**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**по МДК.01.01**

Студент: Давудова Д.И.

Группа: ИСПП-31

Преподаватель: Маломан Ю.С.

Архангельск 2025

**Лабораторная работа №1**

**Оценка сложности алгоритмов сортировки**

**1 Цель работы**

1.1 Научиться реализовывать и оценивать сложность алгоритмов сортировки массивов на C#.

**2 Контрольные вопросы**

2.1 Что такое «массив»?

Массив — это структура данных, которая хранит упорядоченный набор значений

2.2 Как описывается одномерный массив?

тип[] имя = new тип[количество элементов];

2.3 Как обратиться к некоторому элементу одномерного массива?

имя\_массива[индекс]

2.4 Как можно задать одномерный массив?

int[] numbers = { 1, 2, 3, 4, 5 };

2.5 Что такое «сортировка»?

Сортировка - это процесс упорядочивания элементов конечного множества по определённому правилу

2.6 Что такое «алгоритм сортировки»?

Алгоритм сортировки – это последовательность действий, направленная на упорядочивание элементов в заданном множестве по определённому правилу

2.7 Какие виды сортировки массивов существуют?

Сортировка методом простого выбора, сортировка методом «пузырька», сортировка вставками

**3 Вывод**

В ходе лабораторной работы я научилась реализовывать и оценивать сложность алгоритмов сортировки массивов на C#.

**Лабораторная работа №2**

**Оценка сложности алгоритмов поиска**

**1 Цель работы**

1.1 Научиться реализовывать и оценивать сложность алгоритмов поиска элементов массивов на C#.

**2 Контрольные вопросы**

2.1 Что такое «алгоритм сортировки»?

Алгоритм сортировки – это последовательность действий, направленная на упорядочивание элементов в заданном множестве по определённому правилу

2.2 Какие виды поиска элементов массивов существуют?

Линейный поиск, двоичный поиск, алгоритм поиска прыжками

2.3 В чем особенность алгоритма линейного поиска и какова его временная сложность?

Это простейший алгоритм, который перебирает элементы массива один за другим, пока не найдет искомый. Временная сложность: O(N)

2.4 В чем особенность алгоритма двоичного поиска и какова его временная сложность?

Алгоритм делит входную коллекцию на равные половины, и с каждой итерацией сравнивает целевой элемент с элементом в середине. Поиск заканчивается при нахождении элемента. Временная сложность: O(log (N))

2.5 В чем особенность алгоритма поиска прыжками и какова его временная сложность?

Делает "прыжки" фиксированного размера, а затем выполняет линейный поиск в найденном блоке. Временная сложность: O(sqrt(N)).

**3 Вывод**

В ходе лабораторной работы я научилась реализовывать и оценивать сложность алгоритмов поиска элементов массивов на C#.

**Лабораторная работа №3**

**Оценка сложности рекурсивных алгоритмов**

**1 Цель работы**

1.1 Научиться разрабатывать и оценивать сложность рекурсивных функций в программах на C#.

**2 Контрольные вопросы**

2.1 Что такое «рекурсия»?

Рекурсия — это метод, при котором функция вызывает сама себя.

2.2 Какие проблемы могут возникать при реализации рекурсивных алгоритмов на электронных вычислительных машинах?

Переполнение стека, высокое потребление памяти, медленная работа

2.3 Какое определение функции может быть названо рекурсивным?

Функция называется рекурсивной, если имеет условие выхода и вызывает саму себя

Пример:

int Factorial(int n)

{

if (n == 0)

return 1;

return n \* Factorial(n - 1);

}

2.4 Что такое «глубина рекурсии»?

Глубина рекурсии — это максимальное количество уровней вложенных вызовов функции

2.5 Что такое «рекурсивный спуск»?

Рекурсивный спуск - это техника разбора данных, при которой сложная структура обрабатывается с помощью набора взаимно рекурсивных функций.

2.6 Что такое «рекурсивный подъём»?

Рекурсивный подъём — это процесс возврата значений из вложенных рекурсивных вызовов к начальному вызову.

**3 Вывод**

В ходе лабораторной работы я научилась разрабатывать и оценивать сложность рекурсивных функций в программах на C#.

**Лабораторная работа №4**

**Оценка сложности эвристических алгоритмов**

**1 Цель работы**

1.1 Научиться реализовывать, оценивать и применять эвристические алгоритмы.

