(string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

{

this.name\_print = name\_print;

this.name\_pub = name\_pub;

this.author\_print = author\_print;

this.pages\_print = pages\_print;

this.year\_print = year\_print;

}

public string Name\_print

{

get { return name\_print; }

set { name\_print = value; }

}

public abstract void print();

public int Pages\_print

{

get { return pages\_print; }

set

{

if ((value >= 150) && (value <= 500))

{

Console.WriteLine("Усредненный размер типа 'книга'.");

}

else if ((value >= 10) && (value <= 20))

{

Console.WriteLine("Усредненный размер типа 'газета'.");

}

}

}

public static pub\_print operator ++(pub\_print obj1)

{

obj1.pages\_print += 1;

return obj1;

}

}

class journal : pub\_print

{

public string style;

public journal(string style, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print) :base (name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.style = style;

}

public override void print()

{

Console.WriteLine("Печатное издание: журнал;\nИздатель: {0};\nНазвание журнала: {1};\nАвтор: {4};\nНаправление журнала: {5};\nКоличество страниц: {2};\nГод издания: {3}.\n", name\_pub, name\_print, pages\_print, year\_print, author\_print, style);

}

}

class book : pub\_print

{

public string binding;

public book(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print) :base (name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.binding = binding;

}

public override void print()

{ }

public void printer()

{

Console.WriteLine("Печатное издание: книга;\nИздатель: {0};\nНазвание: {1};\nАвтор: {4};\nКоличество страниц: {2};\nГод издания: {3};", name\_pub, name\_print, pages\_print, year\_print, author\_print);

Console.WriteLine("Переплёт: {0};", binding);

}

}

class textbook : book

{

public string discipline;

public textbook(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print) :base (binding, name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

discipline = "Биология";

}

public override void print()

{

printer();

Console.WriteLine("Дисциплина: {0}.\n", discipline);

}

}

public class Program

{

static void Main(string[] args)

{

string b, n\_print, n\_pub, a\_print;

int N, p\_print, y\_print;

StreamReader sr = new StreamReader("text.txt");

N = int.Parse(sr.ReadLine());

textbook[] t = new textbook[N];

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

{

b = sr.ReadLine();

n\_print = sr.ReadLine();

n\_pub = sr.ReadLine();

a\_print = sr.ReadLine();

p\_print = int.Parse(sr.ReadLine());

y\_print = int.Parse(sr.ReadLine());

t[i] = new textbook(b, n\_print, n\_pub, a\_print, p\_print, y\_print);

}

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

t[i].print();

textbook tb = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

tb.print();

Console.WriteLine("Введите количество заполняемых журналов: ");

int n = int.Parse(Console.ReadLine());

journal[] j = new journal[n];

string name\_print, name\_pub, author\_print, style;

int pages\_print, year\_print;

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

{

Console.Write("Введите издательство: ");

name\_pub = Console.ReadLine();

Console.Write("Введите название издания: ");

name\_print = Console.ReadLine();

Console.Write("Введите автора издания: ");

author\_print = Console.ReadLine();

Console.Write("Введите количество страниц: ");

pages\_print = int.Parse(Console.ReadLine());

Console.Write("Введите год издания: ");

year\_print = int.Parse(Console.ReadLine());

Console.Write("Введите направление журнала: ");

style = Console.ReadLine();

j[i] = new journal(style, name\_print, name\_pub, author\_print, pages\_print, year\_print);

Console.WriteLine();

j[0].Pages\_print = pages\_print;

}

j[0].pages\_print++;

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

{

j[i].print();

}

Console.ReadKey();

}

}

}

Тестирование приложения:

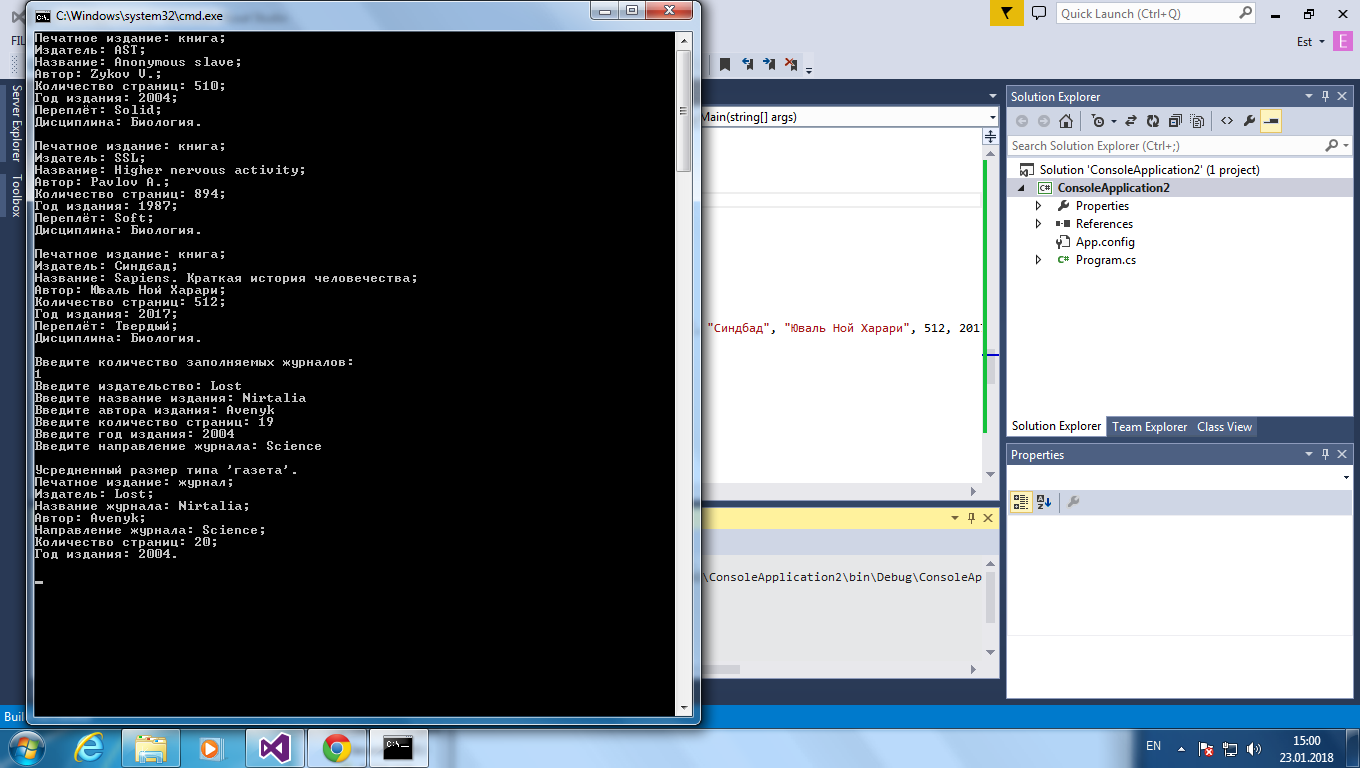


Рисунок 8 – Исполнение программы

1.8 Решение задания с использованием делегатов и событий

Делегат – это объект, имеющий ссылку на метод. Делегат позволяет выбрать вызываемый метод во время выполнения программы. Фактически значение делегата – это адрес области памяти, где находится точка входа метода.

Делегат позволяет указать в коде программы вызов метода, но фактически вызываемый метод определяется во время работы программы, а не во время компилирования.

Делегат объявляется с помощью ключевого слова delegate, за которым указывается тип возвращаемого значения, имя делегата и список параметров вызываемых методов. Примеры объявления классов делегатов:

delegate int ClassDelegate(int key);

delegate void XXX(int intKey, float fKey);

Характерной особенностью делегата является возможность его использования для вызова любого метода, который соответствует сигнатуре делегата. Это дает возможность определить во время выполнения программы, какой из методов должен быть вызван. Вызываемый метод может быть методом экземпляра, ассоциированным с объектом, либо статическим методом, ассоциированным с классом. Метод можно вызвать только тогда, когда его сигнатура (тип возвращаемого значения, набор параметров) соответствует сигнатуре делегата.

Условие задачи: в соответствии с вариантом задания самостоятельно разработать класс и программу, иллюстрирующую его возможности – «печатное издание, журнал, книга, учебник».

Алгоритм решения: Произошла инициализация делегата по статической целочисленной функции, его вызов. Объявлены объекты классов, на которых использовано событие и подписка, затем его вызов, отписка и повторный вызов. Дальнейший алгоритм повторяет предыдущую задачу.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.IO;

namespace ConsoleApp5

{

public delegate void info\_print();

abstract class pub\_print

{

public string name\_print, name\_pub, author\_print;

public int pages\_print, year\_print;

public pub\_print(string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

{

this.name\_print = name\_print;

this.name\_pub = name\_pub;

this.author\_print = author\_print;

this.pages\_print = pages\_print;

this.year\_print = year\_print;

}

public string Name\_print

{

get { return name\_print; }

set { name\_print = value; }

}

public abstract void print();

public int Pages\_print

{

get { return pages\_print; }

set

{

if ((value >= 150) && (value <= 500))

{

Console.WriteLine("Усредненный размер типа 'книга'.");

}

else if ((value >= 10) && (value <= 20))

{

Console.WriteLine("Усредненный размер типа 'газета'.");

}

}

}

public static pub\_print operator ++(pub\_print obj1)

{

obj1.pages\_print += 1;

return obj1;

}

}

class journal : pub\_print

{

public string style;

public journal(string style, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print) :base (name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.style = style;

}

public override void print()

{

Console.WriteLine("Печатное издание: журнал;\nИздатель: {0};\nНазвание журнала: {1};\nАвтор: {4};\nНаправление журнала: {5};\nКоличество страниц: {2};\nГод издания: {3}.\n", name\_pub, name\_print, pages\_print, year\_print, author\_print, style);

}

public void journal\_e\_p(string spam)

{

Console.WriteLine("Обработчик события {0} вызван.", spam);

}

}

class book : pub\_print

{

public event info\_print event\_print;

public string binding;

public book(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.binding = binding;

}

public override void print() { }

public void printer()

{

Console.WriteLine("Печатное издание: книга;\nИздатель: {0};\nНазвание: {1};\nАвтор: {4};\nКоличество страниц: {2};\nГод издания: {3};", name\_pub, name\_print, pages\_print, year\_print, author\_print);

Console.WriteLine("Переплёт: {0};", binding);

}

public void events\_now()

{

event\_print();

}

public void events\_job()

{

Console.WriteLine("Обработчик события книги.");

}

}

class textbook : book

{

public string discipline;

public textbook(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(binding, name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

discipline = "Биология";

}

public override void print()

{

printer();

Console.WriteLine("Дисциплина: {0}.\n", discipline);

}

public void textbook\_event()

{

Console.WriteLine("Обработчик события учебника.");

}

}

public class Program

{

public delegate int Del(DateTime DT);

static int on\_date\_time(DateTime dt)

{

return dt.Day;

}

static void Main(string[] args)

{

DateTime dt = new DateTime(2018, 01, 24);

Del d = new Del(on\_date\_time);

Console.WriteLine("Делегат создан {0} числа", d(dt));

Console.WriteLine();

book bk = new book("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

textbook tbk = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

Console.WriteLine("Подписка на события книги и учебника активирована: ");

bk.event\_print += new info\_print(bk.events\_job);

bk.event\_print += new info\_print(tbk.textbook\_event);

bk.events\_now();

Console.WriteLine();

Console.WriteLine("Подписка на событие учебника отменена: ");

bk.event\_print -= new info\_print(tbk.textbook\_event);

bk.events\_now();

Console.WriteLine();

string b, n\_print, n\_pub, a\_print;

int N, p\_print, y\_print;

StreamReader sr = new StreamReader("text.txt");

N = int.Parse(sr.ReadLine());

textbook[] t = new textbook[N];

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

{

b = sr.ReadLine();

n\_print = sr.ReadLine();

n\_pub = sr.ReadLine();

a\_print = sr.ReadLine();

p\_print = int.Parse(sr.ReadLine());

y\_print = int.Parse(sr.ReadLine());

t[i] = new textbook(b, n\_print, n\_pub, a\_print, p\_print, y\_print);

}

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

t[i].print();

textbook tb = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

tb.print();

Console.WriteLine("Введите количество заполняемых журналов: ");

int n = int.Parse(Console.ReadLine());

journal[] j = new journal[n];

string name\_print, name\_pub, author\_print, style;

int pages\_print, year\_print;

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

{

Console.Write("Введите издательство: ");

name\_pub = Console.ReadLine();

Console.Write("Введите название издания: ");

name\_print = Console.ReadLine();

Console.Write("Введите автора издания: ");

author\_print = Console.ReadLine();

Console.Write("Введите количество страниц: ");

pages\_print = int.Parse(Console.ReadLine());

Console.Write("Введите год издания: ");

year\_print = int.Parse(Console.ReadLine());

Console.Write("Введите направление журнала: ");

style = Console.ReadLine();

j[i] = new journal(style, name\_print, name\_pub, author\_print, pages\_print, year\_print);

Console.WriteLine();

j[0].Pages\_print = pages\_print;

}

j[0].pages\_print++;

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

{

j[i].print();

}

Console.ReadKey();

}

}

}

Тестирование приложения:

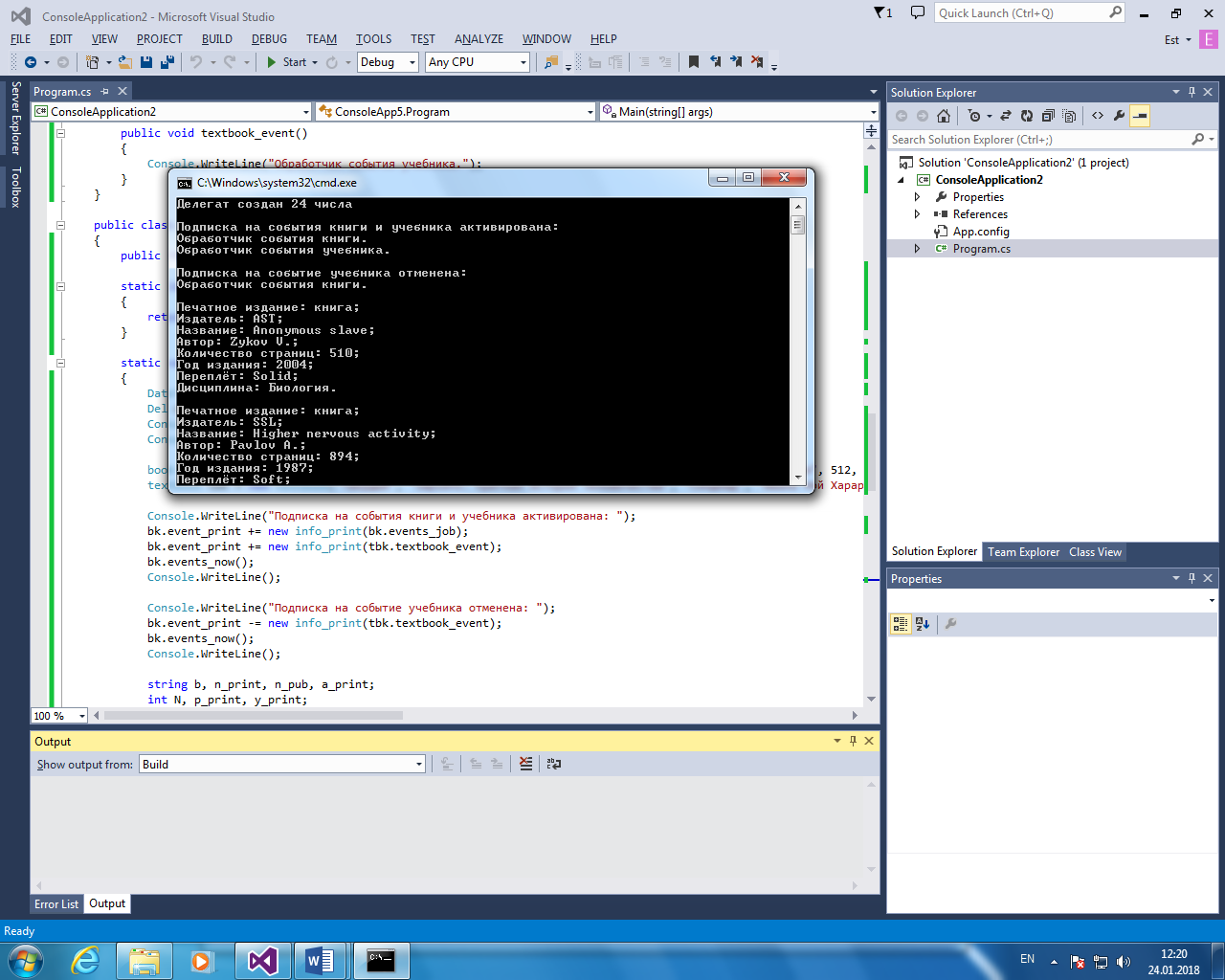


Рисунок 9 – Исполнение программы

1.9 Решение задания с использованием асинхронных делегатов и многопоточности

С помощью языка C# можно создавать приложения, которые выполняют несколько задач одновременно. Задачи, которые потенциально могут задержать выполнение других задач, выполняются в отдельных потоках; такой способ организации работы приложения называется многопоточностью или свободным созданием потоков.

Приложения, использующие многопоточность, более оперативно реагируют на действия пользователя, поскольку пользовательский интерфейс остается активным, в то время как задачи, требующие интенсивной работы процессора, выполняются в других потоках. Многопоточность также эффективна при создании масштабируемых приложений, поскольку пользователь может добавлять потоки при увеличении рабочей нагрузки.

Основной функционал для использования потоков в приложении сосредоточен в пространстве имен System.Threading. В нем определен класс, представляющий отдельный поток - класс Thread.

Класс Thread определяет ряд методов и свойств, которые позволяют управлять потоком и получать информацию о нем.

Асинхронные делегаты позволяют вызывать методы, на которые эти делегаты указывают, в асинхронном режиме. В теме проделегаты говорилось, что делегаты могут вызываться как с помощью метода Invoke, так и в асинхронном режиме с помощью пары методов BeginInvoke/EndInvoke.

Условие задачи: На основе предыдущего задания продемонстрируйте многопоточность приложения. Добавьте асинхронный делегат и продемонстрируйте его работу – «печатное издание, журнал, книга, учебник».

Алгоритм решения: Произошла инициализация делегата по статической целочисленной функции, его вызов. Объявлены объекты классов, на которых использовано событие и подписка, затем его вызов, отписка и повторный вызов. Дальнейший алгоритм повторяет предыдущую задачу.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.IO;

using System.Threading;

namespace ConsoleApp5

{

public delegate void info\_print();

public delegate void asin\_deleg();

abstract class pub\_print

{

public string name\_print, name\_pub, author\_print;

public int pages\_print, year\_print;

public pub\_print(string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

{

this.name\_print = name\_print;

this.name\_pub = name\_pub;

this.author\_print = author\_print;

this.pages\_print = pages\_print;

this.year\_print = year\_print;

}

public string Name\_print

{

get { return name\_print; }

set { name\_print = value; }

}

public abstract void print();

public int Pages\_print

{

get { return pages\_print; }

set

{

if ((value >= 150) && (value <= 500))

{

Console.WriteLine("Усредненный размер типа 'книга'.");

}

else if ((value >= 10) && (value <= 20))

{

Console.WriteLine("Усредненный размер типа 'газета'.");

}

}

}

public static pub\_print operator ++(pub\_print obj1)

{

obj1.pages\_print += 1;

return obj1;

}

}

class journal : pub\_print

{

public string style;

public journal(string style, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.style = style;

}

public override void print()

{

Console.WriteLine("Печатное издание: журнал;\nИздатель: {0};\nНазвание журнала: {1};\nАвтор: {4};\nНаправление журнала: {5};\nКоличество страниц: {2};\nГод издания: {3}.\n", name\_pub, name\_print, pages\_print, year\_print, author\_print, style);

}

public void journal\_e\_p(string spam)

{

Console.WriteLine("Обработчик события {0} вызван.", spam);

}

}

class book : pub\_print

{

public event info\_print event\_print;

public string binding;

public book(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.binding = binding;

}

public override void print() { }

public void printer()

{

Console.WriteLine("Печатное издание: книга;\nИздатель: {0};\nНазвание: {1};\nАвтор: {4};\nКоличество страниц: {2};\nГод издания: {3};", name\_pub, name\_print, pages\_print, year\_print, author\_print);

Console.WriteLine("Переплёт: {0};", binding);

}

public void events\_now()

{

event\_print();

}

public void events\_job()

{

Console.WriteLine("Обработчик события книги.");

}

}

class textbook : book

{

public string discipline;

public textbook(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(binding, name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

discipline = "Биология";

}

public override void print()

{

printer();

Console.WriteLine("Дисциплина: {0}.\n", discipline);

}

public void textbook\_event()

{

Console.WriteLine("Обработчик события учебника.");

}

}

public class Program

{

public delegate int Del(DateTime DT);

static int on\_date\_time(DateTime dt)

{

return dt.Day;

}

static void ThreadFunction(Object input)

{

int flag = (int)input;

if (flag == 1)

{

Console.WriteLine("Это первый поток!");

}

else

{

Console.WriteLine("Это второй поток!");

}

}

static void SpamFunction()

{

int count = 12;

while (count > 0)

{

Console.Write(" -Spam- ");

Thread.Sleep(50);

--count;

}

}

static void EggsFunction()

{

for (int i = 0; i < 7; i++)

{

Console.Write("Eggs! ");

Thread.Sleep(15);

}

}

static void Main(string[] args)

{

asin\_deleg a\_d = SpamFunction;

IAsyncResult ar = a\_d.BeginInvoke(null, null);

while (!ar.IsCompleted)

{

Console.Write("...");

Thread.Sleep(100);

}

Thread thread1 = new Thread(ThreadFunction);

thread1.Start(1);

Thread thread2 = new Thread(SpamFunction);

thread2.Start();

Thread thread3 = new Thread(EggsFunction);

thread3.Start();

Thread thread4 = new Thread(ThreadFunction);

thread4.Start(2);

Thread th = Thread.CurrentThread;

th.Name = "Метод Main";

Console.WriteLine("Имя потока: {0}", th.Name);

Console.WriteLine("Запущен ли поток: {0}", th.IsAlive);

Console.WriteLine("Приоритет потока: {0}", th.Priority);

Console.WriteLine("Статус потока: {0}", th.ThreadState);

Console.WriteLine("Домен приложения: {0}", Thread.GetDomain().FriendlyName);

Console.WriteLine();

DateTime dt = new DateTime(2018, 01, 24);

Del d = new Del(on\_date\_time);

Console.WriteLine("Делегат создан {0} числа", d(dt));

Console.WriteLine();

book bk = new book("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

textbook tbk = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

Console.WriteLine("Подписка на события книги и учебника активирована: ");

bk.event\_print += new info\_print(bk.events\_job);

bk.event\_print += new info\_print(tbk.textbook\_event);

bk.events\_now();

Console.WriteLine();

Console.WriteLine("Подписка на событие учебника отменена: ");

bk.event\_print -= new info\_print(tbk.textbook\_event);

bk.events\_now();

Console.WriteLine();

string b, n\_print, n\_pub, a\_print;

int N, p\_print, y\_print;

StreamReader sr = new StreamReader("text.txt");

N = int.Parse(sr.ReadLine());

textbook[] t = new textbook[N];

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

{

b = sr.ReadLine();

n\_print = sr.ReadLine();

n\_pub = sr.ReadLine();

a\_print = sr.ReadLine();

p\_print = int.Parse(sr.ReadLine());

y\_print = int.Parse(sr.ReadLine());

t[i] = new textbook(b, n\_print, n\_pub, a\_print, p\_print, y\_print);

}

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

t[i].print();

textbook tb = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

tb.print();

Console.WriteLine("Введите количество заполняемых журналов: ");

int n = int.Parse(Console.ReadLine());

journal[] j = new journal[n];

string name\_print, name\_pub, author\_print, style;

int pages\_print, year\_print;

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

{

Console.Write("Введите издательство: ");

name\_pub = Console.ReadLine();

Console.Write("Введите название издания: ");

name\_print = Console.ReadLine();

Console.Write("Введите автора издания: ");

author\_print = Console.ReadLine();

Console.Write("Введите количество страниц: ");

pages\_print = int.Parse(Console.ReadLine());

Console.Write("Введите год издания: ");

year\_print = int.Parse(Console.ReadLine());

Console.Write("Введите направление журнала: ");

style = Console.ReadLine();

j[i] = new journal(style, name\_print, name\_pub, author\_print, pages\_print, year\_print);

Console.WriteLine();

j[0].Pages\_print = pages\_print;

}

j[0].pages\_print++;

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

{

j[i].print();

}

Console.ReadKey();

}

}

}

Тестирование приложения:

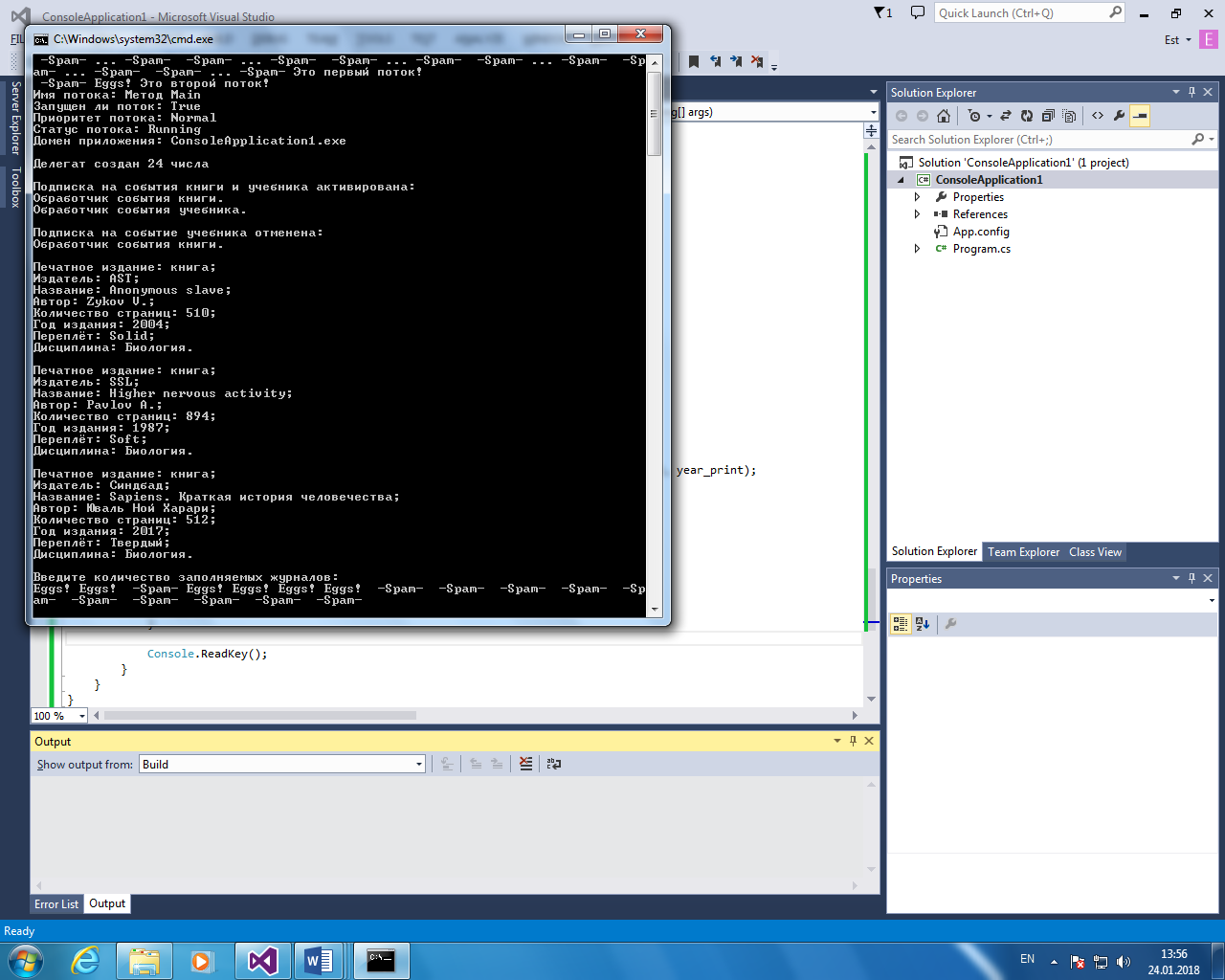


Рисунок 10 – Исполнение программы

1.10 Программирование обобщенных классов

Язык С# позволяет создавать собственные классы-прототипы и их разновидности – интерфейсы, структуры, делегаты, события.