**2 Контрольные вопросы**

2.1 Что такое «эвристика»?

Эвристика в программировании — это набор методов и подходов, которые используются для решения задач, когда точное решение либо невозможно найти, либо требует чрезмерных вычислительных ресурсов

2.2 Зачем используются эвристические алгоритмы?

Эвристические алгоритмы применяются в случаях, когда: важна

скорость, а не точность, когда точный алгоритм требует слишком много памяти.

2.3 Какие алгоритмы относятся к эвристическим?

Жадные алгоритмы, генетические алгоритмы, метод имитации отжига.

**3 Вывод**

В ходе лабораторной работы я научилась разрабатывать, оценивать и применять эвристические алгоритмы.

**Лабораторная работа №5**

**Работа с классами**

**1 Цель работы**

1.1 Изучить процесс разработки и применения классов на языке C#;

1.2 Изучить реализацию механизма инкапсуляции на языке C#.

**2 Контрольные вопросы**

2.1 Какова общая форма объявления класса в C#?

[модификаторы] class ИмяКласса { }

2.2 На какие виды делятся данные класса в C#?

Данные класса делятся на: поля экземпляра, статические поля, константы, только для чтения.

2.3 На какие виды делятся функции класса в C#?

Функции класса делятся на: методы экземпляра, статические методы, абстрактные.

2.4 Для чего применяются конструкторы классов?

Конструкторы используются для: инициализации объекта при создании, установки начальных значений полей, выполнения кода подготовки объекта.

2.5 Что такое «цепочка конструкторов»?

Цепочка конструкторов – это вызов одного конструктора из другого через this().

2.6 Для чего применяются свойства классов?

Свойства используются для: контроля доступа к полям, валидации данных перед записью.

2.7 Когда используются автоматически реализуемые свойства классов?

Автоматические свойстваприменяются, когда: нужно простое свойство без дополнительной логики, валидация не требуется.

2.8 В чем отличие вызова статические членов классов от членов экземпляра класса?

Статические члены класса и члены экземпляра отличаются способом вызова, областью хранения и доступом к данным.

**3 Вывод**

В ходе лабораторной работы я изучила процесс разработки и применения классов и реализации механизма инкапсуляции на языке C#.

**Лабораторная работа №6**

**Перегрузка методов**

**1 Цель работы**

1.1 Изучить процесс выполнения перегрузки методов на языке C#;

1.2 Изучить различные способы передачи параметров в метод.

**2 Контрольные вопросы**

2.1 Что такое «перегрузка методов»?

Перегрузка методов — это возможность в объектно-ориентированном программировании создавать несколько методов с одним и тем же именем, но с разными параметрами

2.2 К какому виду полиморфизма относится перегрузка методов?

Перегрузка методов относится к статическому полиморфизму.

2.3 Какие типы функций класса можно перегружать?

В C# можно перегружать: обычные методы, конструкторы, операторы.

2.4 Какие требования предъявляются к сигнатуре перегружаемых функций класса?

Требования для перегрузки: разные типы параметров, разное количество параметров, разный порядок типов.

2.5 Какие типы функций класса нельзя перегружать?

Нельзя перегружать: методы, отличающиеся только возвращаемым типом, переопределенные методы

**3 Вывод**

В ходе лабораторной работы я изучила процесс выполнения перегрузки методов и различные способы передачи параметров в метод на языке C#.

**Лабораторная работа №7**

**Определение операций в классе**

**1Цель работы**

1.1 Изучить процесс определения операций в классе на языке C#.

**2 Контрольные вопросы**

2.1 Какое ключевое слово определяет операторный метод?

Ключевое слово operator

2.2 Какие унарные операторы можно перегружать в C#?

Можно перегружать следующие унарные операторы: плюс, минус, логическое НЕ, битовое НЕ, инкремент, декремент, true false

2.3 Какие бинарные операторы можно перегружать в C#?

Можно перегружать следующие бинарные операторы: сложение, вычитание, умножение, деление, остаток от деления, битовое И, битовое ИЛИ, битовые сдвиги

2.4 Какие операторы сравнения можно перегружать в C#?

Можно перегружать следующие операторы сравнения: равно, не равно, больше, меньше, меньше или равно, больше или равно

2.5 Какие операторы требуется перегружать в C# попарно?