Обобщенные классы могут входить в иерархию классов аналогично необобщенным классам. Следовательно, обобщенный класс может действовать как базовый или производный класс. Главное отличие между иерархиями обобщенных и необобщенных классов заключается в том, что в первом случае аргументы типа, необходимые обобщенному базовому классу, должны передаваться всеми производными классами вверх по иерархии аналогично передаче аргументов конструктора.

Помимо обобщенных классов и методов, в C# допускаются обобщенные интерфей­сы. Такие интерфейсы указываются аналогично обобщенным классам.

Иногда требуется иметь отдельный метод, параметризованный каким-либо типом данных. Примером такого метода может служить метод сортировки данных.

Как и методы, делегаты также могут быть обобщенными. Ниже приведена общая форма объявления обобщенного делегата.

delegate возвращаемый\_тип имя\_делегата<список\_параметров\_типа>(список\_аргументов);

Условие задачи: На основе предыдущего задания создайте обобщенный класс, обобщенный метод, обобщенный интерфейс и обобщенный делегат – «печатное издание, журнал, книга, учебник».

Алгоритм решения: Произошла инициализация обобщенного делегата по статической целочисленной функции, его вызов. Объявлен целочисленный объект типа int32 обобщенного класса, реализующего обобщенный интерфейс; вызвана его функция. Создан список строк, выведенный на консоль; использован обобщенный метод сортировки списка и его повторное отображение на экране. Дальнейший алгоритм повторяет предыдущую задачу.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.IO;

using System.Threading;

namespace ConsoleApp5

{

public delegate void info\_print();

public delegate void asin\_deleg();

abstract class pub\_print

{

public string name\_print, name\_pub, author\_print;

public int pages\_print, year\_print;

public pub\_print(string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

{

this.name\_print = name\_print;

this.name\_pub = name\_pub;

this.author\_print = author\_print;

this.pages\_print = pages\_print;

this.year\_print = year\_print;

}

public string Name\_print

{

get { return name\_print; }

set { name\_print = value; }

}

public abstract void print();

public int Pages\_print

{

get { return pages\_print; }

set

{

if ((value >= 150) && (value <= 500))

{

Console.WriteLine("Усредненный размер типа 'книга'.");

}

else if ((value >= 10) && (value <= 20))

{

Console.WriteLine("Усредненный размер типа 'газета'.");

}

}

}

public static pub\_print operator ++(pub\_print obj1)

{

obj1.pages\_print += 1;

return obj1;

}

}

class journal : pub\_print

{

public string style;

public journal(string style, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.style = style;

}

public override void print()

{

Console.WriteLine("Печатное издание: журнал;\nИздатель: {0};\nНазвание журнала: {1};\nАвтор: {4};\nНаправление журнала: {5};\nКоличество страниц: {2};\nГод издания: {3}.\n", name\_pub, name\_print, pages\_print, year\_print, author\_print, style);

}

public void journal\_e\_p(string spam)

{

Console.WriteLine("Обработчик события {0} вызван.", spam);

}

}

class book : pub\_print

{

public event info\_print event\_print;

public string binding;

public book(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

this.binding = binding;

}

public override void print()

{ }

public void printer()

{

Console.WriteLine("Печатное издание: книга;\nИздатель: {0};\nНазвание: {1};\nАвтор: {4};\nКоличество страниц: {2};\nГод издания: {3};", name\_pub, name\_print, pages\_print, year\_print, author\_print);

Console.WriteLine("Переплёт: {0};", binding);

}

public void events\_now()

{

event\_print();

}

public void events\_job()

{

Console.WriteLine("Обработчик события книги.");

}

}

class textbook : book

{

public string discipline;

public textbook(string binding, string name\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

: base(binding, name\_print, name\_pub, author\_print, pages\_print, year\_print)

{

discipline = "Биология";

}

public override void print()

{

printer();

Console.WriteLine("Дисциплина: {0}.\n", discipline);

}

public void textbook\_event()

{

Console.WriteLine("Обработчик события учебника.");

}

}

public interface ISort<T>

where T : struct

{

void ReWrite();

}

class MyObj<T> : ISort<T> where T : struct

{

public int longOb { get; set; }

T[] myarr;

public MyObj(int i)

{

longOb = i;

}

public MyObj(int i, T[] arr)

{

longOb = i;

myarr = new T[i];

for (int j = 0; j < arr.Length; j++)

myarr[j] = arr[j];

}

public void ReWrite()

{

Console.WriteLine("Тип: {0}", typeof(T));

Console.WriteLine("Распределение тиражей по городу: ");

foreach (T t in myarr)

Console.Write("{0}; ", t);

Console.WriteLine("\n");

}

public static int SumInt(int a, int b)

{

return a + b;

}

}

delegate T MyDel<T> (T obj1, T obj2);

public class Program

{

public delegate int Del(DateTime DT);

static int on\_date\_time(DateTime dt)

{

return dt.Day;

}

static void ThreadFunction(Object input)

{

int flag = (int)input;

if (flag == 1)

{

Console.WriteLine("Это первый поток!");

}

else

{

Console.WriteLine("Это второй поток!");

}

}

static void SpamFunction()

{

int count = 12;

while (count > 0)

{

Console.Write(" -Spam- ");

Thread.Sleep(50);

--count;

}

}

static void EggsFunction()

{

for (int i = 0; i < 7; i++)

{

Console.Write("Eggs! ");

Thread.Sleep(15);

}

}

static void Sort<T>(ref T[] a) where T : IComparable<T>

{

T buf;

int n = a.Length;

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

{

int im = i;

for (int j = i + 1; j < n; j++)

if (a[j].CompareTo(a[im]) < 0) im = j;

buf = a[i]; a[i] = a[im]; a[im] = buf;

}

}

static void Main(string[] args)

{

MyDel<int> del1 = MyObj<int>.SumInt;

Console.WriteLine("Количество художественных и научных журналов проданных за январь = " + del1(136, 74));

int[] MyArrByte = new int[6] { 41, 50, 183, 566, 87, 65 };

MyObj<int> ByteConst = new MyObj<int>(MyArrByte.Length, MyArrByte);

ByteConst.ReWrite();

Console.WriteLine("Авторы, чьи произведения отправлены в печать в порядке следования: ");

string[] a = { "Сколков", "Форграт", "Мехов", "Зыков", "Норри", "Хох", "Альков" };

foreach (string x in a) Console.Write(x + "; ");

Console.WriteLine();

Console.WriteLine("Упорядочивание авторов в алфавитном порядке: ");

Sort<string>(ref a);

foreach (string x in a) Console.Write(x + "; ");

Console.WriteLine();

Console.WriteLine();

asin\_deleg a\_d = SpamFunction;

IAsyncResult ar = a\_d.BeginInvoke(null, null);

while (!ar.IsCompleted)

{

Console.Write("...");

Thread.Sleep(100);

}

Thread thread1 = new Thread(ThreadFunction);

thread1.Start(1);

Thread thread2 = new Thread(SpamFunction);

thread2.Start();

Thread thread3 = new Thread(EggsFunction);

thread3.Start();

Thread thread4 = new Thread(ThreadFunction);

thread4.Start(2);

Thread th = Thread.CurrentThread;

th.Name = "Метод Main";

Console.WriteLine("Имя потока: {0}", th.Name);

Console.WriteLine("Запущен ли поток: {0}", th.IsAlive);

Console.WriteLine("Приоритет потока: {0}", th.Priority);

Console.WriteLine("Статус потока: {0}", th.ThreadState);

Console.WriteLine("Домен приложения: {0}", Thread.GetDomain().FriendlyName);

Console.WriteLine();

DateTime dt = new DateTime(2018, 01, 24);

Del d = new Del(on\_date\_time);

Console.WriteLine("Делегат создан {0} числа", d(dt));

Console.WriteLine();

book bk = new book("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

textbook tbk = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

Console.WriteLine("Подписка на события книги и учебника активирована: ");

bk.event\_print += new info\_print(bk.events\_job);

bk.event\_print += new info\_print(tbk.textbook\_event);

bk.events\_now();

Console.WriteLine();

Console.WriteLine("Подписка на событие учебника отменена: ");

bk.event\_print -= new info\_print(tbk.textbook\_event);

bk.events\_now();

Console.WriteLine();

string b, n\_print, n\_pub, a\_print;

int N, p\_print, y\_print;

StreamReader sr = new StreamReader("text.txt");

N = int.Parse(sr.ReadLine());

textbook[] t = new textbook[N];

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

{

b = sr.ReadLine();

n\_print = sr.ReadLine();

n\_pub = sr.ReadLine();

a\_print = sr.ReadLine();

p\_print = int.Parse(sr.ReadLine());

y\_print = int.Parse(sr.ReadLine());

t[i] = new textbook(b, n\_print, n\_pub, a\_print, p\_print, y\_print);

}

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

t[i].print();

textbook tb = new textbook("Твердый", "Sapiens. Краткая история человечества", "Синдбад", "Юваль Ной Харари", 512, 2017);

tb.print();

Console.WriteLine("Введите количество заполняемых журналов: ");

int n = int.Parse(Console.ReadLine());

journal[] j = new journal[n];

string name\_print, name\_pub, author\_print, style;

int pages\_print, year\_print;

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

{

Console.Write("Введите издательство: ");

name\_pub = Console.ReadLine();

Console.Write("Введите название издания: ");

name\_print = Console.ReadLine();

Console.Write("Введите автора издания: ");

author\_print = Console.ReadLine();

Console.Write("Введите количество страниц: ");

pages\_print = int.Parse(Console.ReadLine());

Console.Write("Введите год издания: ");

year\_print = int.Parse(Console.ReadLine());

Console.Write("Введите направление журнала: ");

style = Console.ReadLine();

j[i] = new journal(style, name\_print, name\_pub, author\_print, pages\_print, year\_print);

Console.WriteLine();

j[0].Pages\_print = pages\_print;

}

j[0].pages\_print++;

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

{

j[i].print();

}

Console.ReadKey();

}

}

}

Тестирование приложения:

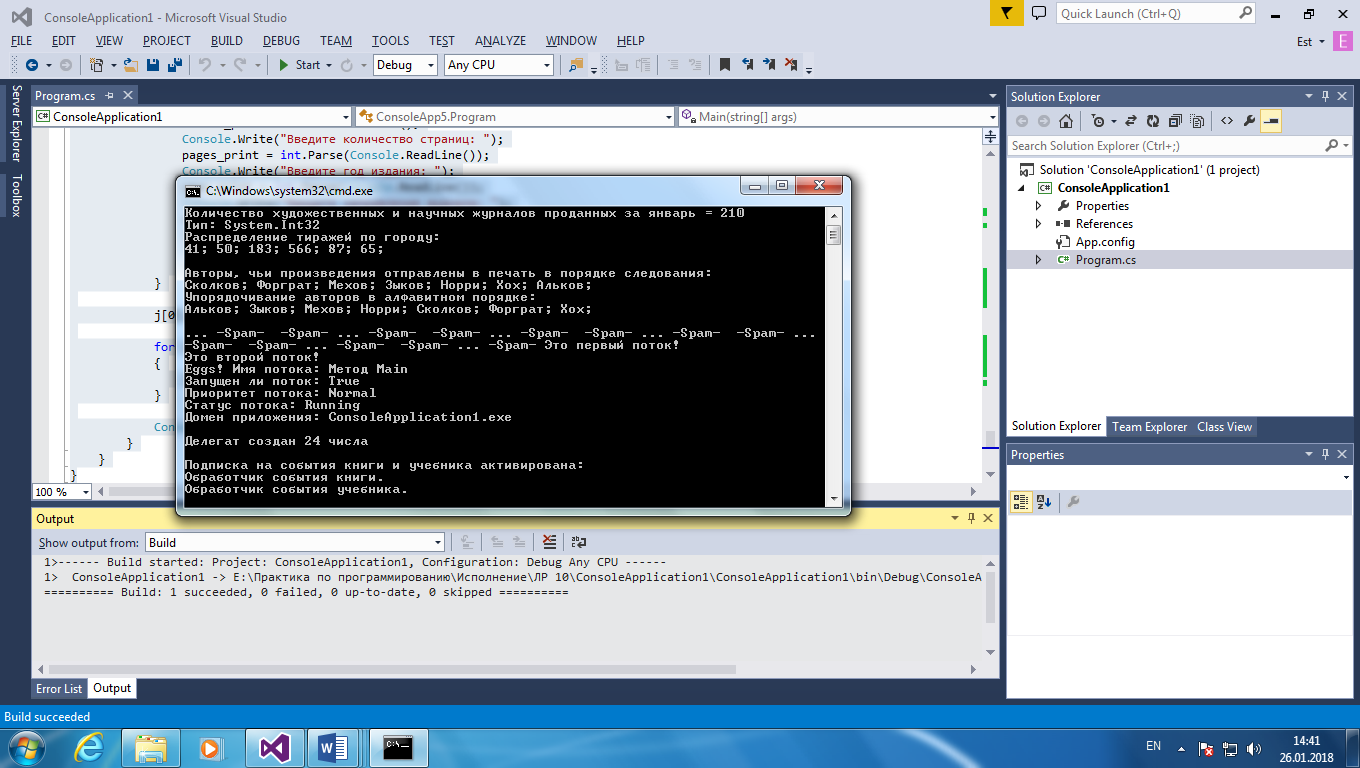


Рисунок 11 – Исполнение программы

1.11 Работа с файлами

Центральную часть потоковой С#-системы занимает класс Stream пространства имен System.IO. Класс Stream представляет байтовый поток и является базовым для всех остальных потоковых классов. Из класса Stream выведены такие байтовые классы потоков как:

* FileStream - байтовый поток, разработанный для файлового ввода-вывода
* BufferedStream - заключает в оболочку байтовый поток и добавляет буферизацию, которая во многих случаях увеличивает производительность программы;
* MemoryStream - байтовый поток, который использует память для хранения данных.

Чтобы создать байтовый поток, связанный с файлом, создается объект класса FileStream. При этом в классе определено несколько конструкторов.

Чтобы создать символьный поток нужно поместить объект класса Stream (например, FileStream ) "внутрь" объекта классаStreamWriter или объекта класса StreamReader. В этом случае байтовый поток будет автоматически преобразовываться в символьный.

Класс StreamWriter предназначен для организации выходного символьного потока.

Условие задачи: Дана последовательность из n вещественных чисел. Записать все эти числа в файл. Вывести на экран все положительные компоненты файла. Дан текстовый файл. Напечатать все строки, длина которых равна данному числу.

Алгоритм решения: Пользователем введен интервал и шаг последовательности. Произошло создание файла и запись в него чисел, которые затем перебором по байтам переведены в вещественный тип и, при удовлетворении условия на положительность, вывелись на консоль. При выполнении следующего задания введено число, затем файл был открыт и циклом при совпавшем введенном числе с длиной строки выводится она выводилась на экран.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите начальную позицию последовательности: ");

double begin\_num = double.Parse(Console.ReadLine());

Console.WriteLine("Введите конечную позицию последовательности: ");

double end\_num = double.Parse(Console.ReadLine());

Console.WriteLine("Введите шаг последовательности: ");

double step\_num = double.Parse(Console.ReadLine());

FileStream f = new FileStream("Numbers\_text.txt", FileMode.Create);

BinaryWriter fOut = new BinaryWriter(f);

for (double i = begin\_num; i <= end\_num; i += step\_num)

{

fOut.Write(i);

}

fOut.Close();

Console.WriteLine("Вывод из файла: ");

f = new FileStream("Numbers\_text.txt", FileMode.Open);

BinaryReader fIn = new BinaryReader(f);

long m = f.Length; //определяем количество байт в потоке

for (long i = 0; i < m; i += 8)

{

f.Seek(i, SeekOrigin.Begin);

double foo = fIn.ReadDouble();

if (foo > 0)

Console.Write("{0:f2} ", foo);

}

fIn.Close();

f.Close();

Console.WriteLine();

Console.WriteLine(“\*\*\*\*\*”);

Console.WriteLine("Задайте целое число: ");

int len\_num = int.Parse(Console.ReadLine());

Console.WriteLine("Вывод из файла: ");

FileStream fSymb = new FileStream("Symbol\_text.txt", FileMode.Open);

StreamReader fO = new StreamReader(fSymb);

String lines;

while ((lines = fO.ReadLine()) != null)

{

if (lines.Length == len\_num)

Console.WriteLine(lines);

}

}

}

}

Тестирование приложения:

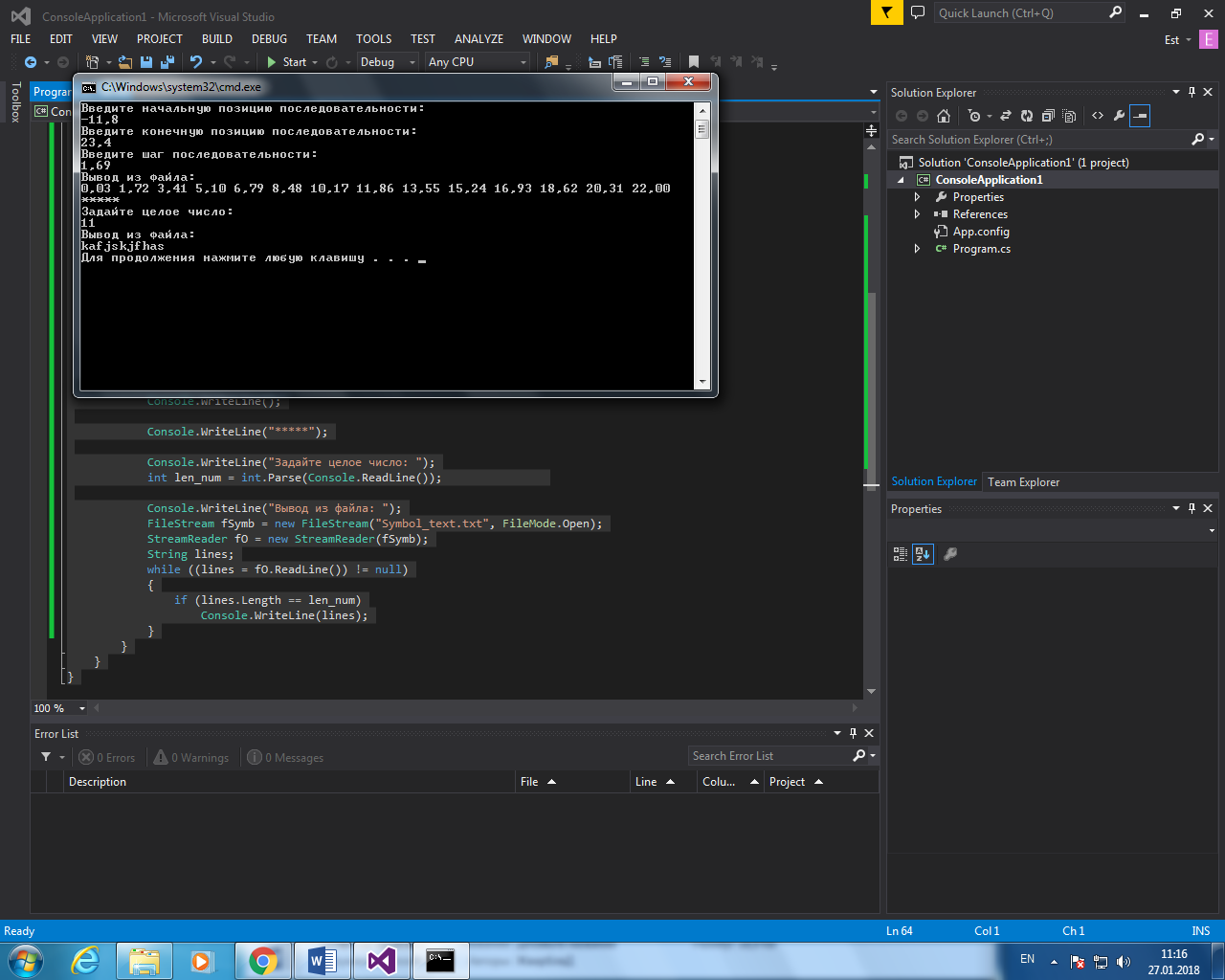


Рисунок 12 – Исполнение программы

1.12 Создание библиотек

В .Net приложения можно развертывать простым копированием всех необходимых сборок в одну папку на жестком диске, но на практике обычно выделяются отдельные подкаталоги для связанного содержимого, например, библиотек. В файлах конфигурации .NET можно указать подкаталоги, в которых среда выполнения будет искать частные сборки при запуске приложения.

В C# определен ряд директив препроцессора, оказывающих влияние на интерпретацию исходного кода программы компилятором. Эти директивы определяют порядок интерпретации текста программы перед ее трансляцией в объектный код в том исходном файле, где они появляются. Все директивы препроцессора начинаются со знака #. Кроме того, каждая директива препроцессора должна быть выделена в отдельную строку кода.

Условие задачи: Создайте из класса, определенного в задании 1 дня 9, библиотеку (сборку). Подпишите ее. Создайте приложение, которое будет использовать Вашу сборку. Настройте ее с помощью конфигурационного файла. Добавьте директивы #pragma, #line.

Алгоритм решения: Задан объекта класса без использования параметров, вызвана его функция вывода данных на консоль. Пользователем введены переменные и создался объект с параметрами, где использовались его функции.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

#line hidden

ClassPrintPub1.pub\_print pp = new ClassPrintPub1.pub\_print();

pp.print();

string name\_print, type\_print, name\_pub, author\_print;

int pages\_print, year\_print;

#pragma warning disable 168

int war;

Console.Write("Введите издательство: ");

name\_pub = Console.ReadLine();

Console.Write("Введите тип издания: ");

type\_print = Console.ReadLine();

Console.Write("Введите название издания: ");

name\_print = Console.ReadLine();

Console.Write("Введите автора издания: ");

author\_print = Console.ReadLine();

Console.Write("Введите количество страниц: ");

pages\_print = int.Parse(Console.ReadLine());

Console.Write("Введите год издания: ");

year\_print = int.Parse(Console.ReadLine());

ClassPrintPub1.pub\_print p\_p = new ClassPrintPub1.pub\_print(name\_print, type\_print, name\_pub, author\_print, pages\_print, year\_print);

Console.WriteLine();

p\_p.print();

p\_p.Pages\_print = pages\_print;

p\_p.pages\_print++;

Console.WriteLine("-----------------------------------");

p\_p.print();

Console.ReadKey();

}

}

}

Листинг библиотеки:

using System;

namespace ClassPrintPub1

{

public class pub\_print

{

public string name\_print, type\_print, name\_pub, author\_print;

public int pages\_print;

private int year\_print;

public pub\_print()

{

name\_print = "Цветы для Элджернона";

type\_print = "Книга";

name\_pub = "Эксмо";

pages\_print = 382;

year\_print = 2016;

author\_print = "Дэниел Киз";

}

public pub\_print(string name\_print, string type\_print, string name\_pub, string author\_print, int pages\_print, int year\_print)

{

this.name\_print = name\_print;

this.type\_print = type\_print;

this.name\_pub = name\_pub;

this.author\_print = author\_print;

this.pages\_print = pages\_print;

this.year\_print = year\_print;

}

public string Name\_print

{

get { return name\_print; }

set { name\_print = value; }

}

public int Pages\_print

{

get { return pages\_print; }

set

{

if ((value >= 150) && (value <= 500))

{

Console.WriteLine("Усредненный размер типа 'книга'.");

}

else if ((value >= 10) && (value <= 20))

{

Console.WriteLine("Усредненный размер типа 'газета'.");

}

}

}

public void print()

{

Console.WriteLine("Печатное издание:\nИздатель: {0}\nТип издания: {2}\nНазвание издания: {1}\nАвтор издания: {5}\nКоличество страниц: {3}\nГод издания: {4}\n", name\_pub, name\_print, type\_print, pages\_print, year\_print, author\_print);

}

public static pub\_print operator ++(pub\_print obj1)

{

obj1.pages\_print += 1;

return obj1;

}

}

}

Листинг конфигурационного файла:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<runtime>

<assemblyBinding xmlns="urn:schemas‐

microsoft‐com:asm.vl">

<probing privatePath="Libraries"/>

</assemblyBinding>

</runtime>

</configuration>

Тестирование приложения:

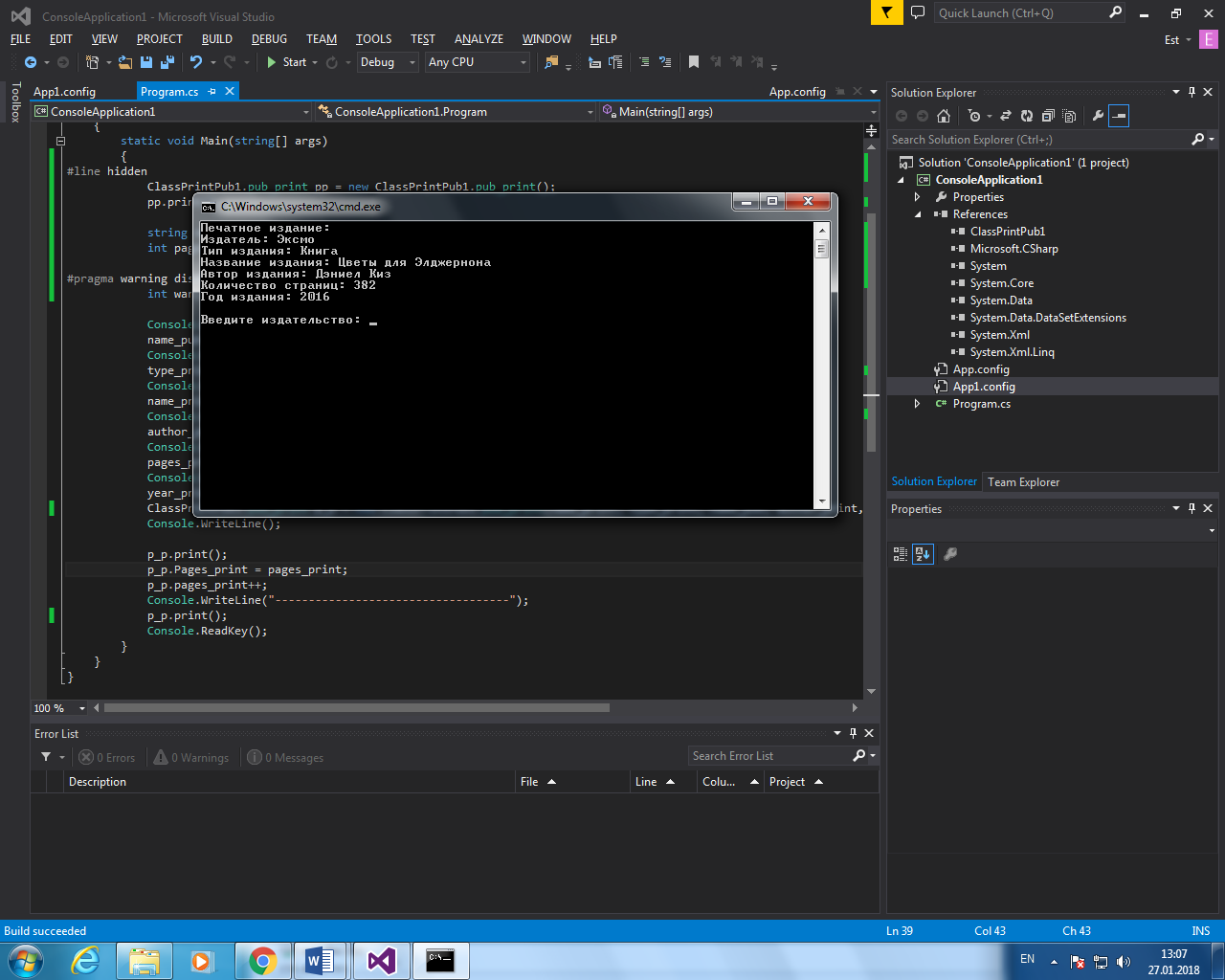


Рисунок 13 – Исполнение программы

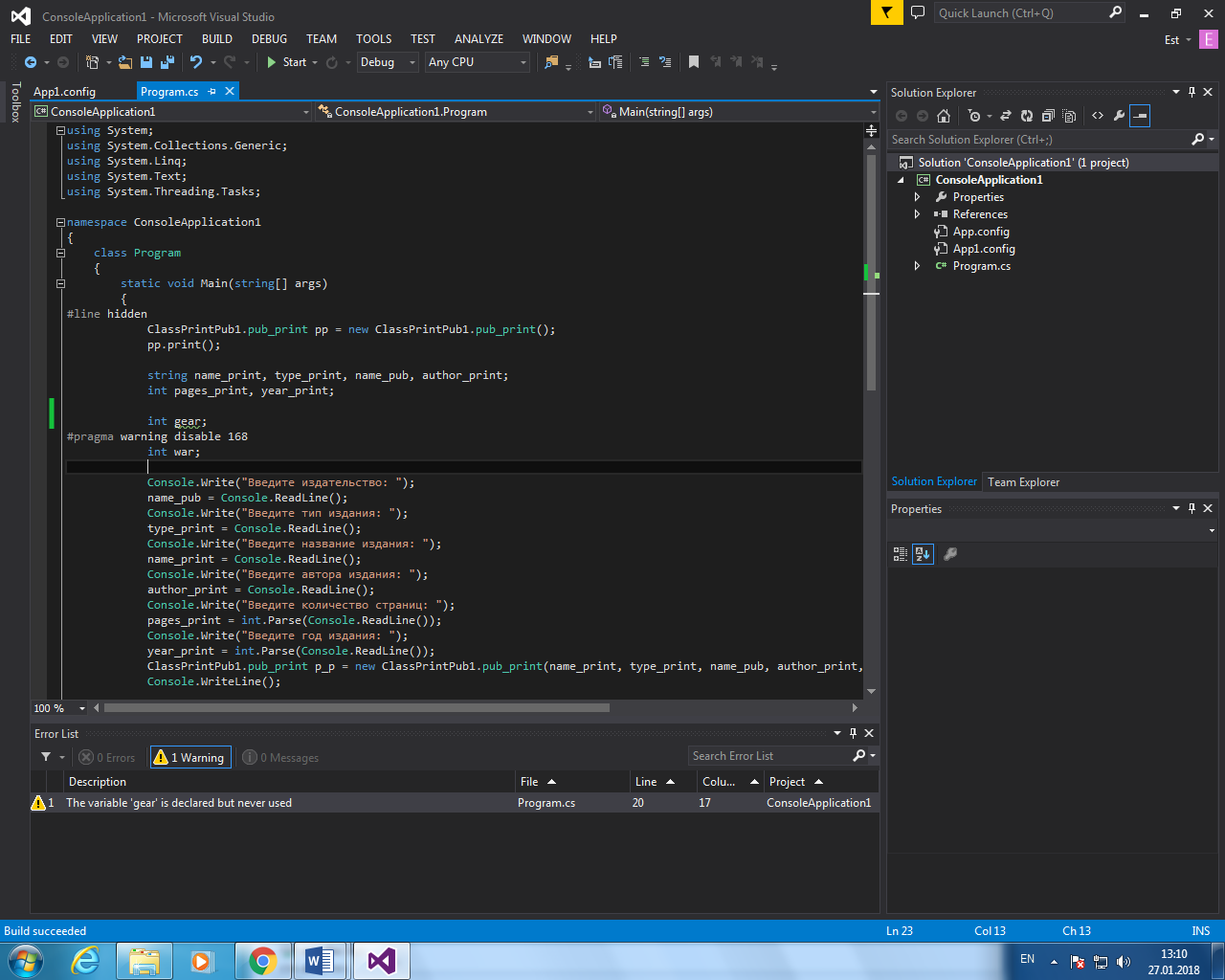


Рисунок 14 – Демонстрация директивы

1.13 Решение задач с помощью структур данных, коллекций и классов-прототипов. Библиотечные классы коллекций

Во многих приложениях требуется создавать группы связанных объектов и управлять ими. Существует два способа группировки объектов: создать массив объектов и создать коллекцию.

Массивы удобнее всего использовать для создания и работы с фиксированным числом строго типизированных объектов.

Коллекции предоставляют более гибкий способ работы с группами объектов. В отличие от массивов, коллекция, с которой вы работаете, может расти или уменьшаться динамически при необходимости. Некоторые коллекции допускают назначение ключа любому объекту, который добавляется в коллекцию, чтобы в дальнейшем можно было быстро извлечь связанный с ключом объект из коллекции.

Коллекция является классом, поэтому необходимо объявить экземпляр класса перед добавлением в коллекцию элементов.

Если коллекция содержит элементы только одного типа данных, можно использовать один из классов в пространстве имен System.Collections.Generic. Универсальная коллекция обеспечивает строгую типизацию, так что в нее нельзя добавить другие типы данных. При извлечении элемента из универсальной коллекции не нужно определять или преобразовывать его тип данных.

Условие задачи: Разработать оконное приложение, используя коллекцию HashSet<T> для данных журнала.

Алгоритм решения: Создан класс, помещенный в коллекцию HashSet. При открытии формы произошла инициализация объектов. Нажатие на кнопку один вызвало добавление записи, считавших информацию из текстовых полей для добавления элемента в коллекцию и DataGrid. Нажатие на кнопку два вызвало сообщение, при нажатии на которую если выражено согласие, произошло удаление элемента из коллекции и DataGrid.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Листинг формы:

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

private DataGridViewColumn dataGridViewColumn1 = null;

private DataGridViewColumn dataGridViewColumn2 = null;

private DataGridViewColumn dataGridViewColumn3 = null;

private DataGridViewColumn dataGridViewColumn4 = null;

private DataGridViewColumn dataGridViewColumn5 = null;

private HashSet<Journal> JournalList = new HashSet<Journal>();

public Form1()

{

InitializeComponent();

initDataGridView();

}

private void addJournal(string number, string name, string redaction, string author, string price)

{

Journal s = new Journal(number, name, redaction, author, price);

JournalList.Add(s);

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

showListInGrid();

}

private void deleteJournal(int elementIndex)

{

var arr = JournalList.ToArray();

JournalList.Remove(arr[elementIndex]);

showListInGrid();

}

private void showListInGrid()

{

dataGridView1.Rows.Clear();

foreach (Journal s in JournalList)

{

DataGridViewRow row = new DataGridViewRow();

DataGridViewTextBoxCell cell1 = new DataGridViewTextBoxCell();

DataGridViewTextBoxCell cell2 = new DataGridViewTextBoxCell();

DataGridViewTextBoxCell cell3 = new DataGridViewTextBoxCell();

DataGridViewTextBoxCell cell4 = new DataGridViewTextBoxCell();

DataGridViewTextBoxCell cell5 = new DataGridViewTextBoxCell();

cell1.ValueType = typeof(string);

cell1.Value = s.getNumber();

cell2.ValueType = typeof(string);

cell2.Value = s.getName();

cell3.ValueType = typeof(string);

cell3.Value = s.getRedaction();

cell4.ValueType = typeof(string);

cell4.Value = s.getAuthor();

cell5.ValueType = typeof(string);

cell5.Value = s.getPrice();

row.Cells.Add(cell1);

row.Cells.Add(cell2);

row.Cells.Add(cell3);

row.Cells.Add(cell4);

row.Cells.Add(cell5);

dataGridView1.Rows.Add(row);

}

}

private void initDataGridView()

{

dataGridView1.DataSource = null;

dataGridView1.Columns.Add(getDataGridViewColumn1());

dataGridView1.Columns.Add(getDataGridViewColumn2());

dataGridView1.Columns.Add(getDataGridViewColumn3());

dataGridView1.Columns.Add(getDataGridViewColumn4());

dataGridView1.Columns.Add(getDataGridViewColumn5());

dataGridView1.AutoResizeColumns();

}

private DataGridViewColumn getDataGridViewColumn1()

{

if (dataGridViewColumn1 == null)

{

dataGridViewColumn1 = new DataGridViewTextBoxColumn();

dataGridViewColumn1.Name = "";

dataGridViewColumn1.HeaderText = "Номер";

dataGridViewColumn1.ValueType = typeof(string);

dataGridViewColumn1.Width = dataGridView1.Width / 3;

}

return dataGridViewColumn1;

}

private DataGridViewColumn getDataGridViewColumn2()

{

if (dataGridViewColumn2 == null)

{

dataGridViewColumn2 = new DataGridViewTextBoxColumn();

dataGridViewColumn2.Name = "";

dataGridViewColumn2.HeaderText = "Название";

dataGridViewColumn2.ValueType = typeof(string);

dataGridViewColumn2.Width = dataGridView1.Width / 3;

}

return dataGridViewColumn2;

}

private DataGridViewColumn getDataGridViewColumn3()

{

if (dataGridViewColumn3 == null)

{

dataGridViewColumn3 = new DataGridViewTextBoxColumn();

dataGridViewColumn3.Name = "";

dataGridViewColumn3.HeaderText = "Редакция";

dataGridViewColumn3.ValueType = typeof(string);

dataGridViewColumn3.Width = dataGridView1.Width / 3;

}

return dataGridViewColumn3;

}

private DataGridViewColumn getDataGridViewColumn4()

{

if (dataGridViewColumn4 == null)

{

dataGridViewColumn4 = new DataGridViewTextBoxColumn();

dataGridViewColumn4.Name = "";

dataGridViewColumn4.HeaderText = "Автор";

dataGridViewColumn4.ValueType = typeof(string);

dataGridViewColumn4.Width = dataGridView1.Width / 3;

}

return dataGridViewColumn4;

}

private DataGridViewColumn getDataGridViewColumn5()

{

if (dataGridViewColumn5 == null)

{

dataGridViewColumn5 = new DataGridViewTextBoxColumn();

dataGridViewColumn5.Name = "";

dataGridViewColumn5.HeaderText = "Цена";

dataGridViewColumn5.ValueType = typeof(string);

dataGridViewColumn5.Width = dataGridView1.Width / 3;

}

return dataGridViewColumn5;

}

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

{

addJournal(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text);

}

private void button2\_Click(object sender, EventArgs e)

{

int selectedRow = dataGridView1.SelectedCells[0].RowIndex;

DialogResult dr = MessageBox.Show("Удалить выделенный журнал?", "",

MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes)

{

try

{

deleteJournal(selectedRow);

}

catch (Exception)

{

}

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

Листинг класса:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WindowsFormsApplication1

{

class Journal

{

public string number;

public string name;

public string redaction;

public string author;

public string price;

public Journal(string number, string name, string redaction, string author, string price)

{

this.number = number;

this.name = name;

this.redaction = redaction;

this.author = author;

this.price = price;

}

public string getNumber()

{

return this.number;

}

public void setNumber(string number)

{

this.number = number;

}

public string getName()

{

return this.name;

}

public void setName(string name)

{

this.name = name;

}

public string getRedaction()

{

return this.redaction;

}

public void setRedaction(string redaction)

{

this.redaction = redaction;

}

public string getAuthor()

{

return this.author;

}

public void setAuthor(string author)

{

this.author = author;

}

public string getPrice()

{

return this.price;

}

public void setPrice(string price)

{

this.price = price;

}

}

}

Тестирование приложения:

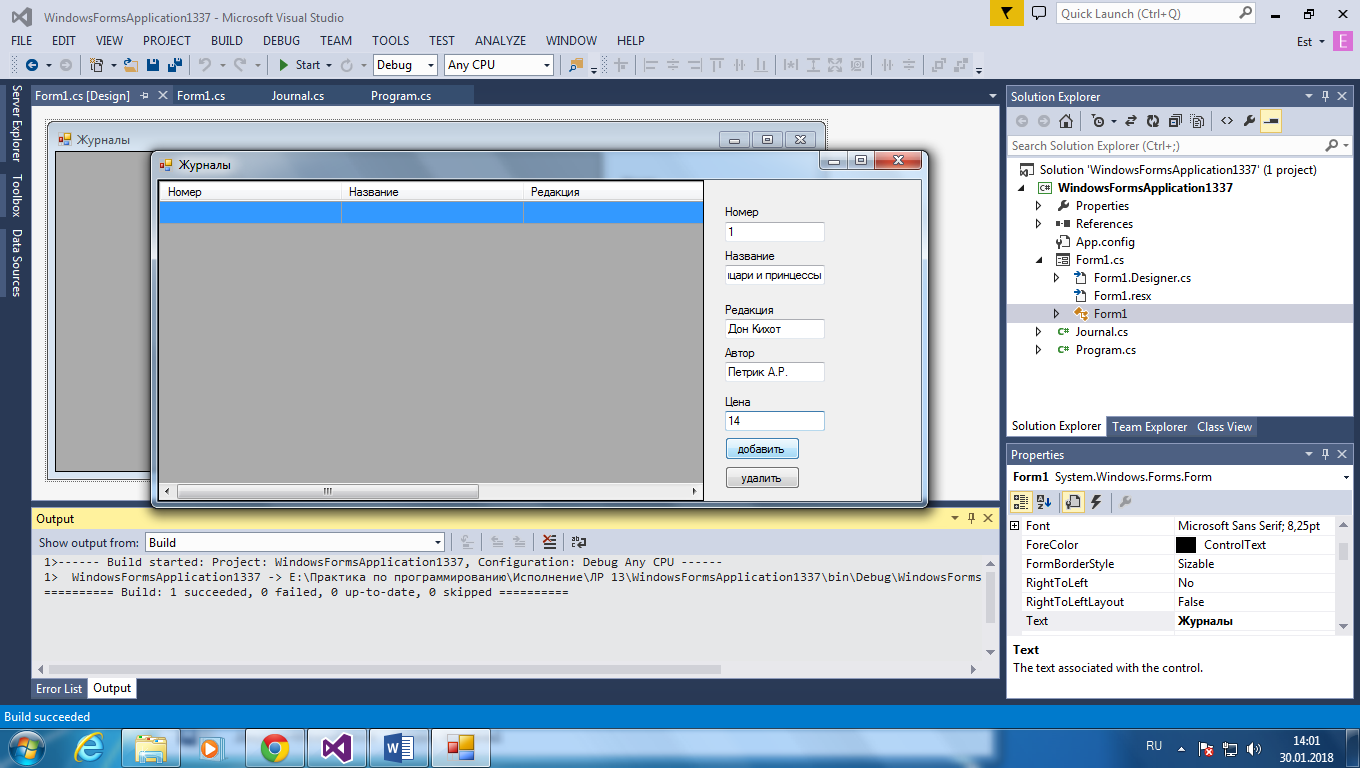


Рисунок 15 – Заполненные данные

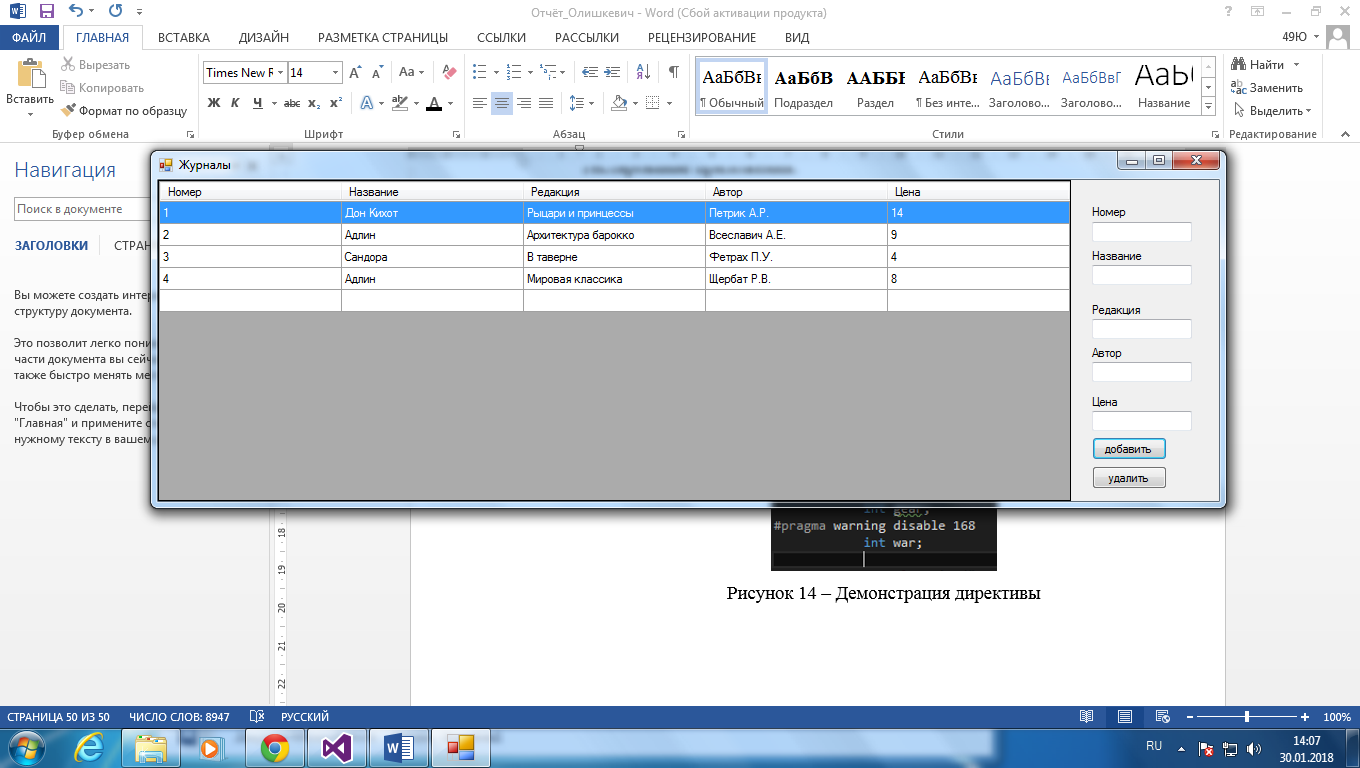


Рисунок 16 – Демонстрация данных

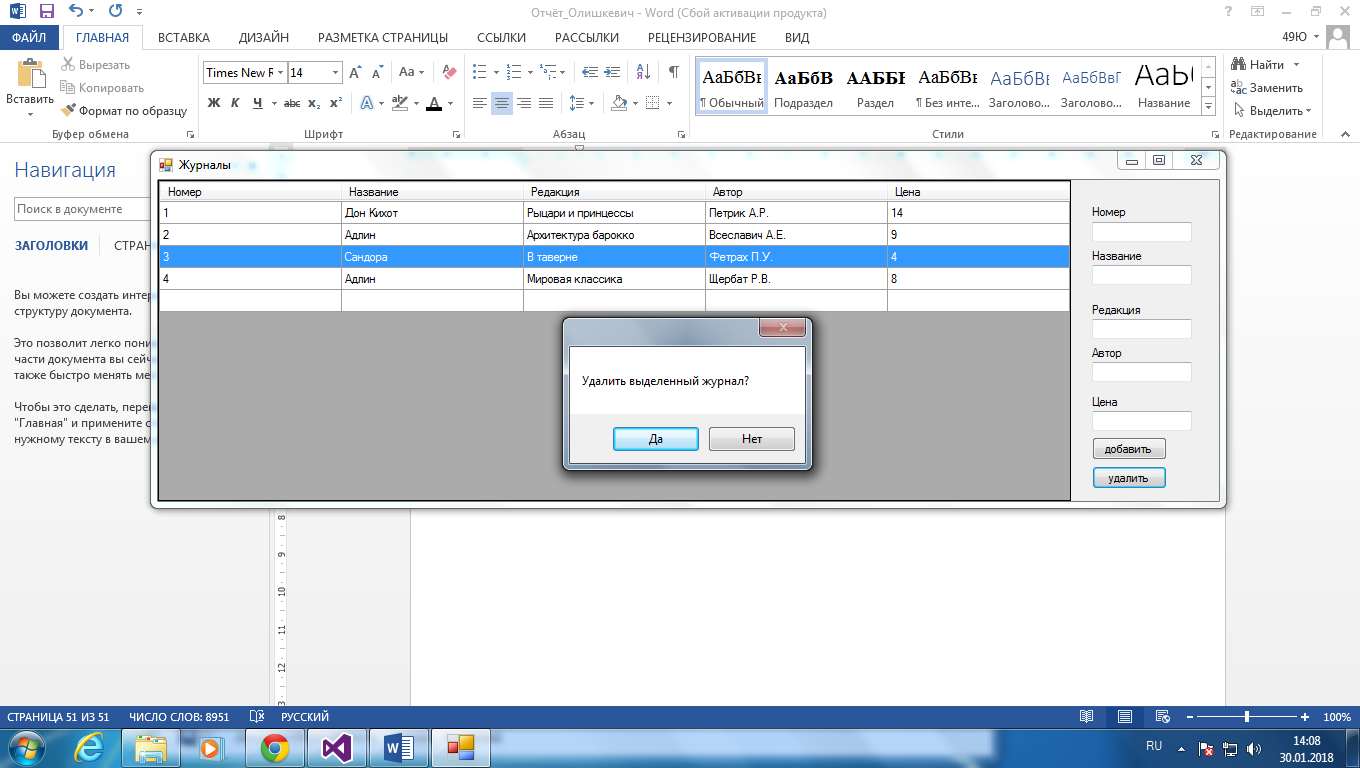


Рисунок 17 – Удаление данных

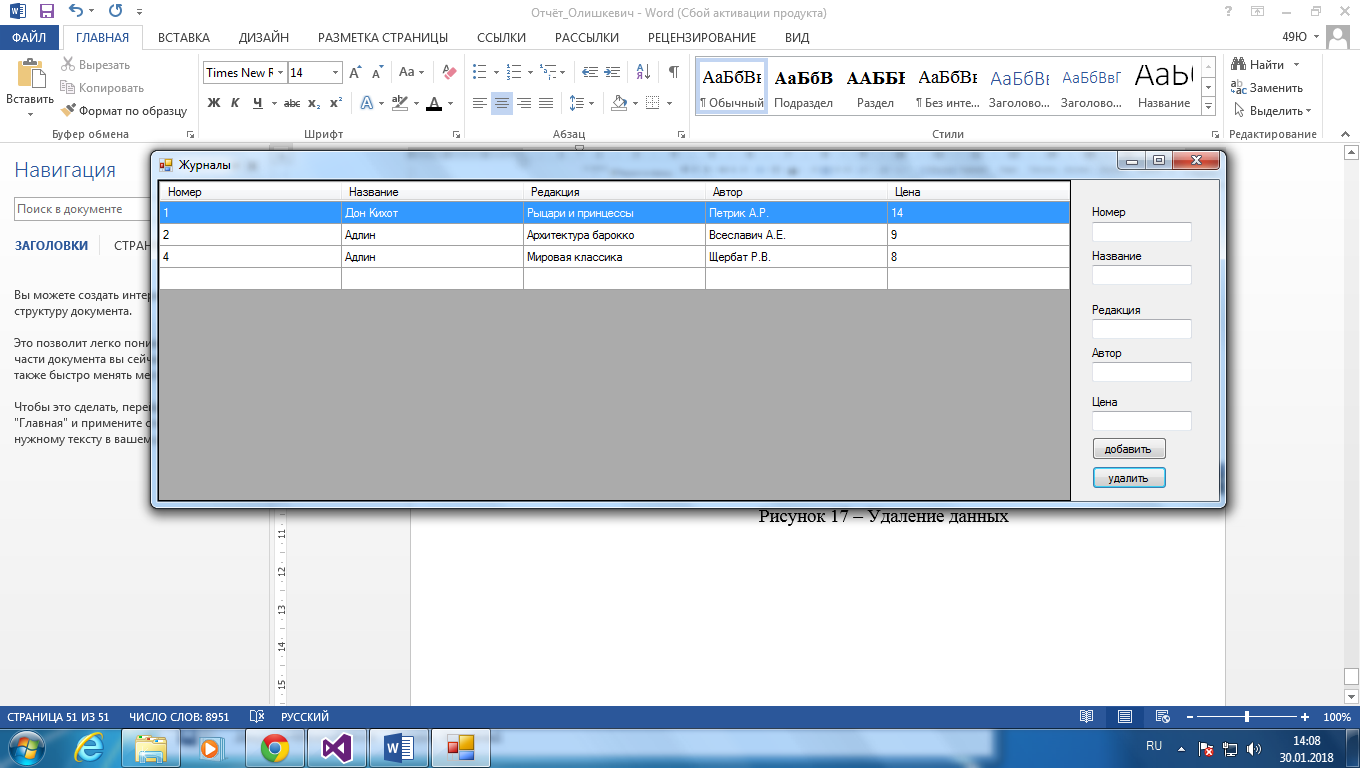


Рисунок 18 – Демонстрация обновленных данных

1.14 Операции с указателями. Регулярные выражения

В .Net приложения можно развертывать простым копированием всех необходимых сборок в одну папку на жестком диске, но на практике обычно выделяются отдельные подкаталоги для связанного содержимого, например, библиотек. В файлах конфигурации .NET можно указать подкаталоги, в которых среда выполнения будет искать частные сборки при запуске приложения.