Операторы, которые требуется перегружать попарно: равно и не равно, больше и меньше, больше или равно и меньше или равно.

**3 Вывод**

В ходе лабораторной работы я изучила процесс определения операций в классе на языке C#.

**Лабораторная работа №8**

**Создание наследованных классов**

**1 Цель работы**

1.1 Изучить процесс разработки дочерних классов на языке C#;

1.2 Изучить реализацию механизма наследования на языке C#.

**2 Контрольные вопросы**

2.1 Что такое «наследование»?

Наследование — это механизм ООП, при котором один класс может перенимать свойства, методы и поведение другого класса.

2.2 Сколько родительских классов может быть у дочернего класса в C#?

Дочерний класс может наследоваться только от одного родительского класса

2.3 Какое ключевое слово позволяет обратиться к реализации родительского класса из дочернего?

Ключевое слово base используется для вызова конструктора

родителя

2.4 Что такое «переопределение метода» и как оно выполняется?

Переопределение метода — это замена реализации метода родителя в дочернем классе. Нужно в дочернем классе объявите метод с той же сигнатурой, что и в родительском

2.5 Что такое «абстрактный класс»?

Абстрактный класс — это класс, который обозначается ключевым словом abstract и служит основой для других классов. Его главная особенность в том, что нельзя создать экземпляр абстрактного класса напрямую

2.6 Для чего предназначены модификаторы virtual, override, abstract, new?

Virtual — разрешает переопределение метода, override — переопределяет метод родителя, abstract — требует переопределения метода, new — скрывает метод родителя.

**3 Код**

4 Задания

В Customer.cs

public string FullName { get; set; }

public string Address { get; set; }

public decimal SpentAmount { get; set; }

public Customer(string fullName, string address, decimal spentAmount)

{

FullName = fullName;

Address = address;

SpentAmount = spentAmount;

}

public void PrintInfo()

{

Console.WriteLine($"Заказчик: {FullName}, Адрес: {Address}, Потраченная сумма: {SpentAmount}");

}

public override string ToString()

{

return $"Заказчик: {FullName}, Адрес: {Address}, Потраченная сумма: {SpentAmount}";

}

public override bool Equals(object obj)

{

if (obj == null || GetType() != obj.GetType())

return false;

Customer other = (Customer)obj;

return FullName == other.FullName &&

Address == other.Address &&

SpentAmount == other.SpentAmount;

}

В Program.cs

using LabWork8;

Customer customer1 = new("Иванов Алексей", "ул. Мира, 1", 5600);

Customer customer2 = new("Окунь Александр", "ул. Папанина, 18", 1000);

Customer customer3 = new("Щукин Петр", "ул. Ленина, 278", 2400);

Console.WriteLine("Тестирование ToString():");

Console.WriteLine(customer1.ToString());

Console.WriteLine(customer1);

Console.WriteLine("\nТестирование Equals():");

Console.WriteLine($"customer1 и customer2: {customer1.Equals(customer2)}");

Console.WriteLine($"customer1 и customer3: {customer1.Equals(customer3)}");

Square square = new(5);

Console.WriteLine("\nТестирование класса Square:");

square.PrintInfo();

CustomRandom customRandom = new();

Console.WriteLine("\nТестирование CustomRandom:");

Console.WriteLine($"Случайная строка: {customRandom.GetRandomUpperCaseString(10)}");

В Shape.cs

public abstract class Shape

{

public abstract string Name { get; }

public abstract double GetArea();

public abstract double GetPerimeter();

public abstract void PrintInfo();

}

В Square.cs

public class Square : Shape

{

public double Side { get; set; }

public Square(double side)

{

Side = side;

}

public override string Name => "Квадрат";

public override double GetArea()

{

return Side \* Side;

}

public override double GetPerimeter()

{

return 4 \* Side;

}

public override void PrintInfo()

{

Console.WriteLine($"Фигура: {Name}, Сторона: {Side}, Площадь: {GetArea()}, Периметр: {GetPerimeter()}");

}

}

В CustomRandom.cs

public class CustomRandom : Random

{

public string GetRandomUpperCaseString(int length)

{

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

return new string(Enumerable.Repeat(chars, length)

.Select(s => s[Next(s.Length)]).ToArray());

}

}

**4 Вывод**

В ходе лабораторной работы я изучила процесс разработки дочерних классов и реализации механизма наследования на языке C#

**Лабораторная работа №9**

**Работа с объектами через интерфейсы**

**1 Цель работы**

1.1 Изучить процесс разработки и реализации интерфейсов на языке C#.