В C# определен ряд директив препроцессора, оказывающих влияние на интерпретацию исходного кода программы компилятором. Эти директивы определяют порядок интерпретации текста программы перед ее трансляцией в объектный код в том исходном файле, где они появляются. Все директивы препроцессора начинаются со знака #. Кроме того, каждая директива препроцессора должна быть выделена в отдельную строку кода.

Условие задачи: Обязательно должно быть реализовано выполнение следующих функций:

- добавления элемента в конец, начало, середину (до и после введенного номера);

- удаления элемента из начала, середины, конца;

- печати содержимого списка;

- задания конкретного варианта (поиск по заданным полям).

Для решения задач обязательно использовать функции. Все необходимые данные для функций должны передаваться им в качестве параметров. Использование глобальных переменных в функциях не допускается.

Для задания полей даты и времени логических полей, полей ввода целых, вещевственных или логических данных использовать регулярные выражения.

Структура записи о сельскохозяйственном продукте содержит поля: наименование района (где выращивают), наименование продукта, площадь (га), урожайность (кг/га), цена за 1 кг, потери при транспортировке (%), стоимость продукта, предполагаемая дата сбора. Поиск - по наименованию района, урожайности, предполагаемой дате сбора.

Алгоритм решения: Была открыта консоль, пользователь ввел число, соответствующее необходимому номеру меню. Введены данные и помещены в класс, указатель выставлен на начальный. Осуществлен просмотр данных по указателю, перемещающемуся от начального элемента к последнему с выводом на экран. Для поиска пользователь ввел фамилию клиента и перебором каждого элемента при совпадении выводятся все данные. Добавление элемента осуществляется справа и слева от элемента по введенному ключу, далее данные заполнены данными и указатели переопределяются с добавлением нового. Удаление элемента произошло после нахождения элемента по его ключу, после чего указатели были переопределены.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

namespace hyin9

{

class SportsComplex

{

public string LastNameOfTheClient;

public int SportsActivityCode;

public string KindOfSportsActivity;

public string LastNameOfTrainer;

public DateTime DateOfTraining;

public string StartTimeOfTraining;

public int NumberOfMinutes;

public double Rate;

public int Key;

public SportsComplex Next;

public SportsComplex()

{

this.Next = null;

}

public SportsComplex(string LastNameOfTheClient, int SportsActivityCode, string KindOfSportsActivity,

string LastNameOfTrainer, DateTime DateOfTraining, string StartTimeOfTraining, int NumberOfMinutes,

double Rate, int Key)

{

this.LastNameOfTheClient = LastNameOfTheClient;

this.SportsActivityCode = SportsActivityCode;

this.KindOfSportsActivity = KindOfSportsActivity;

this.LastNameOfTrainer = LastNameOfTrainer;

this.DateOfTraining = DateOfTraining;

this.StartTimeOfTraining = StartTimeOfTraining;

this.NumberOfMinutes = NumberOfMinutes;

this.Rate = Rate;

this.Key = Key;

this.Next = null;

}

}

class LineList

{

public string LastNameOfTheClient;

public int SportsActivityCode;

public string KindOfSportsActivity;

public string LastNameOfTrainer;

public DateTime DateOfTraining;

public string StartTimeOfTraining;

public int NumberOfMinutes;

public double Rate;

public int Key;

public SportsComplex head = null;

public SportsComplex current = null;

public LineList() { }

public void Menu()

{

Console.Clear();

Console.WriteLine("Главное меню:\n");

Console.WriteLine(" 1. Заполнить список.");

Console.WriteLine(" 2. Просмотреть список.");

Console.WriteLine(" 3. Добавить новый элемент справа.");

Console.WriteLine(" 4. Добавить новый элемент слева.");

Console.WriteLine(" 5. Поиск элемента.");

Console.WriteLine(" 6. Удалить элемент.");

Console.WriteLine(" 7. Очистить список.");

Console.WriteLine(" 8. Выход.\n");

Console.WriteLine("\nВЫБЕРИТЕ НОМЕР НУЖНОГО ЗАДАНИЯ!\n");

}

public void FirstElem()

{

Console.Clear();

Console.Write("Введите количество спорткомплексов: ");

int n = int.Parse(Console.ReadLine());

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

{

Console.WriteLine("Заполнение {0} спорткомплекса: \n");

Console.Write("Введите ключ: ");

Key = int.Parse(Console.ReadLine());

Console.Write("Введите фамилию клиента: ");

LastNameOfTheClient = Console.ReadLine();

Console.Write("Введите код спортивного занятия: ");

SportsActivityCode = int.Parse(Console.ReadLine());

Console.Write("Введите вид спортивного занятия: ");

KindOfSportsActivity = Console.ReadLine();

Console.Write("Введите фамилию тренера: ");

LastNameOfTrainer = Console.ReadLine();

Console.Write("Введите дату тренировки: ");

DateOfTraining = DateTime.Parse(Console.ReadLine());

Console.Write("Введите время начала тренировки: ");

StartTimeOfTraining = Console.ReadLine();

Console.Write("Введите количество минут: ");

NumberOfMinutes = int.Parse(Console.ReadLine());

Console.Write("Введите тариф: ");

Rate = double.Parse(Console.ReadLine());

string text = Convert.ToString(Key) + " " + LastNameOfTheClient + " " +

Convert.ToString(SportsActivityCode) + " " + Convert.ToString(KindOfSportsActivity) +

" " + LastNameOfTrainer + " " + Convert.ToString(DateOfTraining) + " " +

StartTimeOfTraining + " " + Convert.ToString(NumberOfMinutes) + " " + Convert.ToString(Rate);

Regex theReg = new Regex(@"(?<key>\d\b\s)" +

@"(?<lnots>\^[А-ЯЁ][а-яё]+ [А-ЯЁ][а-яё]+$)" +

@"(?<sac>\^[А-ЯЁ][а-яё]+ [А-ЯЁ][а-яё]+$)" +

@"(?<kosa>\^[А-ЯЁ][а-яё]+[А-ЯЁ][а-яё]+$)" +

@"(?<lnot>\^[А-ЯЁ][а-яё]+ [А-ЯЁ][а-яё]+$)" +

@"(?<dot>\d([0-9]{1,4}).\d([0-9]{1,2}).\d([0-9]{1,2}))\b\s" +

@"(?<stot>\d([0-9]{1,2}):\d([0-9]{1,2}):\d([0-9]{1,2}))\b\s" +

@"(?<nom>\d\b\s)" +

@"(?<rate>\d+\b)");

MatchCollection thematches1 = theReg.Matches(text);

foreach (Match sss in thematches1)

{

if (!theReg.IsMatch(Convert.ToString(sss))) { Console.WriteLine("ошибка"); };

Console.WriteLine("\ntheMatch: {0}", sss.ToString());

Console.WriteLine("key: {0}", sss.Groups["key"]);

Console.WriteLine("lnots: {0}", sss.Groups["lnots"]);

Console.WriteLine("sac: {0}", sss.Groups["sac"]);

Console.WriteLine("kosa: {0}", sss.Groups["kosa"]);

Console.WriteLine("lnot: {0}", sss.Groups["lnot"]);

Console.WriteLine("dot: {0}", sss.Groups["dot"]);

Console.WriteLine("stot: {0}", sss.Groups["stot"]);

Console.WriteLine("nom: {0}", sss.Groups["nom"]);

Console.WriteLine("rate: {0}", sss.Groups["rate"]);

}

SportsComplex temp = new SportsComplex(LastNameOfTheClient, SportsActivityCode,

KindOfSportsActivity, LastNameOfTrainer, DateOfTraining, StartTimeOfTraining,

NumberOfMinutes, Rate, Key);

if (head == null)

head = temp;

else

{

temp.Next = head;

head = temp;

}

}

}

public void Show()

{

Console.Clear();

Console.WriteLine("Однонаправленный список: ");

if (head == null)

Console.WriteLine("список пуст!");

else

{

SportsComplex temp = head;

while (temp != null)

{

Console.Write("{0}) фамилия клиента - {1}, код спортивного занятия - {2}, вид спортивного занятия - {3}, фамилия тренера - {4}, дата тренировки - {5}, время начала тренировки - {6}, количество минут - {7}, тариф - {8}\n",

temp.Key, temp.LastNameOfTheClient, temp.SportsActivityCode, temp.KindOfSportsActivity,

temp.LastNameOfTrainer, temp.DateOfTraining, temp.StartTimeOfTraining,

temp.NumberOfMinutes, temp.Rate);

temp = temp.Next;

}

Console.WriteLine("");

}

}

public void FindElem()

{

Console.WriteLine("Введите наименование района для поиска: ");

LastNameOfTheClient = Console.ReadLine();

SportsComplex temp = head;

Console.WriteLine("Результаты поиска: ");

while (temp != null)

{

if (temp.LastNameOfTheClient == LastNameOfTheClient)

Console.Write("Ключ - {0}, фамилия клиента - {1}, код спортивного занятия - {2}, вид спортивного занятия - {3}, фамилия тренера - {4}, дата тренировки - {5}, время начала тренировки - {6}, количество минут - {7}, тариф - {8}\n",

temp.Key, temp.LastNameOfTheClient, temp.SportsActivityCode, temp.KindOfSportsActivity,

temp.LastNameOfTrainer, temp.DateOfTraining,

temp.StartTimeOfTraining,

temp.NumberOfMinutes, temp.Rate);

temp = temp.Next;

}

}

public void Right\_Element()

{

Console.Clear();

if (head == null)

{

Console.WriteLine("Заполните список!");

} else {

Console.WriteLine("После какого элемента вставить новый? ");

int m = int.Parse(Console.ReadLine());

current = head;

for (int i = 1; i < m; i++)

current = current.Next;

Console.WriteLine("Заполнение {0} спорткомплекса: \n");

Console.Write("Введите ключ: ");

Key = int.Parse(Console.ReadLine());

Console.Write("Введите фамилию клиента: ");

LastNameOfTheClient = Console.ReadLine();

Console.Write("Введите код спортивного занятия: ");

SportsActivityCode = int.Parse(Console.ReadLine());

Console.Write("Введите вид спортивного занятия: ");

KindOfSportsActivity = Console.ReadLine();

Console.Write("Введите фамилию тренера: ");

LastNameOfTrainer = Console.ReadLine();

Console.Write("Введите дату тренировки: ");

DateOfTraining = DateTime.Parse(Console.ReadLine());

Console.Write("Введите время начала тренировки: ");

StartTimeOfTraining = Console.ReadLine();

Console.Write("Введите количество минут: ");

NumberOfMinutes = int.Parse(Console.ReadLine());

Console.Write("Введите тариф: ");

Rate = double.Parse(Console.ReadLine());

SportsComplex temp = new SportsComplex(LastNameOfTheClient, SportsActivityCode,

KindOfSportsActivity, LastNameOfTrainer, DateOfTraining, StartTimeOfTraining,

NumberOfMinutes, Rate, Key);

temp.Next = current.Next;

current.Next = temp;

current = temp;

} }

public void Left\_Element()

{

Console.Clear();

if (head == null)

{

Console.WriteLine("Заполните список!");

}

else

{

Console.WriteLine("Перед каким элементом вставить новый? ");

int m = int.Parse(Console.ReadLine());

current = head;

for (int i = 1; i < m - 1; i++)

current = current.Next;

Console.WriteLine("Заполнение {0} спорткомплекса: \n");

Console.Write("Введите ключ: ");

Key = int.Parse(Console.ReadLine());

Console.Write("Введите фамилию клиента: ");

LastNameOfTheClient = Console.ReadLine();

Console.Write("Введите код спортивного занятия: ");

SportsActivityCode = int.Parse(Console.ReadLine());

Console.Write("Введите вид спортивного занятия: ");

KindOfSportsActivity = Console.ReadLine();

Console.Write("Введите фамилию тренера: ");

LastNameOfTrainer = Console.ReadLine();

Console.Write("Введите дату тренировки: ");

DateOfTraining = DateTime.Parse(Console.ReadLine());

Console.Write("Введите время начала тренировки: ");

StartTimeOfTraining = Console.ReadLine();

Console.Write("Введите количество минут: ");

NumberOfMinutes = int.Parse(Console.ReadLine());

Console.Write("Введите тариф: ");

Rate = double.Parse(Console.ReadLine());

SportsComplex temp = new SportsComplex(LastNameOfTheClient, SportsActivityCode,

KindOfSportsActivity, LastNameOfTrainer, DateOfTraining, StartTimeOfTraining,

NumberOfMinutes, Rate, Key);

temp.Next = current.Next;

current.Next = temp;

current = temp;

}

}

public void Delete\_Elements()

{

Console.Clear();

if (head != null)

{

Show();

SportsComplex temp = head;

SportsComplex prev = head;

int i = 1;

Console.WriteLine("Введите позицию в списке (число): ");

int pos = int.Parse(Console.ReadLine());

while (pos != i)

{ prev = temp;

temp = temp.Next;

i++; }

if (head == temp)

head = head.Next;

else

prev.Next = temp.Next; }

else

Console.WriteLine("Список и так пустой! Удалять нечего."); }

public void Clear\_list()

{

SportsComplex temp;

while (head != null)

{

temp = head.Next;

head = temp;

} } }

class Program

{

static void Main(string[] args)

{

LineList List = new LineList();

List.Menu();

int c = int.Parse(Console.ReadLine());

while (c != 8)

{

switch (c)

{

case 1: List.FirstElem(); break;

case 2: List.Show(); break;

case 3: List.Right\_Element(); break;

case 4: List.Left\_Element(); break;

case 5: List.FindElem(); break;

case 6: List.Delete\_Elements(); break;

case 7: List.Clear\_list(); break;

case 8: break;

default: Console.Write("Команды с таким номером нет!"); break;

}

do Console.Write("\nДля продолжения нажмите клавишу Enter....");

while (Console.ReadKey().Key !=

ConsoleKey.Enter);

List.Menu();

c = int.Parse(Console.ReadLine());

} } }}

Тестирование приложения:

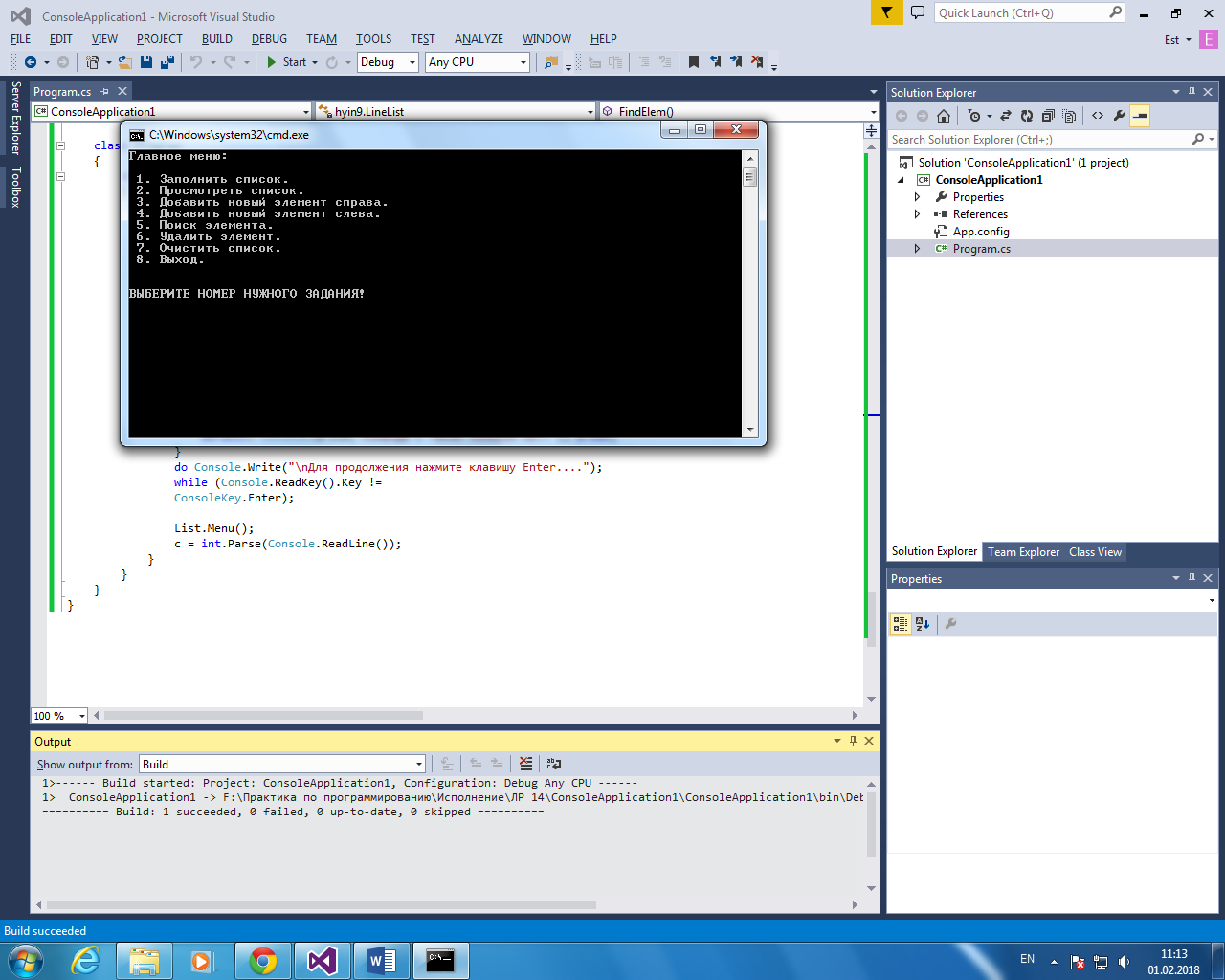


Рисунок 19 – Демонстрация главного меню

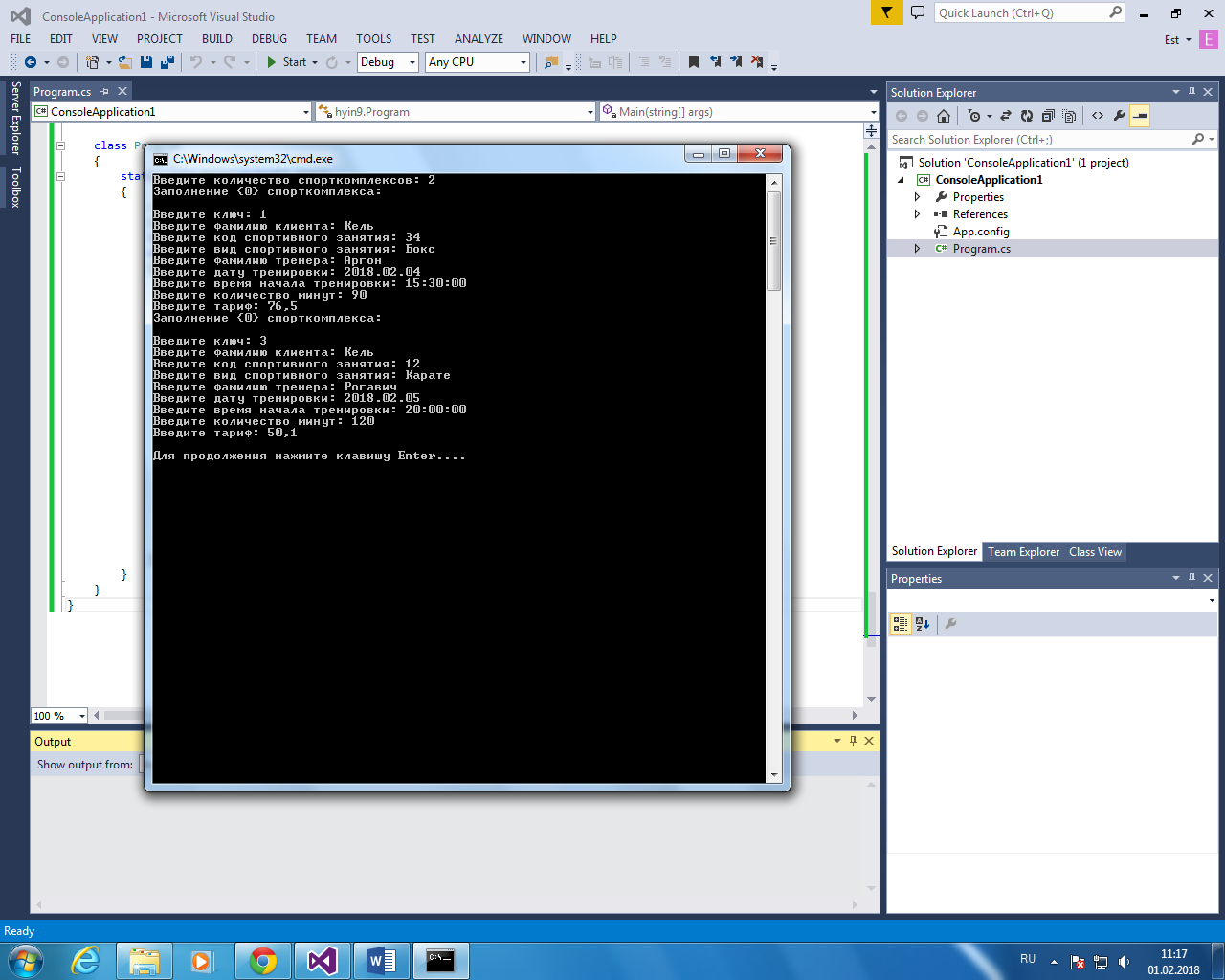


Рисунок 20 – Демонстрация заполнения списка

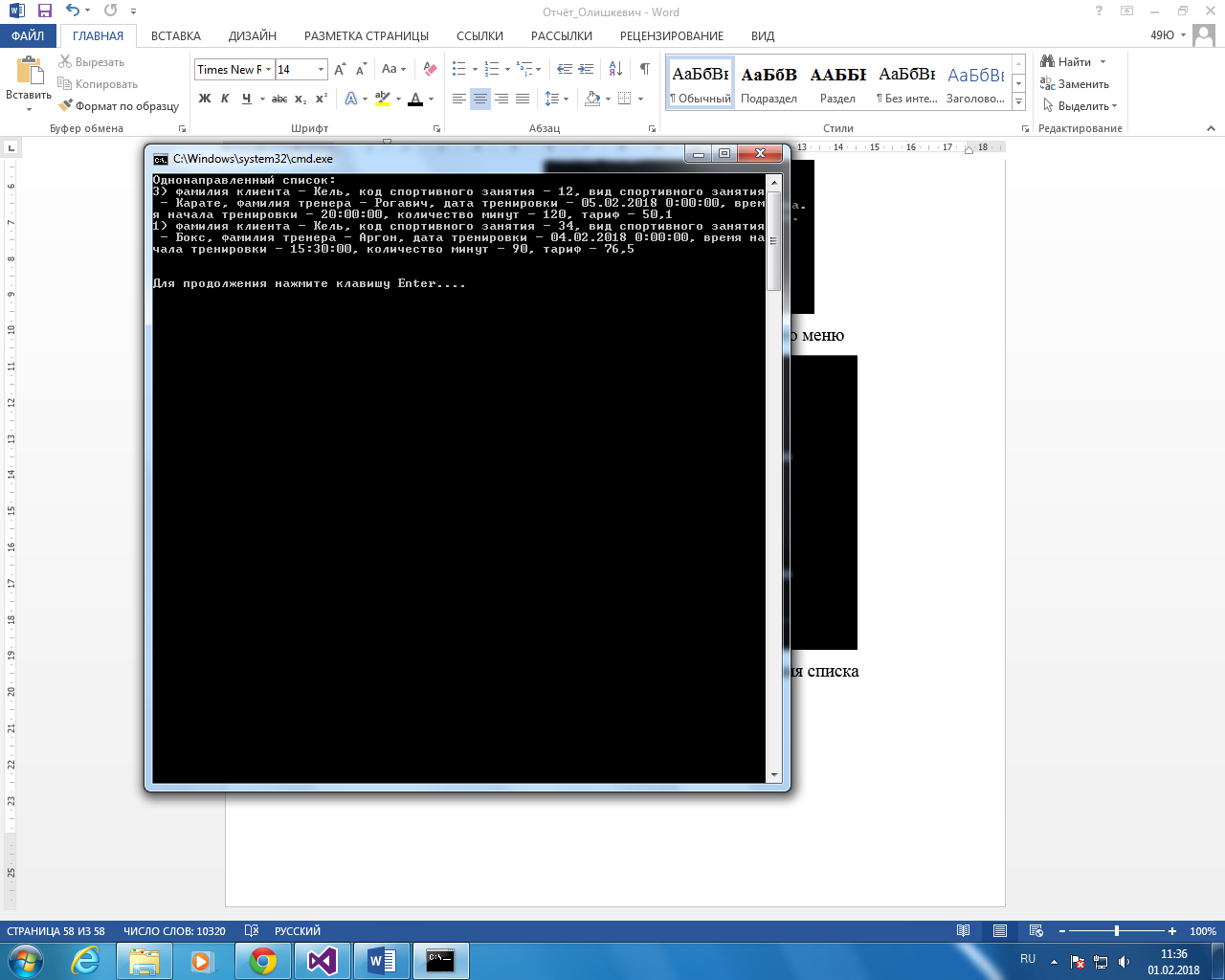


Рисунок 21 – Демонстрация списка

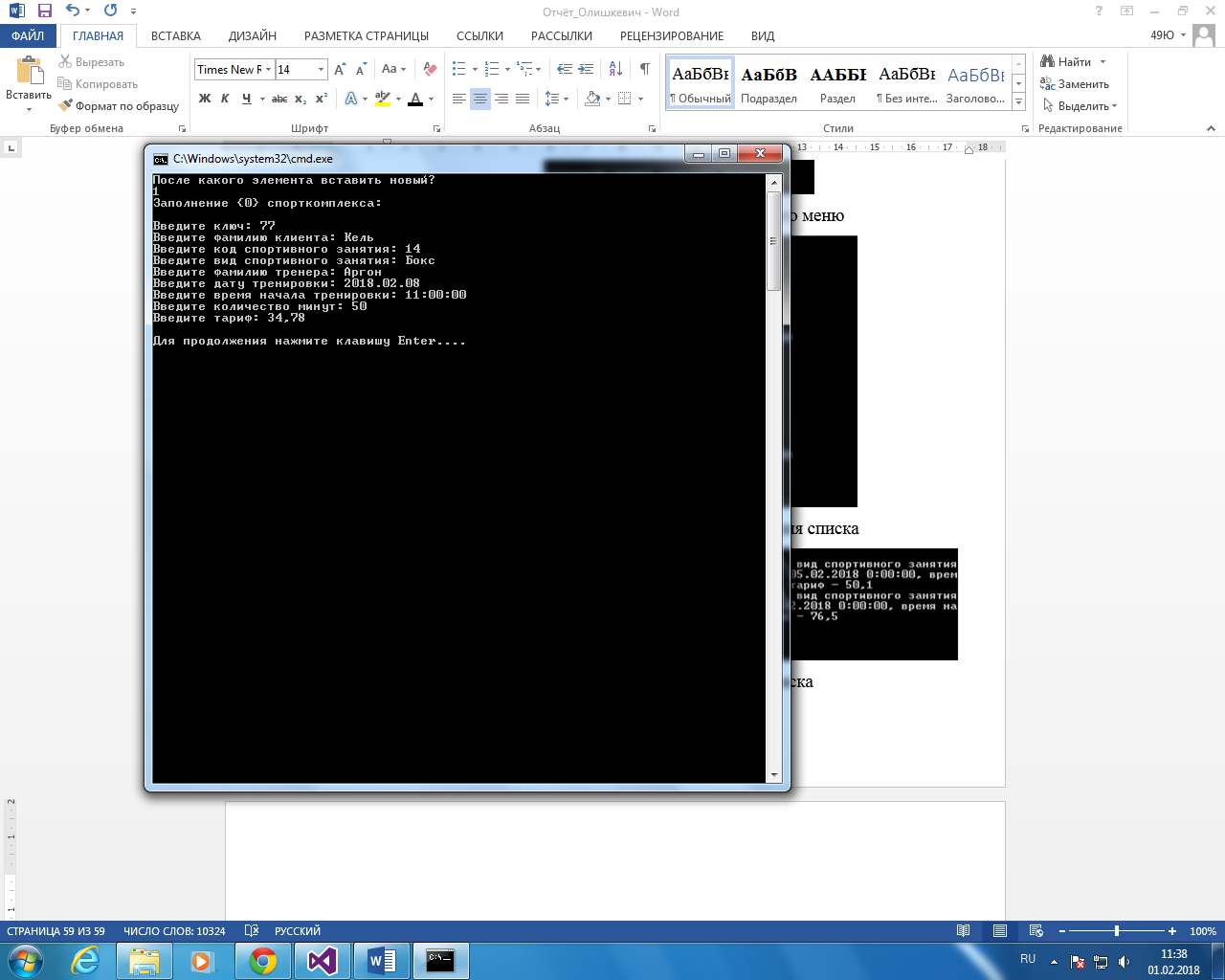


Рисунок 21 – Демонстрация вставки элемента после заданного

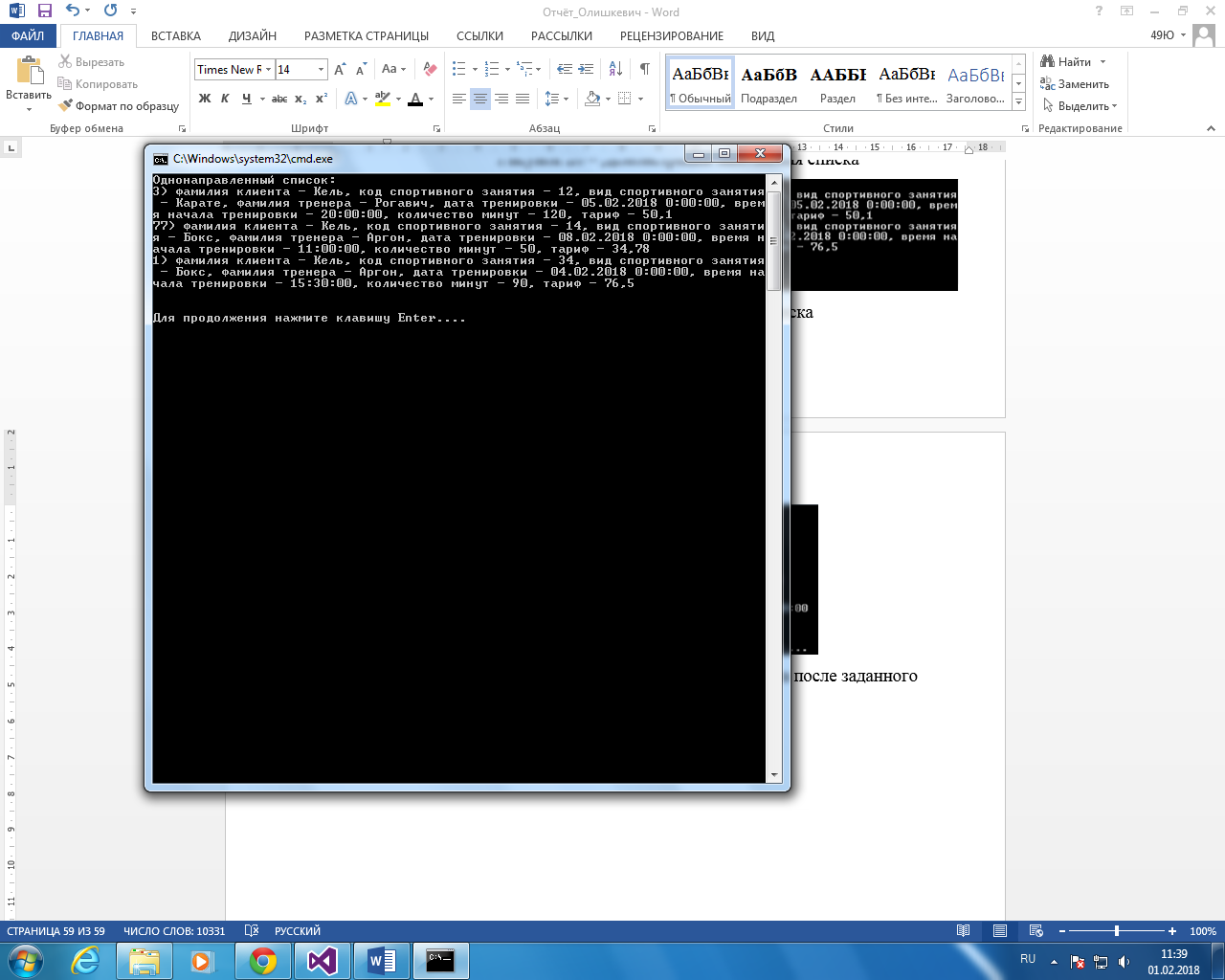


Рисунок 22 – Демонстрация вставленного элемента после заданного

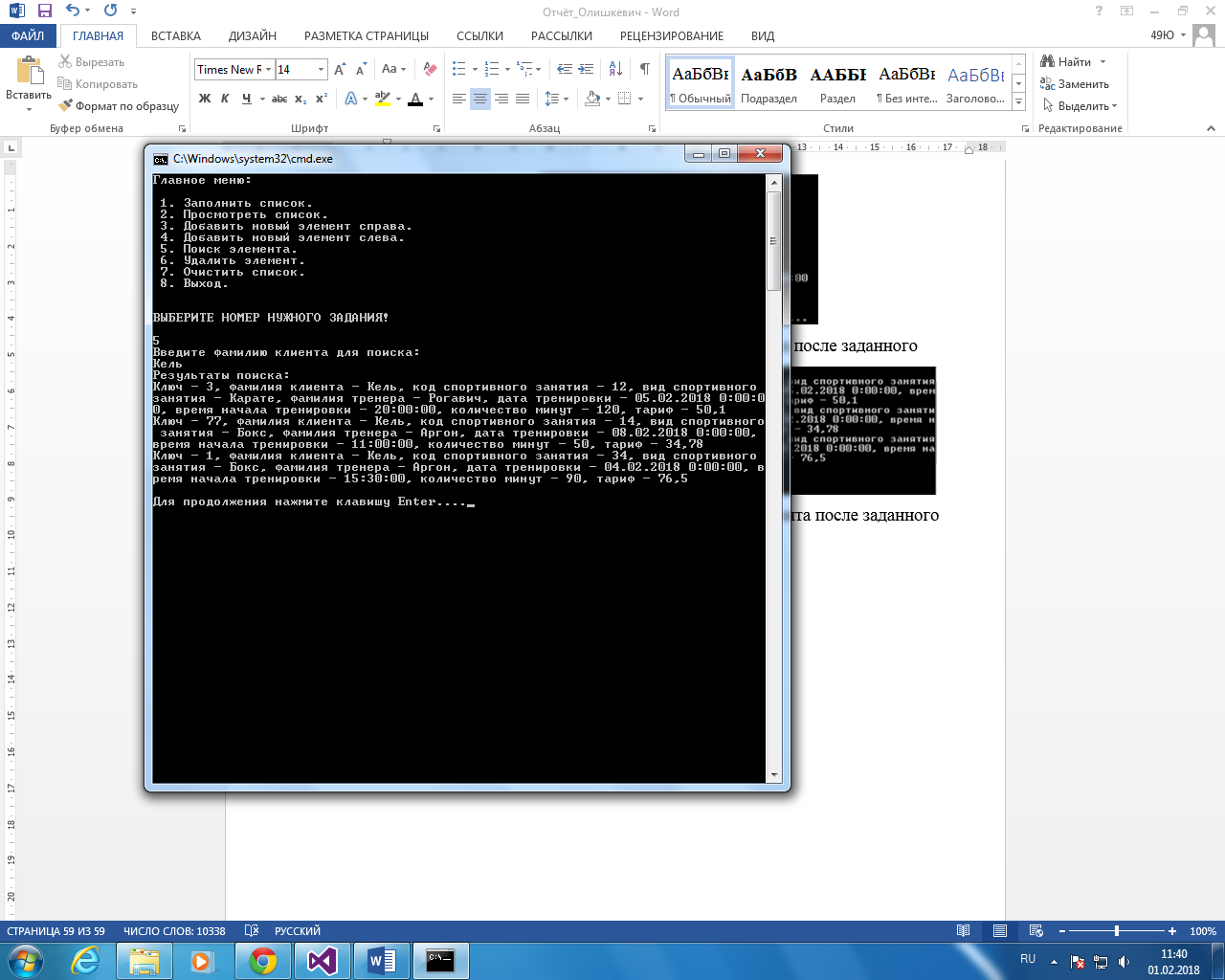


Рисунок 23 – Демонстрация поиска элемента

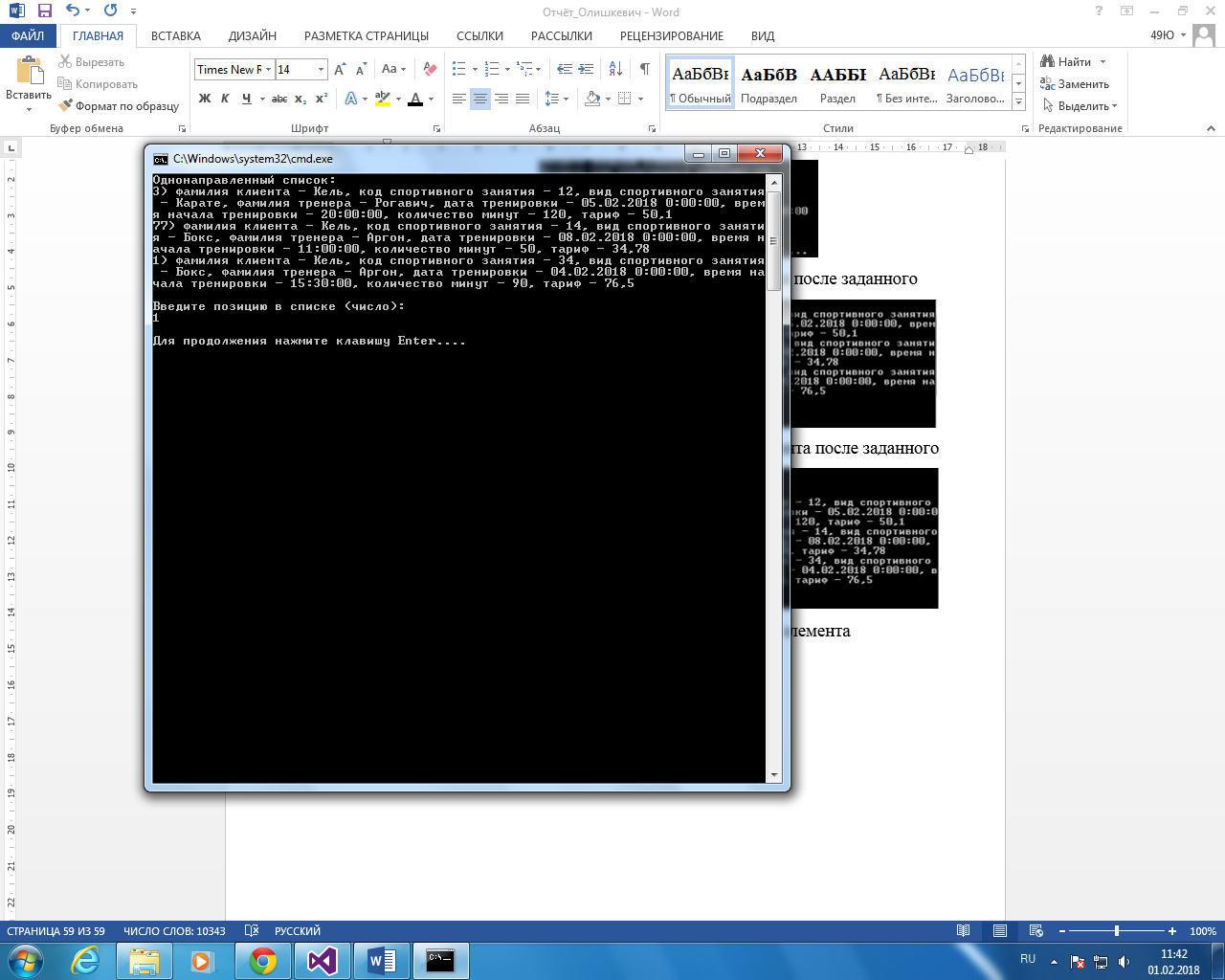


Рисунок 24 – Демонстрация удаления элемента

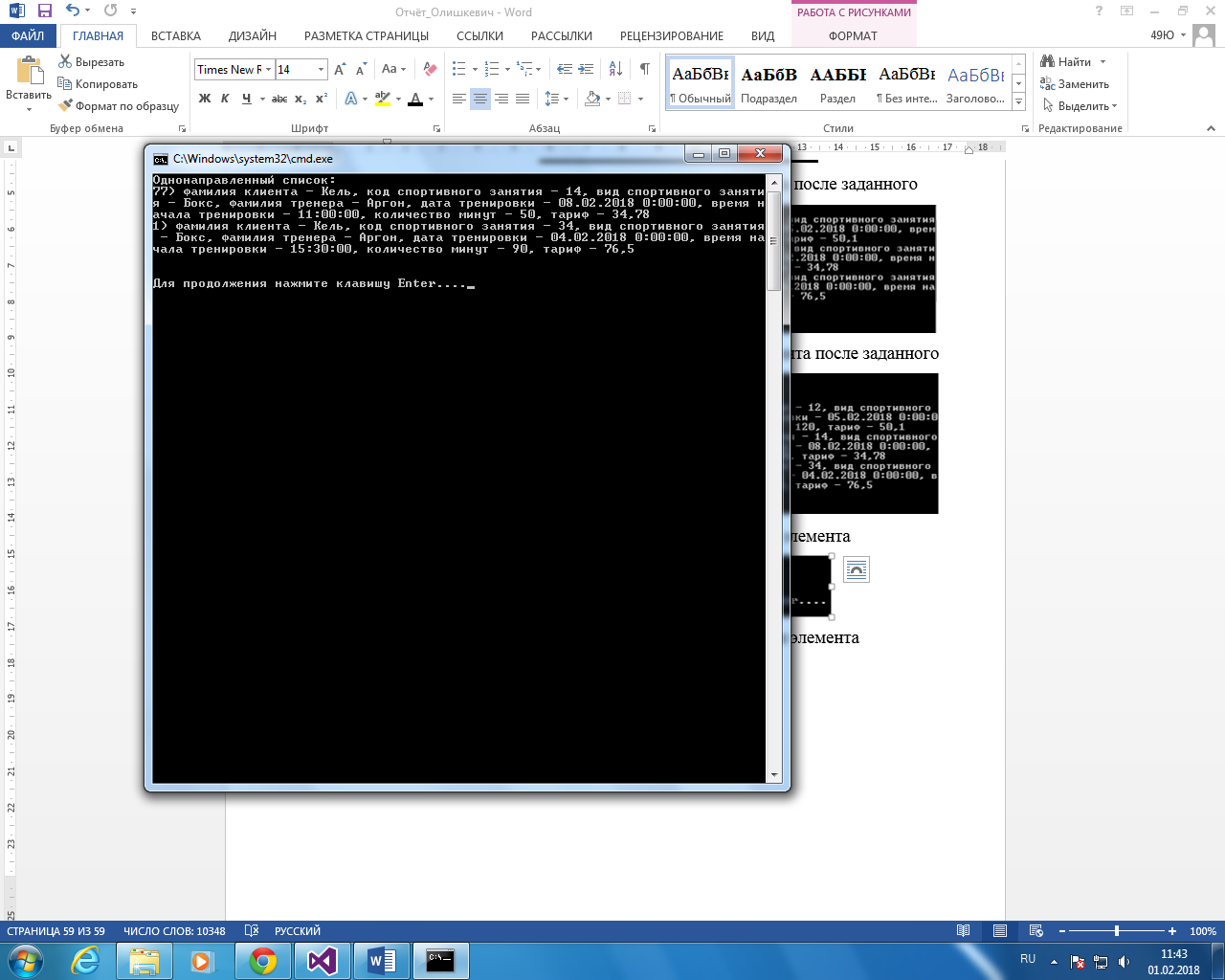


Рисунок 25 – Демонстрация удаленного элемента

2 Разработка Windows-приложений средствами библиотеки классов Microsoft .NET Framework

Оконные приложения строятся по принципам событийно-управляемого программирования (event-driven programming) - стиля программирования, при котором поведение компонента системы определяется набором возможных внешних событий и ответных реакций компонента на них. Такими компонентами в Windows являются окна.

С каждым окном в Windows связана определенная функция обработки событий – оконная функция. События для окон называются сообщениями. Сообщение относится к тому или иному типу, идентифицируемому определенным кодом (32-битным целым числом), и сопровождается парой 32-битных параметров (WPARAM и LPARAM), интерпретация которых зависит от типа сообщения.

Задача любого оконного приложения - создать главное окно и сообщить Windows функцию обработки событий для этого окна. Все самое интересное для приложения будет происходить именно в функции обработки событий главного окна.

В Windows программа пассивна. После запуска она ждет, когда ей уделит внимание операционная система. Операционная система делает это посылкой сообщений. Сообщения могут быть разного типа, они функционируют в системе достаточно хаотично, и приложение не знает, какого типа сообщение придет следующим. Логика построения Windows-приложения должна обеспечивать корректную и предсказуемую работу при поступлении сообщений любого типа.

Условие задачи 1: Создать оконное приложение для вычисления значений функций:

|  |  |
| --- | --- |
|  |  |

Алгоритм решения: Кнопка 1 преобразовывает текстовую информацию, введенную пользователем в вещественные значения. Исходя из условия вычисляется соответствующее значение. Предусмотрены исключительные ситуации. Найденные значения записываются в таблицу и текстовые поля. Кнопка 2 позволяет закрыть программу, кнопка 3 – переход к следующему заданию. Компонент PictureBox отображает задание.

Открылась форма, пользователь заполнил данные и нажал на «Выполнить задание». На экране отобразился результат.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Листинг формы:

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

double xn, xk, xh, x, y, a, ymax, ymin, yt;

int n, i;

if ((textBox3.Text != "") && (textBox4.Text != "") && (textBox5.Text != "") && (textBox6.Text != ""))

{

xn = Convert.ToDouble(textBox3.Text);

xk = Convert.ToDouble(textBox4.Text);

xh = Convert.ToDouble(textBox5.Text);

a = Convert.ToDouble(textBox6.Text);

dataGridView1.Columns.Clear();

dataGridView1.ColumnCount = 2;

dataGridView1.Rows.Add(Math.Ceiling((xk - xn) / xh) + 1);

dataGridView1.Columns[0].Name = " X";

dataGridView1.Columns[1].Name = " Y";

i = 0;

x = xn;

ymax = -1.8e307;

ymin = 1.8e307;

while (x <= xk)

{

if (x <= 0)

{

y = 0;

}

else if (x <= a)

{

try

{

if (x < 0 || Math.Log10(x) < 0)

{

throw new Exception();

}

y = Math.Log(Math.Log10(x));

}

catch (Exception)

{

dataGridView1.Rows[i].Cells[0].Value = Convert.ToString(x);

dataGridView1.Rows[i].Cells[1].Value = "Область недопустимых значений";

x += xh;

i++;

dataGridView1.Rows.Add(1);

continue;

}

}

else

{

y = Math.Pow(Math.Sin((Math.Pow(a, 2)\* x)), 0.33333333333333);

}

dataGridView1.Rows[i].Cells[0].Value = Convert.ToString(x);

yt = Math.Ceiling(y \* 100) / 100;

dataGridView1.Rows[i].Cells[1].Value = Convert.ToString(yt);

string str = Convert.ToString(dataGridView1.Rows[i].Cells[1].Value);

if (str == "NaN")

{

dataGridView1.Rows[i].Cells[1].Value = "Требуются доп. исследования";

}

if (y > ymax)

ymax = Math.Ceiling(y \* 100) / 100;

if (y < ymin)

ymin = Math.Ceiling(y \* 100) / 100;

x += xh;

i++;

dataGridView1.Rows.Add(1);

}

textBox1.Text = Convert.ToString(ymax);

textBox2.Text = Convert.ToString(ymin);

dataGridView1.Rows.RemoveAt(i);

}

else

{

MessageBox.Show("Заполните, пожалуйста, данные", "Ошибка ввода данных",

MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button3\_Click(object sender, EventArgs e)

{

Form2 f2 = new Form2();

f2.Show();

}

}

}

Тестирование приложения:

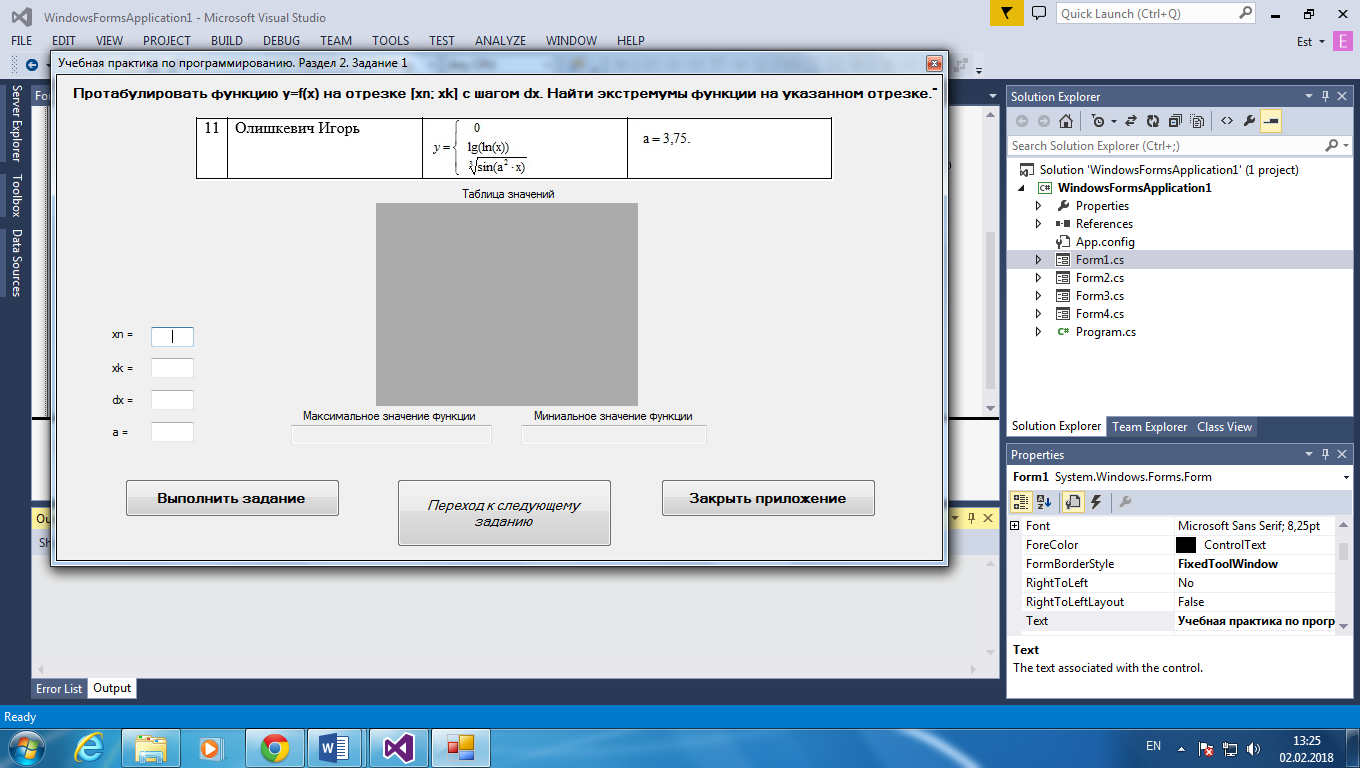


Рисунок 26 – Демонстрация рабочего окна

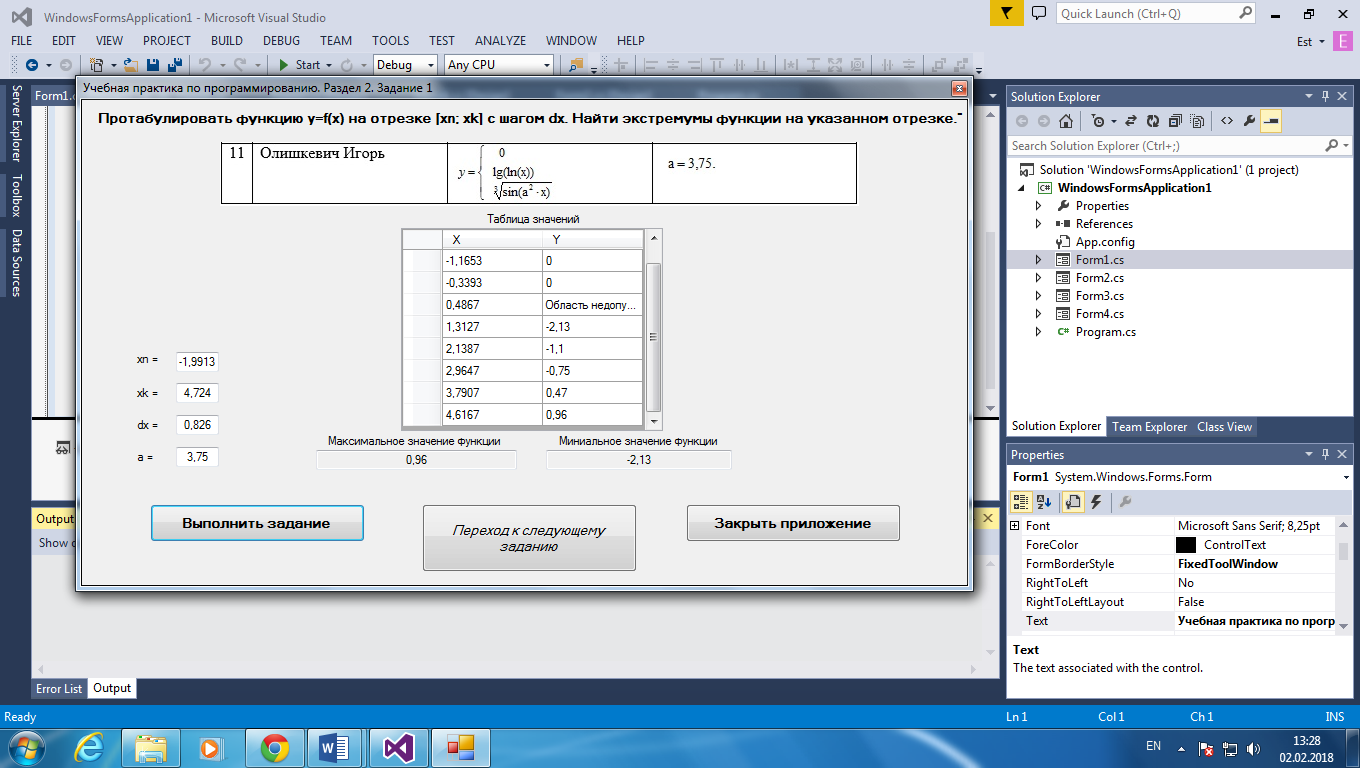


Рисунок 27 – Демонстрация полученных значений

Условие задачи 2: Создать оконное приложение для вычисления значений функций:

1. Вычислить среднеарифметическое значение положительных элементов всего массива.

2. Определить наибольший элемент среди каждых четвертых элементов второй строки массива.

Алгоритм решения: Кнопка 1 преобразовывает текстовую информацию, введенную пользователем в целые значения, затем создает таблицу соответствующего размера. Кнопка 3 считывает значения из таблицы, затем производит вычисления. Предусмотрены исключительные ситуации. Найденные значения записываются в таблицу и текстовые поля. Кнопка 2 позволяет закрыть программу, кнопка 4 – переход к форме «Справка». Компонент TabControl отображает задание на первой вкладке, на второй позволяет выполнить задание. Компонент MainMenu по клику на вкладку «О раработчике» открывает форму с контактной информацией исполнителя.

Открылась форма, пользователь открыл таблицу, заполнил данные и нажал на кнопку выполнения. На экране отобразился результат.

Листинг основной формы:

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;

namespace WindowsFormsApplication1

{

public partial class Form2 : Form

{

public int n, m;

public Form2()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

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

{

if ((textBox1.Text != "") && (textBox2.Text != ""))

{

m = Convert.ToInt32(textBox1.Text);

n = Convert.ToInt32(textBox2.Text);

dataGridView1.Columns.Clear();

dataGridView1.ColumnCount = n;

dataGridView1.RowCount = m;

}

else

{

MessageBox.Show("Заполните, пожалуйста, данные", "Ошибка ввода данных",

MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void button4\_Click(object sender, EventArgs e)

{

this.Hide();

Form4 f4 = new Form4();

f4.ShowDialog();

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

this.Hide();

Form3 f3 = new Form3();

f3.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

int count = 0, index = 0;

double sum = 0, sr = 0;

int size = Convert.ToInt32(Math.Round(Convert.ToDouble(n) / 4)) + 1;

double[] max = new double[size];

double[,] A = new double[m, n];

for (int i = 1; i <= m; i++)

for (int j = 1; j <= n; j++)

{

A[i - 1, j - 1] = Convert.ToDouble(this.dataGridView1.Rows[i - 1].Cells[j - 1].Value);

if (A[i - 1, j - 1] > 0)

{

sum += A[i - 1, j - 1];

count++;

}

if ((i == 2) && (j % 4 == 0))

{

max[index] = A[i - 1, j - 1];

index++;

}

}

sr = sum / count;

if (checkBox1.Checked == true)

{

this.textBox3.Text = Convert.ToString(sr);

}

else

if (checkBox1.Checked == true)

{

this.textBox3.Text = Convert.ToString("нет элементов");

}

if (checkBox2.Checked == true)

{

this. textBox4.Text = Convert.ToString(max.Max());

}

else

if (checkBox2.Checked == true)

{

this.textBox4.Text = Convert.ToString("нет элементов");

}

return;

}

}

}

Листинг формы контактной информации разработчика:

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;

namespace WindowsFormsApplication1

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

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

{

this.Hide();

Form2 f2 = new Form2();

f2.Show();

}

}

}

Листинг формы справки:

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;

namespace WindowsFormsApplication1

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

}

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

{

this.Hide();

Form2 f2 = new Form2();

f2.Show();

}

}

}

Тестирование приложения:

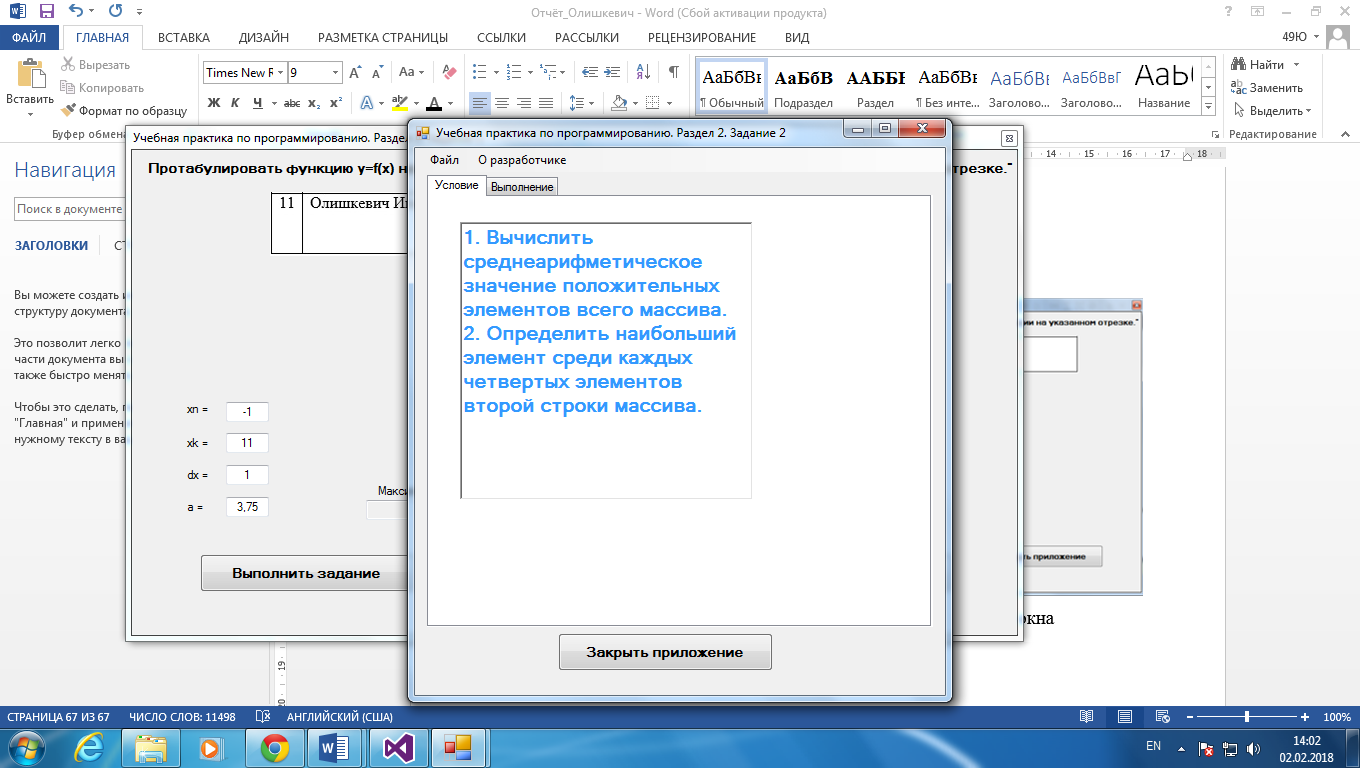


Рисунок 26 – Демонстрация условия

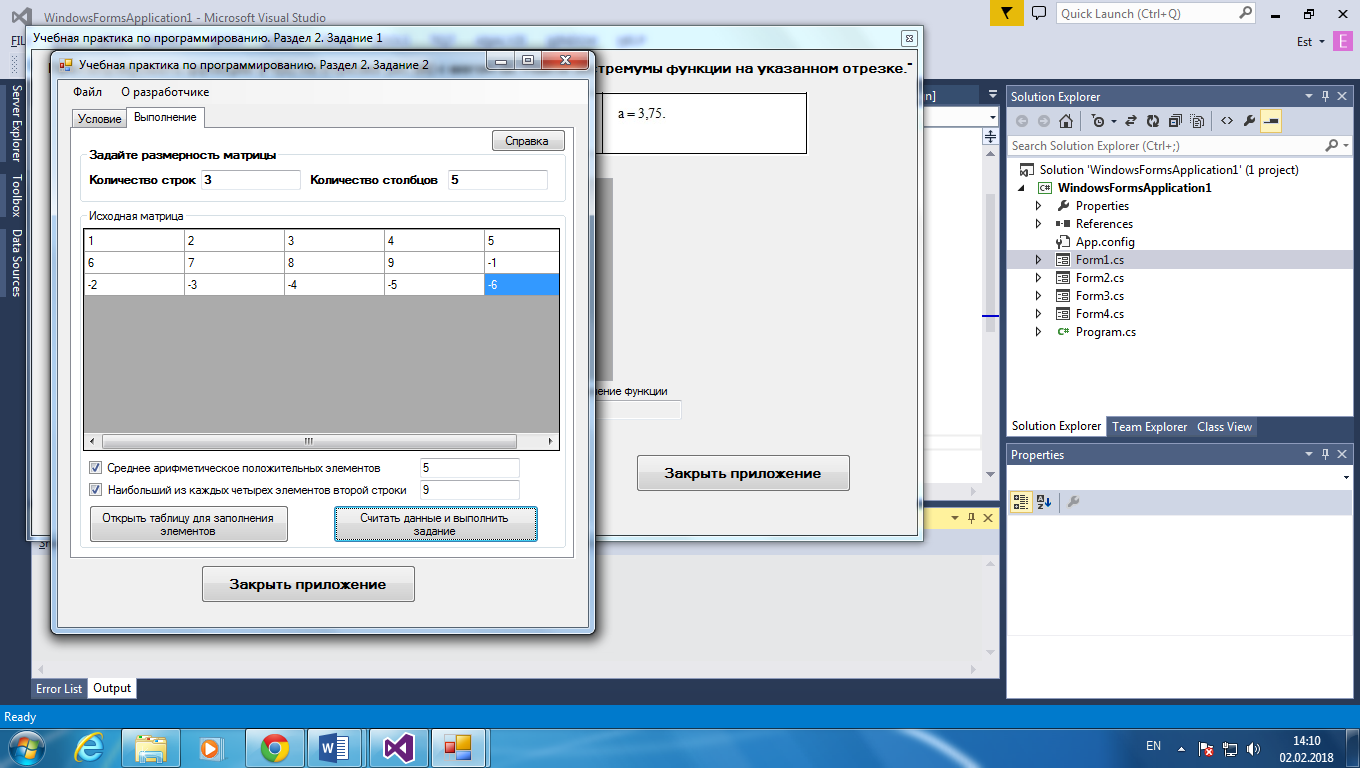


Рисунок 27 – Демонстрация выполнения

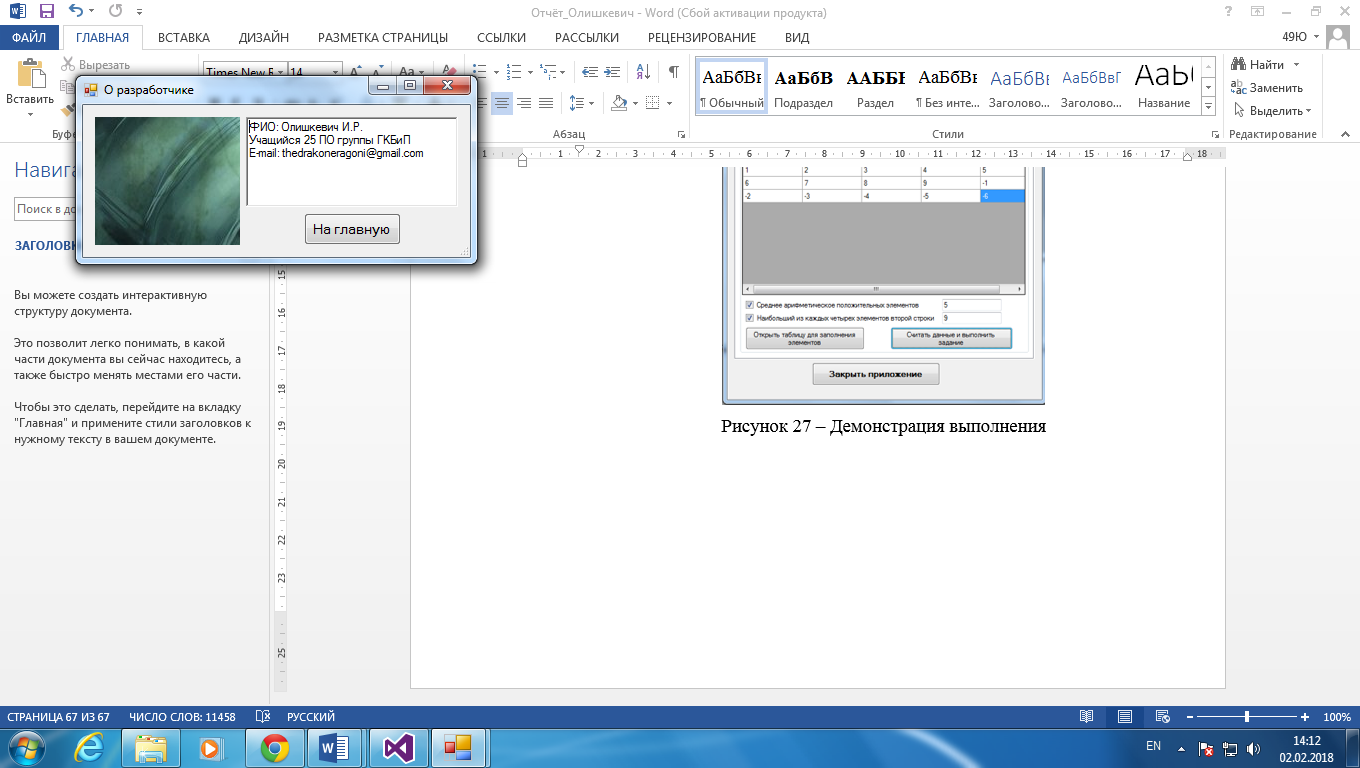


Рисунок 28 – Демонстрация формы «О разработчике»

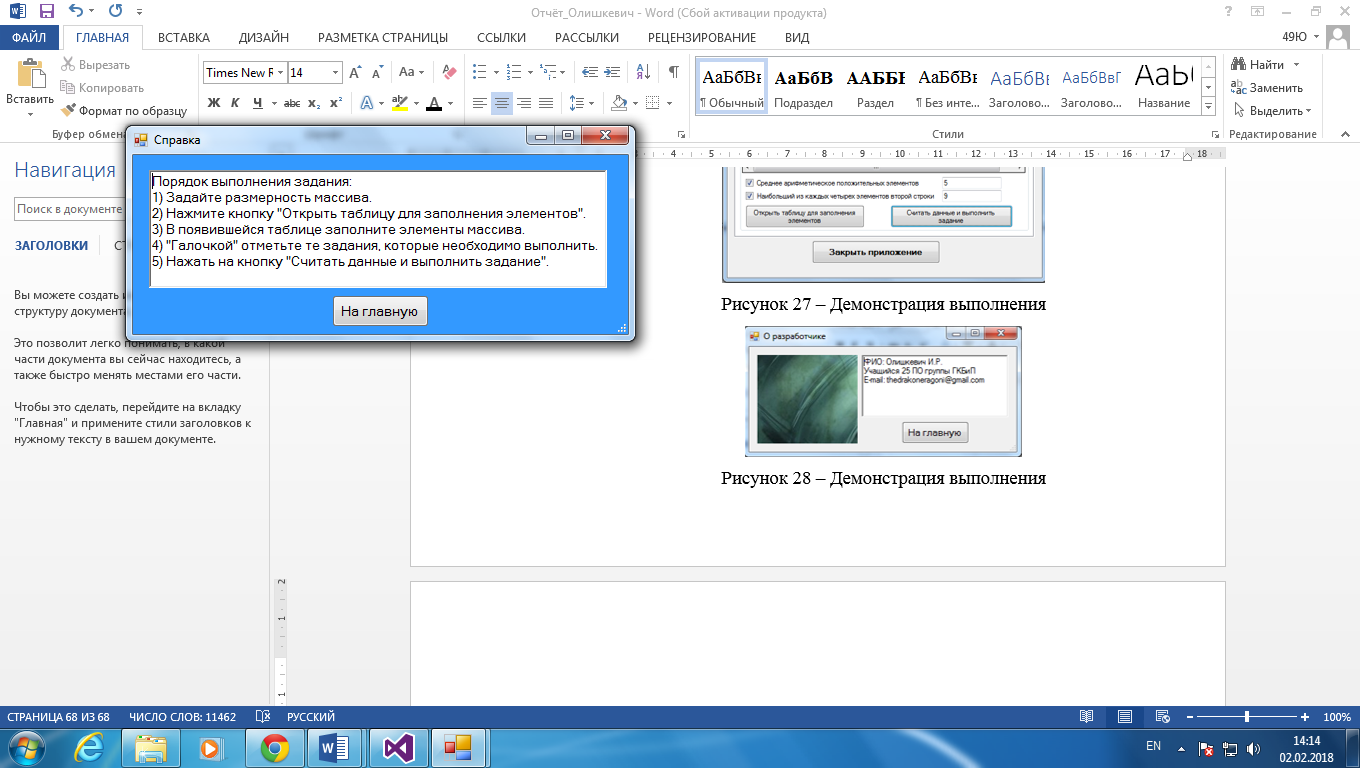


Рисунок 29 – Демонстрация формы «Справка»

Условие задачи 3: Создать собственный компонент: компонент-графический редактор

Алгоритм решения: В библиотеке описали компонент с использованием конструктора и деструктора, использующий счётчик. Задали возврат значения. Описали компонент на основе PictureBox. Добавили описание свойств, обработку событий. Создали новый проект к обозревателю решения, подключили библиотеку компонента. Разместили компонент PictureBox, реализуя компоненту.

Листинг компонента:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

namespace MyControl

{

public partial class MyControl : Component

{

public MyControl()

{

InitializeComponent();

InstanceID = NextInstanceID++;

ClassInstanceCount++;

}

public MyControl(IContainer container)

{

container.Add(this);

InitializeComponent();

}

~MyControl()

{

ClassInstanceCount--;

}

public readonly int InstanceID;

private static int NextInstanceID = 0;

private static long ClassInstanceCount = 0;

public static long InstanceCount

{

get

{

return ClassInstanceCount;

}

}

}

[Description("Модифицированный компонент PictureBox")]

[ToolboxItem(true)]

[ToolboxBitmap(typeof(PictureBoxControl), "icon.bmp")]

[DesignerAttribute("MyControl.PictureBoxControlDesigner")]

[DefaultBindingPropertyAttribute("Image")]

[DockingAttribute(DockingBehavior.Ask)]

public partial class PictureBoxControl : PictureBox

{

private bool changebutton = true;

private Color startcolor = Color.White;

private Color endcolor = Color.Black;

private float size = 8.25F;

private string fontname = "Microsoft Sans Serif";

public PictureBoxControl()

{}

public PictureBoxControl(IContainer container)

{

container.Add(this);

}

[Category("Градиент"), Description("Цвет начала заливки"), DefaultValue(typeof(Color), "Red")]

public Color StartColor

{

get { return startcolor; }

set

{

startcolor = value;

onChange();

}

}

[Category("Градиент"), Description("Цвет завершения заливки"), DefaultValue(typeof(Color), "Blue")]

public Color EndColor

{

get { return endcolor; }

set

{

endcolor = value;

onChange();

}

}

public void onChange()

{

Invalidate();

}

public string onEnableStatmentVoid()

{

return "8986838.jpeg";

}

[Category("Трансформация"), Description("Изменение размера"), DefaultValue(typeof(bool), "true")]

public bool DynamicSize

{

get { return changebutton; }

set

{

changebutton = value;

}

}

protected override void OnMouseEnter(EventArgs e)

{

if (changebutton == true)

{

base.OnMouseEnter(e);

this.Width += 20;

this.Height += 10;

this.Font = new System.Drawing.Font(fontname, size + 1.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.Location = new System.Drawing.Point(Location.X - 10, Location.Y - 5);

}

}

protected override void OnMouseLeave(EventArgs e)

{

if (changebutton == true)

{

base.OnMouseLeave(e);

this.Width -= 20;

this.Height -= 10;

this.Font = new System.Drawing.Font(fontname, size, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.Location = new System.Drawing.Point(Location.X + 10, Location.Y + 5);

}

}

}

}

Листинг основной формы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MyControl;

namespace MyControlTest

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

timer1.Enabled = true;

}

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

{

this.Text = "MyControl instances: " + MyControl.MyControl.InstanceCount;

}

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

{

MyControl.MyControl cd;

int ct;

for (ct = 0; ct < 1000; ct++)

cd = new MyControl.MyControl();

MyControl.PictureBoxControl dd;

dd = new MyControl.PictureBoxControl();

pictureBox1.Load(dd.onEnableStatmentVoid());

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

MyControl.PictureBoxControl cd;

cd = new MyControl.PictureBoxControl();

cd.onChange();

}

}

}

Тестирование приложения:



Рисунок 30 – Демонстрация программы

3 Программирование быстродействующих информационных систем

Можно быстро и легко отображать данные из локального файла базы данных в приложении. Для этого требуется создать набор данных и добавить в приложение элементы управления с привязкой к данным.

Первое, что нужно сделать при работе с поставщиком данных — это установить сеанс с источником данных с помощью объекта подключения (порожденного, как вы помните, от DbConnection). У объектов подключения .NET имеется форматированная строка подключения, которая содержит ряд пар имя/значение, разделенных точками с запятой. Эта информация содержит имя машины, к которой нужно подключиться, необходимые параметры безопасности, имя базы данных на этой машине и другую информацию, зависящую от поставщика.

Условие задачи 1: Разработать программу с базой данных по заданной предметной области:

**Распределение дополнительных обязанностей**

Описание предметной области

Вы работаете в коммерческой компании и занимаетесь распределением дополнительных разовых работ. Вашей задачей является отслеживание хода выполнения дополнительных работ.

Компания имеет определенный штат сотрудников, каждый из которых получает определенный оклад. Время от времени, возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных должностных обязанностей сотрудников. Для наведения порядка в этой сфере деятельности Вы проклассифицировали все виды дополнительных работ, определившись с суммой оплаты по факту их выполнения. При возникновении дополнительной работы определенного вида Вы назначаете ответственного, фиксируя дату начала. По факту окончания Вы фиксируете дату и выплачиваете дополнительную сумму к зарплате с учетом Вашей классификации.

**Таблицы**

Сотрудники (Код сотрудника, Фамилия, Имя, Отчество, Оклад).

Виды работ (Код вида, Описание, Оплата за день).

Работы (Код сотрудника, Код вида, Дата начала, Дата окончания)

Алгоритм решения: Кнопки удаления возле соответствующих таблиц отправляют запрос на удаление выделенного поля из базы данных. Кнопки «Добавить» и «Изменить» открывают новую форму для внесения изменений в СУБД MySQL при помощи компонентов TextBox; «Высчитать зарплату» получает данные работника размер из его введенного номера при условии, что у него имеются сверхурочные работы: находится оклад, его и всех количество работ, высчитывается количество дней, по которым находится общая оплата за сверхурочные, которые после суммируются и выводятся в виде сообщения и на компоненте TextBox. Поля по правую сторону от таблицы выводят записи с совпадающими значениями искомой колонки.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace WindowsFormsApp1

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Листинг главной формы:

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.Data.SqlClient;

using MySql.Data;

using MySql.Data.MySqlClient;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public string IDIndex;

private void Form1\_Load(object sender, EventArgs e)

{

SelectDataFromDB();

}

public void RefreshSQL()

{

SelectDataFromDB();

}

private void SelectDataFromDB()

{

String SQLWorkers = GetSQLStringWorkers();

String SQLTypeOfWork = GetSQLStringTypeOfWork();

String SQLWorks = GetSQLStringWorks();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLWorkers, Connection);

MySqlDataAdapter AdapterTypeOfWork = new MySqlDataAdapter(SQLTypeOfWork, Connection);

MySqlDataAdapter AdapterWorks = new MySqlDataAdapter(SQLWorks, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Workers");

dataGridView1.DataSource = ds.Tables["Workers"];

AdapterTypeOfWork.Fill(ds, "TypeOfWork");

dataGridView2.DataSource = ds.Tables["TypeOfWork"];

AdapterWorks.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Код";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Оклад";

dataGridView2.Columns[0].HeaderText = "ID";

dataGridView2.Columns[1].HeaderText = "Код";

dataGridView2.Columns[2].HeaderText = "Описание";

dataGridView2.Columns[3].HeaderText = "Оплата за день";

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private string GetSQLStringWorkers()

{

string sSQL = "SELECT ID\_Workers, Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers FROM Workers";

return sSQL;

}

private string GetSQLStringTypeOfWork()

{

string sSQL = "SELECT ID\_TypeOfWork, Code\_TypeOfWork, Description\_TypeOfWork, Payment\_TypeOfWork FROM TypeOfWork";

return sSQL;

}

private string GetSQLStringWorks()

{

string sSQL = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork";

return sSQL;

}

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

{

AddChangeForm addForm = new AddChangeForm();

addForm.iIdRecordWorkers = 0;

addForm.iIdRecordTypeOfWork = 0;

addForm.iIdRecordWorks = 0;

if (addForm.ShowDialog() == DialogResult.OK)

SelectDataFromDB();

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.CurrentCell != null)

{

AddChangeForm addForm = new AddChangeForm();

addForm.iIdRecordWorkers = Convert.ToInt32(dataGridView1.CurrentCell.Value);

addForm.iIdRecordTypeOfWork = Convert.ToInt32(dataGridView2.CurrentCell.Value);

addForm.iIdRecordWorks = Convert.ToInt32(dataGridView3.CurrentCell.Value);

if (addForm.ShowDialog() == DialogResult.OK)

SelectDataFromDB();

}

}

private void button3\_Click(object sender, EventArgs e)

{

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection ConnectionSQL = new MySqlConnection(sConnectionString);

MySqlCommand cmd = new MySqlCommand("DELETE FROM Workers WHERE ID\_Workers=" + dataGridView1.CurrentCell.Value.ToString(), ConnectionSQL);

ConnectionSQL.Open();

cmd.ExecuteNonQuery();

ConnectionSQL.Close();

SelectDataFromDB();

MessageBox.Show("Удалено.");

}

private void dataGridView1\_MouseClick(object sender, MouseEventArgs e)

{

IDIndex = dataGridView1.CurrentCell.Value.ToString();

}

private void button4\_Click(object sender, EventArgs e)

{

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection ConnectionSQL = new MySqlConnection(sConnectionString);

MySqlCommand cmd = new MySqlCommand("DELETE FROM TypeOfWork WHERE ID\_TypeOfWork = " + dataGridView2.CurrentCell.Value.ToString(), ConnectionSQL);

ConnectionSQL.Open();

cmd.ExecuteNonQuery();

ConnectionSQL.Close();

SelectDataFromDB();

MessageBox.Show("Удалено.");

}

private void button5\_Click(object sender, EventArgs e)

{

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection ConnectionSQL = new MySqlConnection(sConnectionString);

MySqlCommand cmd = new MySqlCommand("DELETE FROM Works WHERE ID\_Works = " + dataGridView3.CurrentCell.Value.ToString(), ConnectionSQL);

ConnectionSQL.Open();

cmd.ExecuteNonQuery();

ConnectionSQL.Close();

SelectDataFromDB();

MessageBox.Show("Удалено.");

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

String SQLWorkers = "SELECT ID\_Workers, Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers FROM Workers WHERE LastName\_Wrokers like '%" + textBox2.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLWorkers, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Workers");

dataGridView1.DataSource = ds.Tables["Workers"];

Connection.Close();

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Код";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Оклад";

String SQLTypeOfWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND Workers.LastName\_Wrokers like '%" + textBox2.Text + "%'";

MySqlConnection Connection1 = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers1 = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds1 = new DataSet();

AdapterWorkers.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

String SQLWorkers = "SELECT ID\_Workers, Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers FROM Workers WHERE Code\_Workers like '%" + textBox6.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLWorkers, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Workers");

dataGridView1.DataSource = ds.Tables["Workers"];

Connection.Close();

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Код";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Оклад";

String SQLTypeOfWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND Workers.Code\_Workers like '%" + textBox6.Text + "%'";

MySqlConnection Connection1 = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers1 = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds1 = new DataSet();

AdapterWorkers.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

String SQLWorkers = "SELECT ID\_Workers, Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers FROM Workers WHERE FirstName\_Workers like '%" + textBox3.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLWorkers, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Workers");

dataGridView1.DataSource = ds.Tables["Workers"];

Connection.Close();

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Код";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Оклад";

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

String SQLWorkers = "SELECT ID\_Workers, Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers FROM Workers WHERE Patronymic\_Workers like '%" + textBox4.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLWorkers, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Workers");

dataGridView1.DataSource = ds.Tables["Workers"];

Connection.Close();

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Код";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Оклад";

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

String SQLWorkers = "SELECT ID\_Workers, Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers FROM Workers WHERE Salary\_Workers like '%" + textBox5.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLWorkers, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Workers");

dataGridView1.DataSource = ds.Tables["Workers"];

Connection.Close();

dataGridView1.Columns[0].HeaderText = "ID";

dataGridView1.Columns[1].HeaderText = "Код";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Оклад";

String SQLTypeOfWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND Workers.Salary\_Workers like '%" + textBox5.Text + "%'";

MySqlConnection Connection1 = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers1 = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds1 = new DataSet();

AdapterWorkers.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void textBox7\_TextChanged(object sender, EventArgs e)

{

String SQLTypeOfWorks = "SELECT ID\_TypeOfWork, Code\_TypeOfWork, Description\_TypeOfWork, Payment\_TypeOfWork FROM TypeOfWork WHERE Code\_TypeOfWork like '%" + textBox7.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "TypeOfWorks");

dataGridView2.DataSource = ds.Tables["TypeOfWorks"];

Connection.Close();

dataGridView2.Columns[0].HeaderText = "ID";

dataGridView2.Columns[1].HeaderText = "Код";

dataGridView2.Columns[2].HeaderText = "Описание";

dataGridView2.Columns[3].HeaderText = "Оплата за день";

String SQLWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND TypeOfWork.Code\_TypeOfWork like '%" + textBox7.Text + "%'";

MySqlConnection Connection1 = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers1 = new MySqlDataAdapter(SQLWorks, Connection);

Connection.Open();

DataSet ds1 = new DataSet();

AdapterWorkers1.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void textBox8\_TextChanged(object sender, EventArgs e)

{

String SQLTypeOfWorks = "SELECT ID\_TypeOfWork, Code\_TypeOfWork, Description\_TypeOfWork, Payment\_TypeOfWork FROM TypeOfWork WHERE Description\_TypeOfWork like '%" + textBox8.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "TypeOfWorks");

dataGridView2.DataSource = ds.Tables["TypeOfWorks"];

Connection.Close();

dataGridView2.Columns[0].HeaderText = "ID";

dataGridView2.Columns[1].HeaderText = "Код";

dataGridView2.Columns[2].HeaderText = "Описание";

dataGridView2.Columns[3].HeaderText = "Оплата за день";

}

private void textBox9\_TextChanged(object sender, EventArgs e)

{

String SQLTypeOfWorks = "SELECT ID\_TypeOfWork, Code\_TypeOfWork, Description\_TypeOfWork, Payment\_TypeOfWork FROM TypeOfWork WHERE Payment\_TypeOfWork like '%" + textBox9.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "TypeOfWorks");

dataGridView2.DataSource = ds.Tables["TypeOfWorks"];

Connection.Close();

dataGridView2.Columns[0].HeaderText = "ID";

dataGridView2.Columns[1].HeaderText = "Код";

dataGridView2.Columns[2].HeaderText = "Описание";

dataGridView2.Columns[3].HeaderText = "Оплата за день";

String SQLWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND TypeOfWork.Payment\_TypeOfWork like '%" + textBox9.Text + "%'";

MySqlConnection Connection1 = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers1 = new MySqlDataAdapter(SQLWorks, Connection);

Connection.Open();

DataSet ds1 = new DataSet();

AdapterWorkers1.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void textBox12\_TextChanged(object sender, EventArgs e)

{

String SQLTypeOfWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND StartDate\_Works like '%" + textBox12.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void textBox13\_TextChanged(object sender, EventArgs e)

{

String SQLTypeOfWorks = "SELECT ID\_Works, Workers.Code\_Workers, Workers.LastName\_Wrokers, Workers.Salary\_Workers, TypeOfWork.Code\_TypeOfWork, TypeOfWork.Payment\_TypeOfWork, StartDate\_Works, EndDate\_Works FROM Works, Workers, TypeOfWork WHERE Works.Code\_Workers = Workers.Code\_Workers AND Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND EndDate\_Works like '%" + textBox13.Text + "%'";

SelectDataFromDB();

string sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection Connection = new MySqlConnection(sConnectionString);

MySqlDataAdapter AdapterWorkers = new MySqlDataAdapter(SQLTypeOfWorks, Connection);

Connection.Open();

DataSet ds = new DataSet();

AdapterWorkers.Fill(ds, "Works");

dataGridView3.DataSource = ds.Tables["Works"];

Connection.Close();

dataGridView3.Columns[0].HeaderText = "ID";

dataGridView3.Columns[1].HeaderText = "Код работника";

dataGridView3.Columns[2].HeaderText = "Фамилия";

dataGridView3.Columns[3].HeaderText = "Оклад";

dataGridView3.Columns[4].HeaderText = "Код работы";

dataGridView3.Columns[5].HeaderText = "Оплата за день";

dataGridView3.Columns[6].HeaderText = "Начало работ";

dataGridView3.Columns[7].HeaderText = "Окончание работ";

}

private void button6\_Click(object sender, EventArgs e)

{

double Oklad = 0, Payment = 0, AllSalary = 0;

int StrCount = 0, StrCountAll = 0, ii = 0;

DateTime StartWorks;

DateTime EndWorks = new DateTime(2000, 1, 1);

MySqlConnection Connection = new MySqlConnection("server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;");

object Code\_Workers = "";

MySqlCommand command = new MySqlCommand("SELECT Code\_Workers FROM Works WHERE Code\_Workers = " + textBox1.Text, Connection);

Connection.Open();

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Code\_Workers = reader.GetValue(0);

}

reader.Close();

Connection.Close();

if (Code\_Workers == "")

{

MessageBox.Show("У работника нет сверхурочных");

}

else

{

command = new MySqlCommand("SELECT Salary\_Workers FROM Workers WHERE Code\_Workers = " + Code\_Workers, Connection);

Connection.Open();

reader = command.ExecuteReader();

while (reader.Read())

{

Oklad = Convert.ToDouble(reader.GetValue(0));

}

reader.Close();

Connection.Close();

MessageBox.Show("Oklad = " + Convert.ToString(Oklad));

command = new MySqlCommand("SELECT count(\*) FROM Works WHERE Code\_Workers = " + Code\_Workers, Connection);

Connection.Open();

reader = command.ExecuteReader();

while (reader.Read())

{

StrCount = Convert.ToInt32(reader.GetValue(0));

}

reader.Close();

Connection.Close();

MessageBox.Show("Количество работ = " + Convert.ToString(StrCount));

command = new MySqlCommand("select \* from Works where ID\_Works=(select max(ID\_Works) from Works) ", Connection);

Connection.Open();

reader = command.ExecuteReader();

while (reader.Read())

{

StrCountAll = Convert.ToInt32(reader.GetValue(0));

}

reader.Close();

MessageBox.Show("Количество всех работ = " + Convert.ToString(StrCountAll));

double[] SumPayment = new double[StrCount];

int IndexPayment = 0;

while (ii < StrCountAll)

{

int foo = 0;

next: if (foo == 1)

{

ii++;

continue;

}

ii++;

command = new MySqlCommand("SELECT StartDate\_Works FROM Works WHERE ID\_Works = " + (ii + 1) +

" AND Code\_Workers = " + Code\_Workers, Connection);

reader = command.ExecuteReader();

while (reader.Read())

{

if (reader.GetValue(0) != "")

{

foo = 0;

StartWorks = Convert.ToDateTime(reader.GetValue(0));

Connection.Close();

MessageBox.Show("Начальная дата = " + Convert.ToString(StartWorks));

command = new MySqlCommand("SELECT EndDate\_Works FROM Works WHERE ID\_Works = " + (ii + 1) +

" AND Code\_Workers = " + Code\_Workers, Connection);

Connection.Open();

MySqlDataReader reader2 = command.ExecuteReader();

while (reader2.Read())

{

EndWorks = Convert.ToDateTime(reader2.GetValue(0));

}

reader2.Close();

MessageBox.Show("Конечная дата = " + Convert.ToString(EndWorks));

int TotalDays = ((TimeSpan)(EndWorks - StartWorks)).Days + 1;

MessageBox.Show("Итого дней = " + Convert.ToString(TotalDays));

command = new MySqlCommand("SELECT TypeOfWork.Payment\_TypeOfWork FROM Works, TypeOfWork WHERE Works.Code\_TypeOfWork = TypeOfWork.Code\_TypeOfWork AND ID\_Works = " + (ii + 1) +

" AND Code\_Workers = " + Code\_Workers, Connection);

reader2 = command.ExecuteReader();

while (reader2.Read())

{

Payment = Convert.ToDouble(reader2.GetValue(0));

}

reader2.Close();

MessageBox.Show("Оплата за день = " + Convert.ToString(Payment));

SumPayment[IndexPayment] = Convert.ToDouble(TotalDays) \* Payment;

MessageBox.Show("Оплата за все дни работы = " + Convert.ToString(SumPayment[IndexPayment]));

IndexPayment++;

goto next;

}

else

{

foo = 1;

goto next;

}

}

reader.Close();

}

Connection.Close();

AllSalary += Oklad;

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

AllSalary += + SumPayment[i];

MessageBox.Show("Общие начисления = " + AllSalary);

textBox1.Text = Convert.ToString(AllSalary);

}

}

}

}

Листинг формы редактирования:

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.Data.SqlClient;

using MySql.Data.MySqlClient;

namespace WindowsFormsApp1

{

public partial class AddChangeForm : Form

{

public int iIdRecordWorkers { get; set; }

public int iIdRecordTypeOfWork { get; set; }

public int iIdRecordWorks { get; set; }

public AddChangeForm()

{

InitializeComponent();

}

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

{

Form1 f1 = new Form1();

string sSQL = "";

String sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection ConnectionSQL = new MySqlConnection(sConnectionString);

if (iIdRecordWorkers == 0)

sSQL = GetInsertSQLWorkers();

else

{

sSQL = GetUpdateSQLWorkers(iIdRecordWorkers);

}

MySqlCommand cmd = new MySqlCommand(sSQL, ConnectionSQL);

ConnectionSQL.Open();

cmd.ExecuteNonQuery();

f1.RefreshSQL();

ConnectionSQL.Close();

}

private string GetInsertSQLWorkers()

{

string sSQL = "INSERT INTO Workers (Code\_Workers, LastName\_Wrokers, FirstName\_Workers, Patronymic\_Workers, Salary\_Workers) VALUES (";

sSQL += textBox1.Text + ",";

sSQL += "'" + textBox2.Text + "', ";

sSQL += "'" + textBox3.Text + "', ";

sSQL += "'" + textBox4.Text + "', ";

sSQL += textBox5.Text + ")";

return sSQL;

}

private string GetUpdateSQLWorkers(int ID\_Workers)

{

string sSQL = "UPDATE Workers SET Code\_Workers = ";

sSQL += textBox1.Text + ",";

sSQL += "LastName\_Wrokers = '" + textBox2.Text + "',";

sSQL += "FirstName\_Workers = '" + textBox3.Text + "',";

sSQL += "Patronymic\_Workers = '" + textBox4.Text + "',";

sSQL += "Salary\_Workers = " + textBox5.Text + " ";

sSQL += "WHERE ID\_Workers = " + ID\_Workers;

return sSQL;

}

private void button3\_Click(object sender, EventArgs e)

{

Form1 f1 = new Form1();

string sSQL = "";

String sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection ConnectionSQL = new MySqlConnection(sConnectionString);

if (iIdRecordTypeOfWork == 0)

sSQL = GetInsertSQLTypeOfWork();

else

{

sSQL = GetUpdateSQLTypeOfWork(iIdRecordTypeOfWork);

}

MySqlCommand cmd = new MySqlCommand(sSQL, ConnectionSQL);

ConnectionSQL.Open();

cmd.ExecuteNonQuery();

f1.RefreshSQL();

ConnectionSQL.Close();

}

private string GetInsertSQLTypeOfWork()

{

string sSQL = "INSERT INTO TypeOfWork (Code\_TypeOfWork , Description\_TypeOfWork, Payment\_TypeOfWork) VALUES (";

sSQL += textBox6.Text + ",";

sSQL += "'" + textBox7.Text + "', ";

sSQL += textBox8.Text + ")";

return sSQL;

}

private string GetUpdateSQLTypeOfWork(int ID\_Workers)

{

string sSQL = "UPDATE TypeOfWork SET Code\_TypeOfWork = ";

sSQL += textBox6.Text + ", ";

sSQL += "Description\_TypeOfWork = '" + textBox7.Text + "',";

sSQL += "Payment\_TypeOfWork = " + textBox8.Text + " ";

sSQL += "WHERE ID\_TypeOfWork = " + ID\_Workers;

return sSQL;

}

private void button2\_Click(object sender, EventArgs e)

{

Form1 f1 = new Form1();

string sSQL = "";

String sConnectionString = "server=localhost;user=mysql;database=MyCompanyPractica;port=3306;password=mysql;";

MySqlConnection ConnectionSQL = new MySqlConnection(sConnectionString);

if (iIdRecordTypeOfWork == 0)

sSQL = GetInsertSQLWorks();

else

{

sSQL = GetUpdateSQLWorks(iIdRecordWorks);

}

MySqlCommand cmd = new MySqlCommand(sSQL, ConnectionSQL);

ConnectionSQL.Open();

cmd.ExecuteNonQuery();

f1.RefreshSQL();

ConnectionSQL.Close();

}

private string GetInsertSQLWorks()

{

string sSQL = "INSERT INTO Works (Code\_Workers, Code\_TypeOfWork, StartDate\_Works, EndDate\_Works) VALUES (";

sSQL += textBox10.Text + ",";

sSQL += textBox11.Text + ",";

sSQL += "'" + textBox12.Text + "', ";

sSQL += "'" + textBox13.Text + "')";

return sSQL;

}

private string GetUpdateSQLWorks(int ID\_Workers)

{

Form1 f1 = new Form1();

string sSQL = "UPDATE Works SET Code\_Workers = ";

sSQL += textBox10.Text + ", ";

sSQL += "Code\_TypeOfWork = " + textBox11.Text + ", ";

sSQL += "StartDate\_Works = '" + textBox12.Text + "', ";

sSQL += "EndDate\_Works = '" + textBox13.Text + "' ";

sSQL += "WHERE ID\_Works = " + ID\_Workers;

return sSQL;

}

}

}

Тестирование приложения:

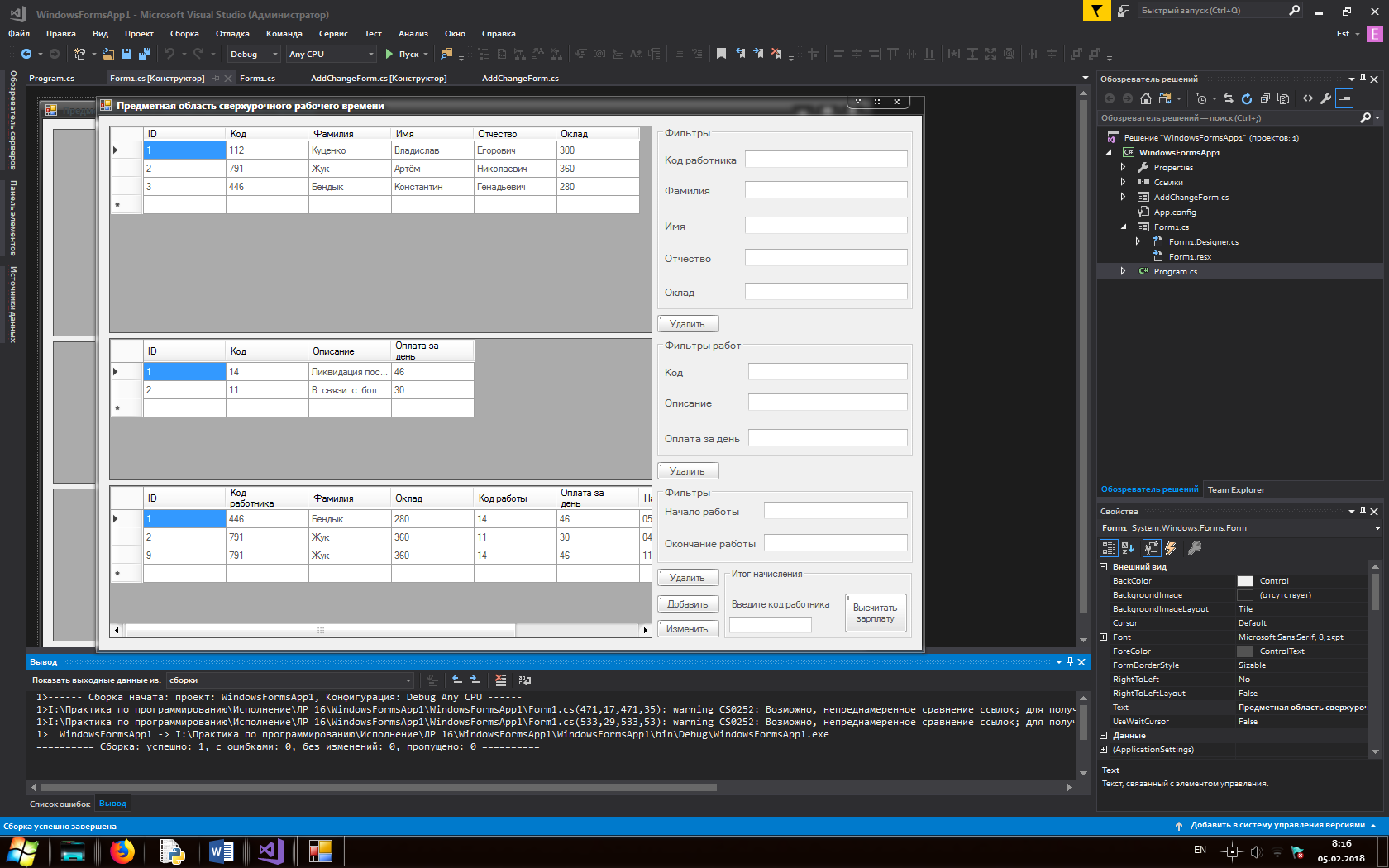


Рисунок 31 – Демонстрация рабочего окна



Рисунок 32 – Демонстрация поиска по фамлии

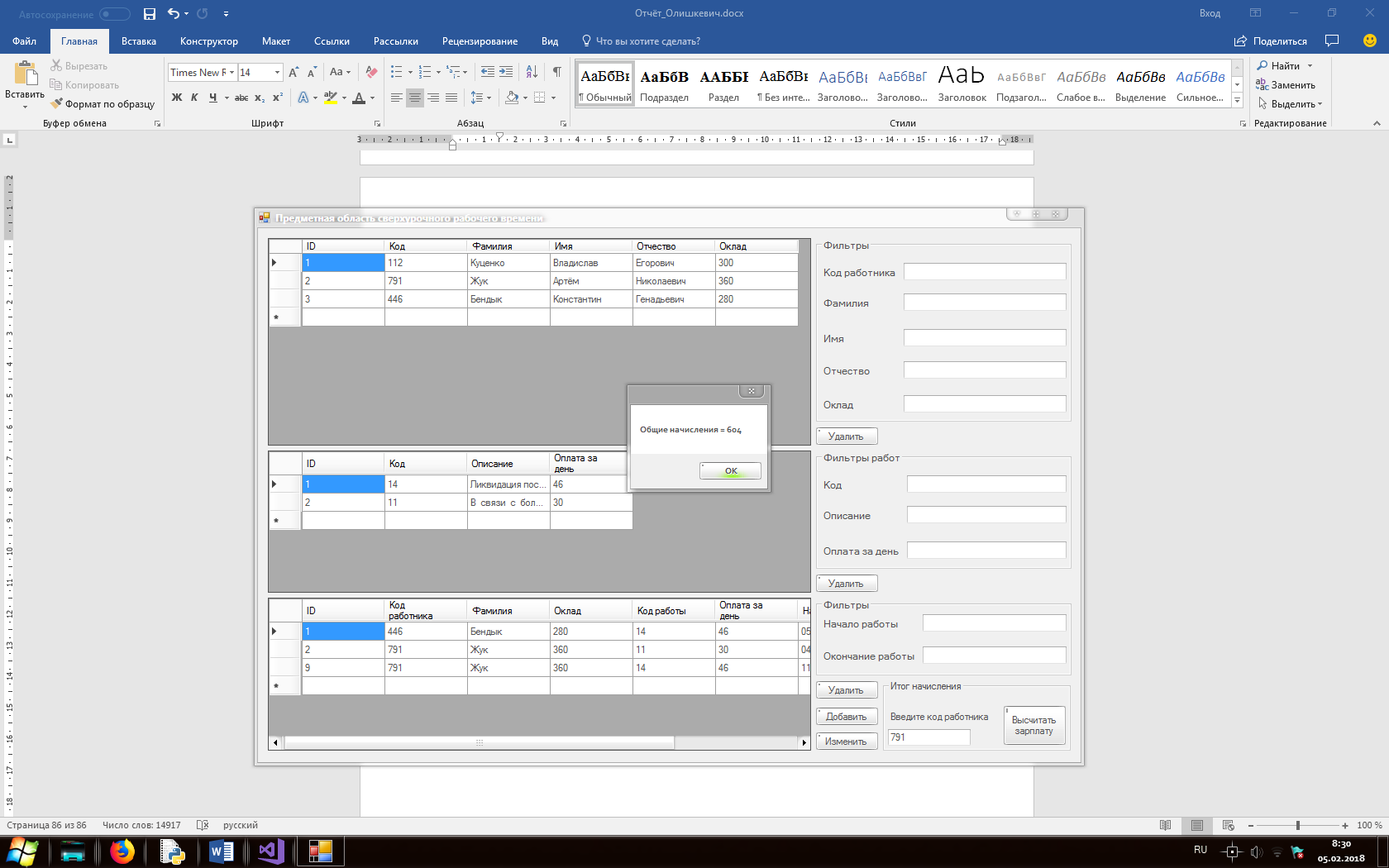


Рисунок 33 – Демонстрация высчета начислений работника

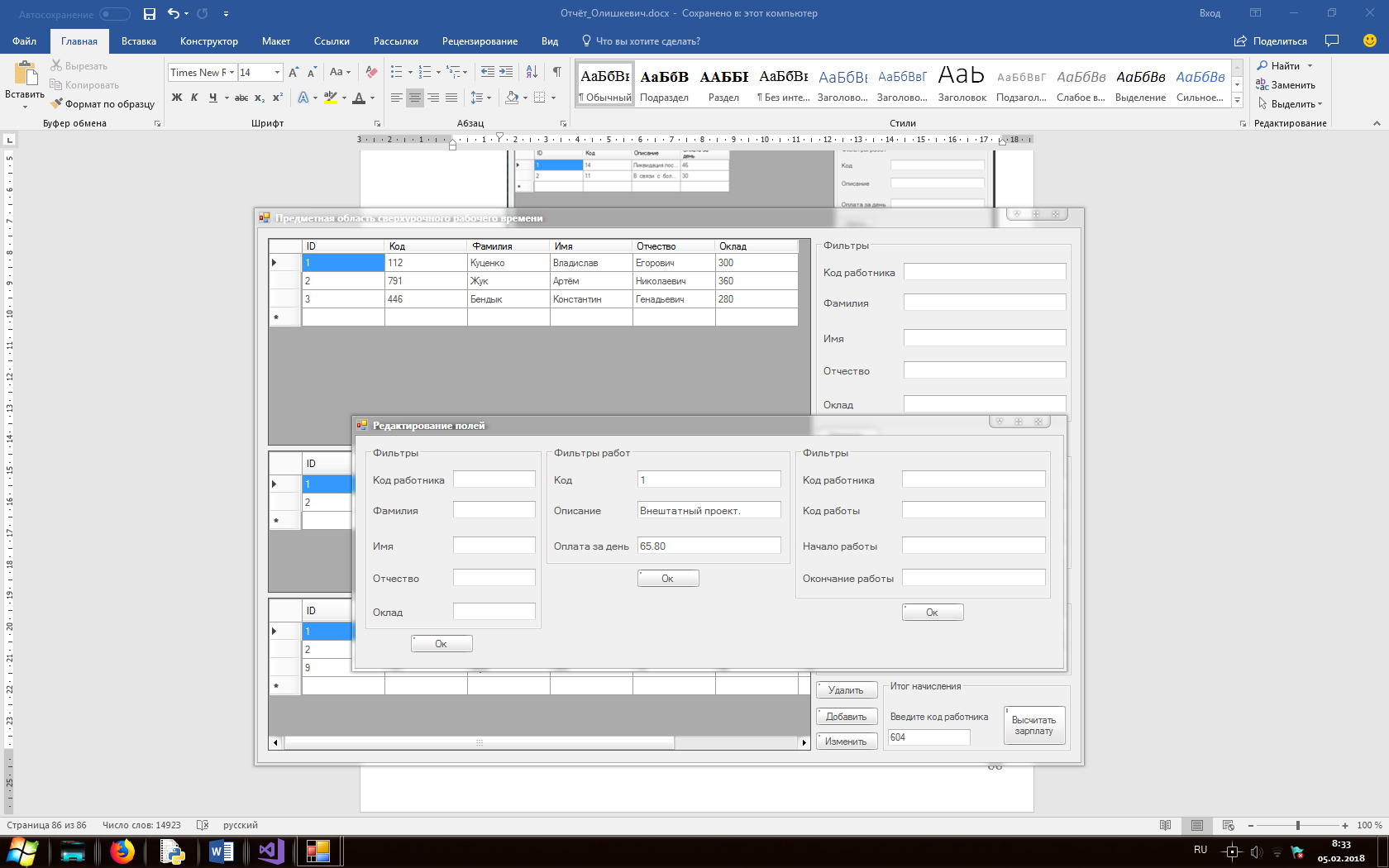


Рисунок 34 – Демонстрация добавления записи

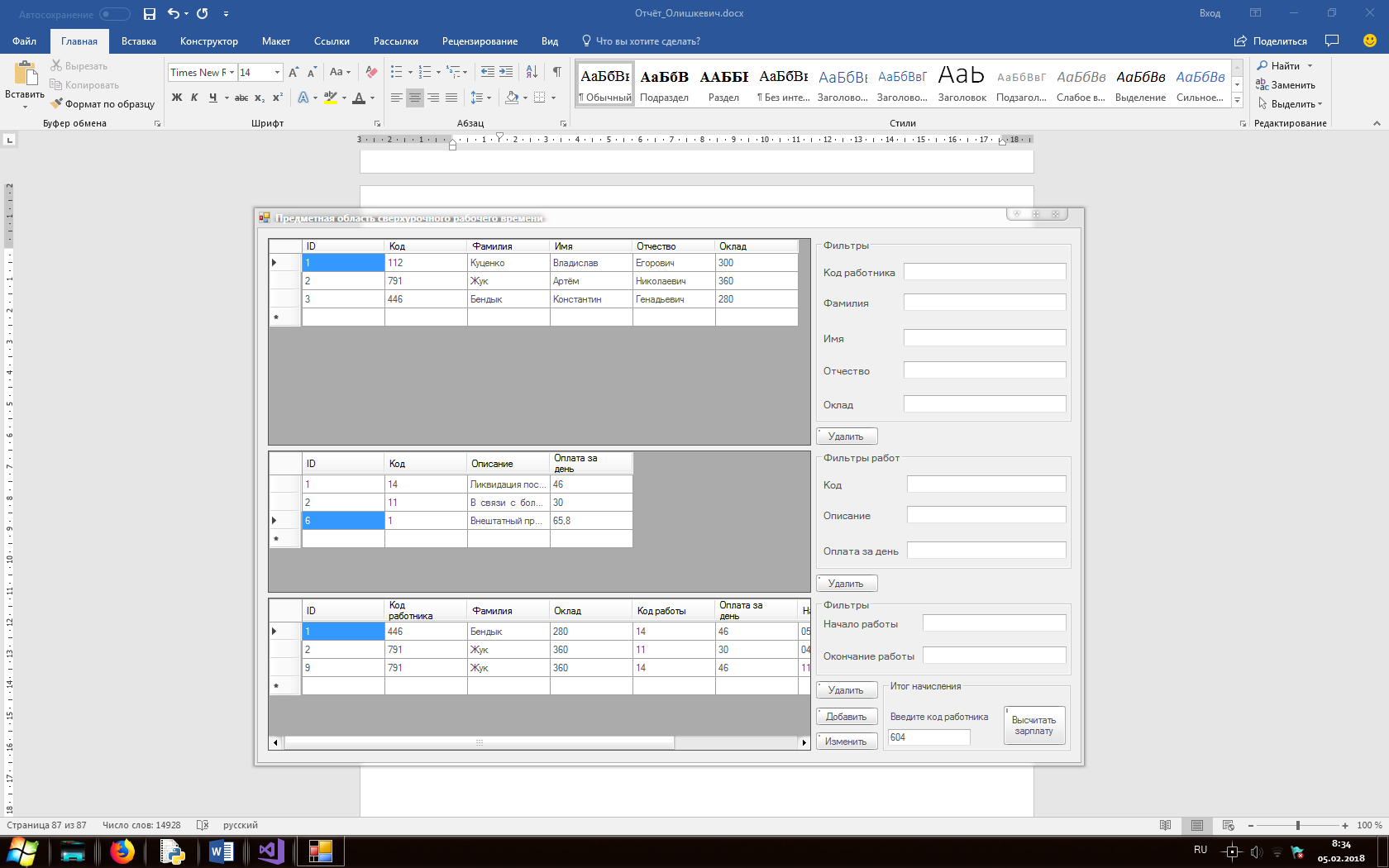


Рисунок 35 – Демонстрация добавленной записи

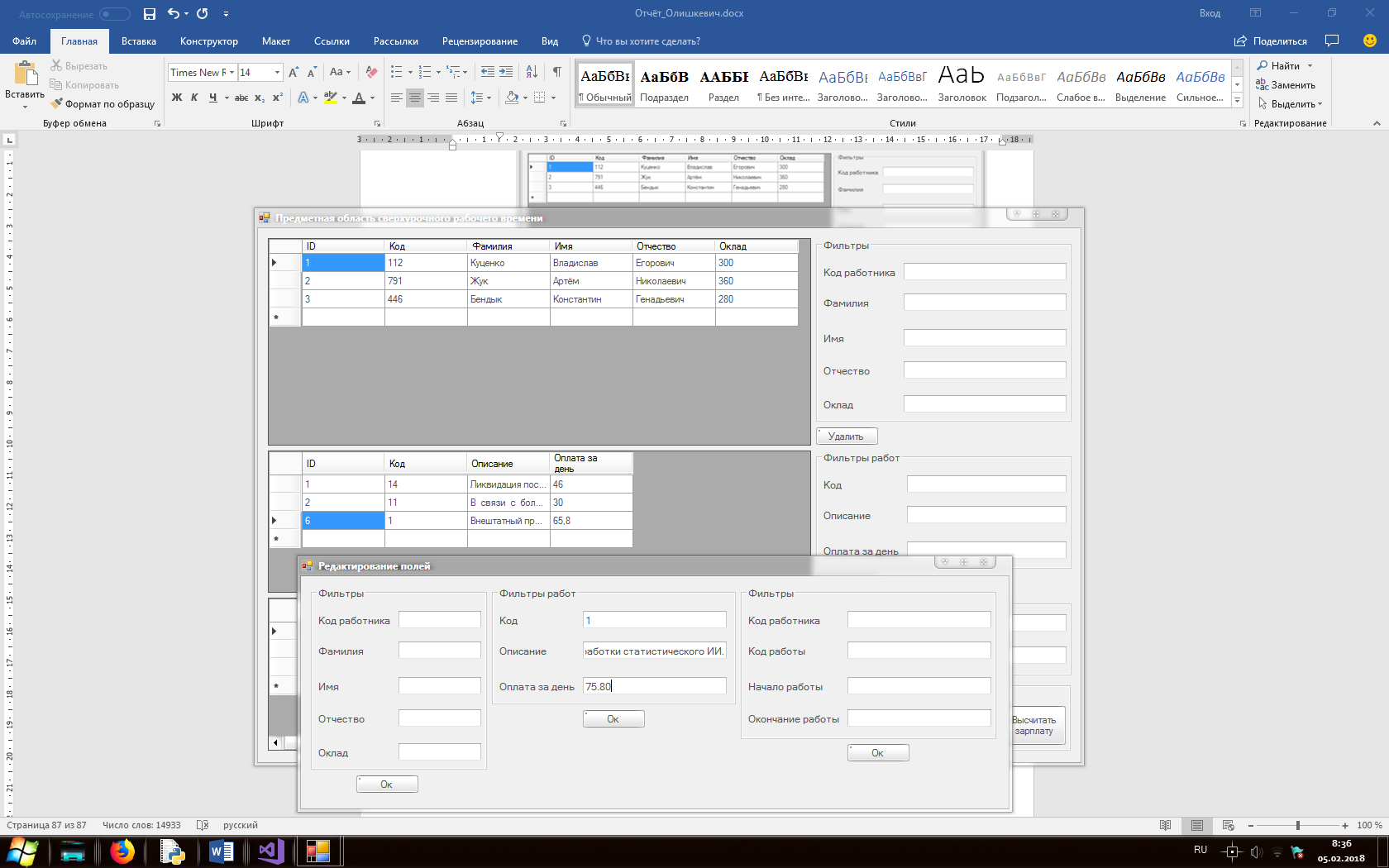


Рисунок 36 – Демонстрация редактирования записи

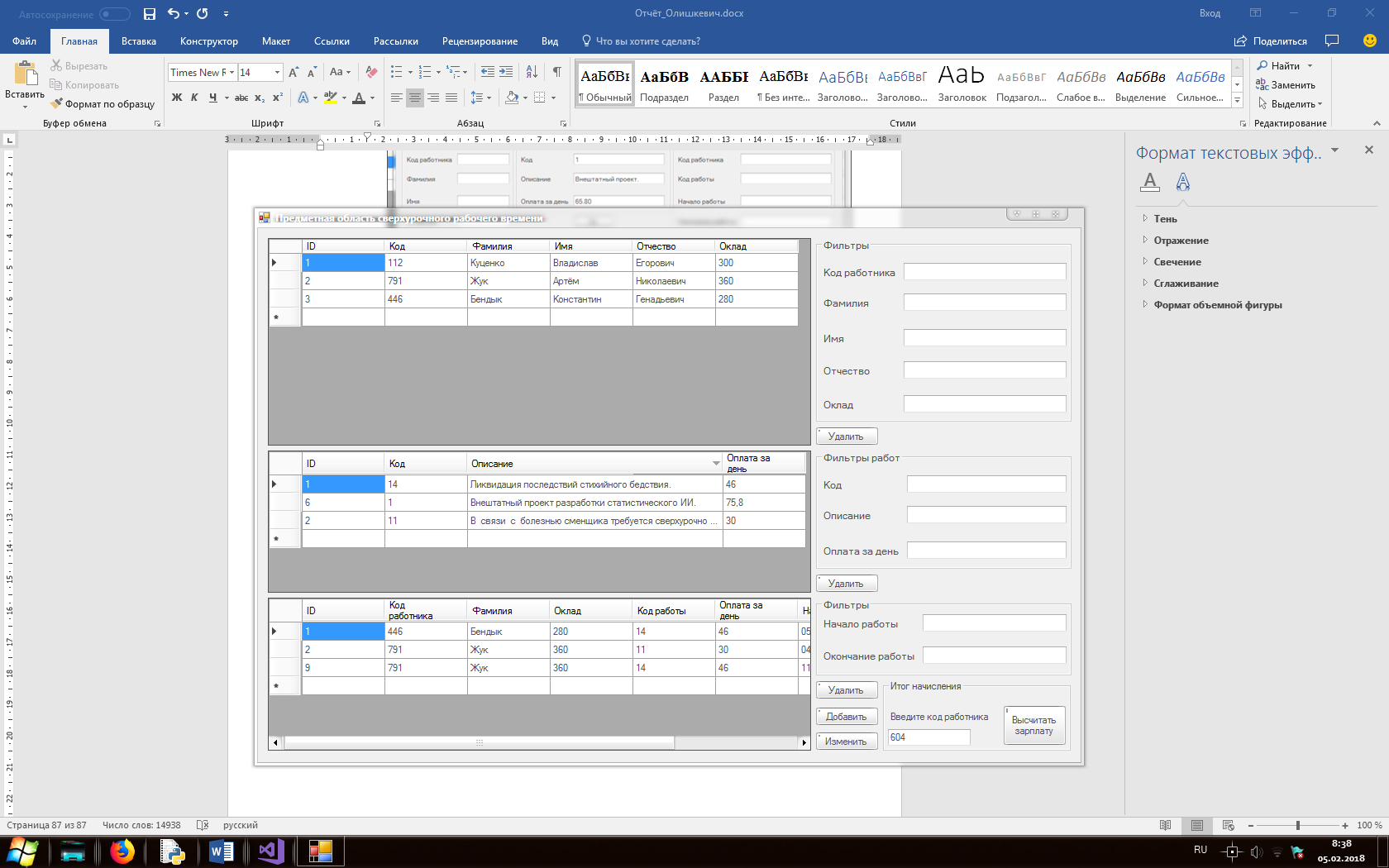


Рисунок 37 – Демонстрация измененной записи

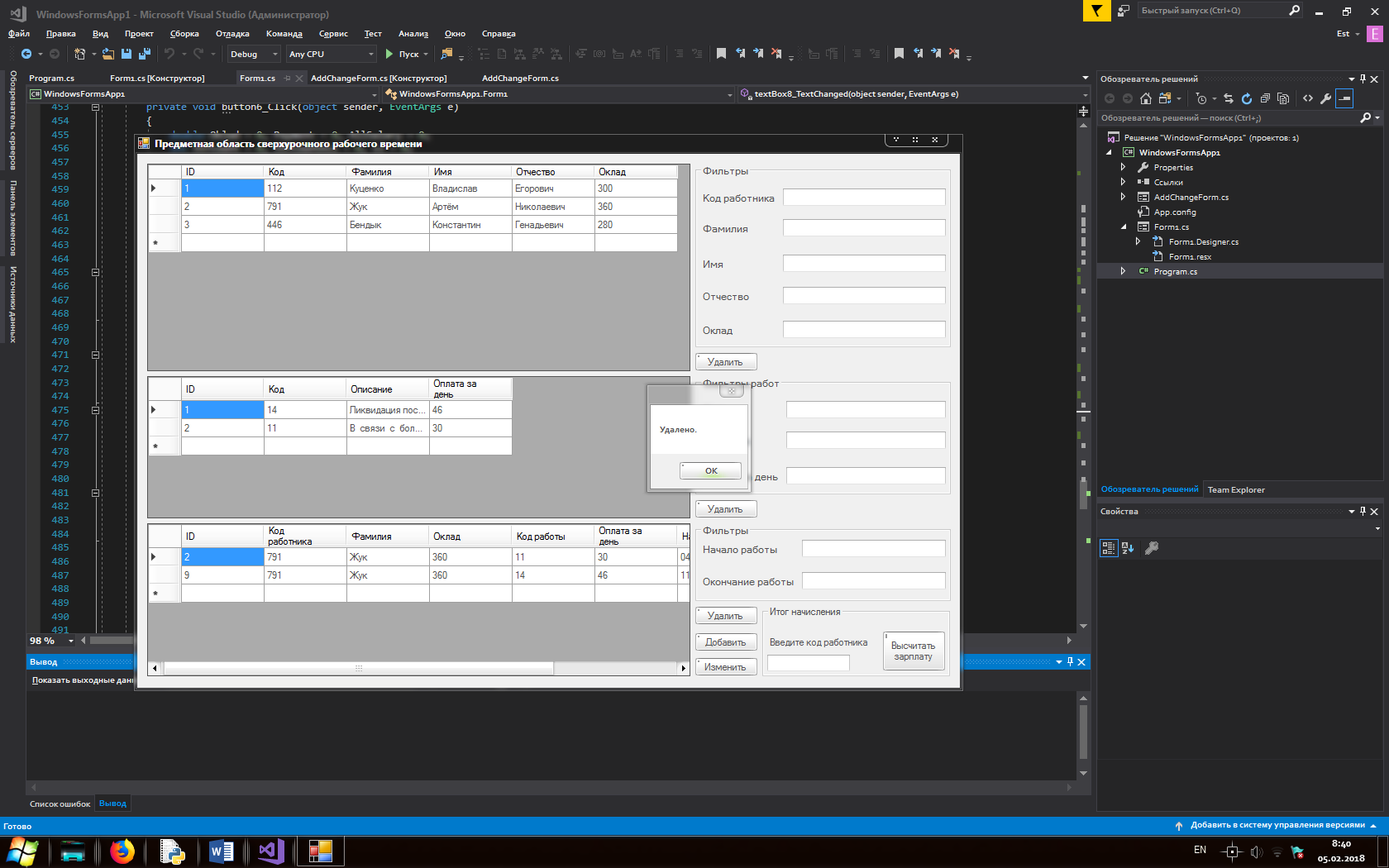


Рисунок 38 – Демонстрация удаленной записи

**ЗАКЛЮЧЕНИЕ**

C# — простой, современный объектно-ориентированный и типобезопасный язык программирования со строгой типизацией. Он относится к широко известному семейству языков C. Освоение C# является актуальным на текущий день и в последующем будущем. Этот язык реализуют множество аспектов программирования, по этой причине востребован и высокооплачиваем. Также следует обратить внимание на то, что изучение языков программирования очень помогает развивать логические и аналитические навыки студента, которые так необходимы для специалиста в сфере IT-технологий.

В период учебной практики изучен язык программирования C#. Этому свидетельствует освоение:

1. базовых приемов обработки строк, приемы работы с массивами и особенности их использования, реализация классов и их механизмов, работы с потоками, файлами и библиотеками. Также изучение коллекций и структур, регулярных выражений;
2. особенностей GUI, работа с графическими компонентами, создание Desktop-приложений;
3. использования СУБД и работа с ним на основе созданных Windows-приложений.

Следует отметить, что весь теоретический материал по курсу практики по программированию был изучен. Об этом свидетельствуют успешно выполненные задания, позволившие освоить и закрепить навыки в программировании на языке C#.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Фленов, М. Библия C# 3-е изд./ Фленов М. – Издательство «БХВ-Петербург», 2016.

2 Шарп, Д. Microsoft Visual C#. Подробное руководство 8-е изд./ Шарп Д. – Питер, 2017.

3 Зиборов, В.В. Visual С# 2012 на примерах – БХВ-Петербург, 2013.

4 Рубанцев, В. Тотальный тренинг по Си-шарпу – Питер, 2012.

5 Виссер, Дж. Building Maintainable Software (C# Edition) – Издательство ДМК Пресс, 2017.

6 Шилдт, Г. C# 4.0. Полное руководство – Издательство «Вильямс», 2015.