**2 Контрольные вопросы**

2.1 Что такое «интерфейс»?

Интерфейс — это контракт, который определяет набор методов, свойств, событий или индексаторов, но не содержит их реализации.

2.2 Чем отличается интерфейс от абстрактного класса?

Интерфейс: содержит только абстрактные методы и свойства, класс может реализовывать несколько интерфейсов

Абстрактный класс: может содержать как абстрактные, так и обычные методы, класс может наследовать только один абстрактный класс

2.3 Есть ли у класса ограничения по количеству реализуемых интерфейсов?

Класс в C# может реализовывать сколько угодно интерфейсов.

2.4 Какова общая форма объявления интерфейсов и их элементов?

[модификаторы] interface ИмяИнтерфейса

{

тип Метод(параметры); // Метод

тип Свойство { get; set; } // Свойство

событие Делегат Событие; // Событие

тип this[индекс] { get; set; } // Индексатор

}

**3 Вывод**

В ходе лабораторной работы я изучила процесс разработки и реализации интерфейсов на языке C#

**Лабораторная работа №10**

**Использование стандартных интерфейсов**

**1 Цель работы**

1.1 Изучить процесс реализации стандартных интерфейсов на языке C#

**2 Контрольные вопросы**

2.1 Для чего используется интерфейс IComparable?

Интерфейс IComparable позволяет объектам определять естественный порядок сортировки

2.2 Для чего используется интерфейс IEquatable?

Интерфейс IEquatable<T> предоставляет метод для проверки равенства объектов

* 1. В чем отличие между обобщенным и необобщенным интерфейсами?

Обобщенные интерфейсы: используют параметр типа T, что позволяет указать конкретный тип данных при реализации.

Необобщенные интерфейсы: работают с объектами типа object, что позволяет хранить данные разных типов.

**3 Код**

4 задания

В Customer.cs

public class Customer : IComparable, IComparable<Customer>, IEquatable<Customer>, ICloneable

{

public string FullName { get; set; }

public string Address { get; set; }

public decimal AmountSpent { get; set; }

public Customer(string fullName, string address, decimal amountSpent)

{

FullName = fullName;

Address = address;

AmountSpent = amountSpent;

}

public int CompareTo(object obj)

{

if (obj == null)

return 1;

Customer otherCustomer = obj as Customer;

if (otherCustomer != null)

return AmountSpent.CompareTo(otherCustomer.AmountSpent);

throw new ArgumentException("Объект не является заказчком.");

}

public int CompareTo(Customer other)

{

if (other == null)

return 1;

return AmountSpent.CompareTo(other.AmountSpent);

}

public bool Equals(Customer other)

{

if (other == null)

return false;

return FullName == other.FullName && Address == other.Address && AmountSpent == other.AmountSpent;

}

public object Clone()

{

return new Customer(FullName, Address, AmountSpent);

}

public override string ToString()

{

return $"{FullName}, {Address}, {AmountSpent}";

}

}

В Program.cs

Customer[] customers = new Customer[]

{

new("Акуловна Алеся Николаевна", "г. Пенза, ул. Мира, 22", 250),

new("Рыбкова Надежда Валерьевна", "г. Архангельск, ул. Троицкий, 13", 150),

new("Пушкина Валерия Юрьевна", "г. Москва, ул. Мира, 78", 300)

};

Console.WriteLine("Исходный массив:");

foreach (var customer in customers)

{

Console.WriteLine(customer);

}

Array.Sort(customers);

Console.WriteLine("\nОтсортированный массив:");

foreach (var customer in customers)

{

Console.WriteLine(customer);

}

Customer customer1 = new("Акуловна Алеся Николаевна", "г. Пенза, ул. Мира, 22", 200);

Customer customer2 = new("Акуловна Алеся Николаевна", "г. Пенза, ул. Мира, 22", 200);

Customer customer3 = new("Рыбкова Надежда Валерьевна", "г. Архангельск, ул. Троицкий, 13", 150);

Console.WriteLine($"\nСравнение customer1 и customer2: {customer1.Equals(customer2)}");

Console.WriteLine($"Сравнение customer1 и customer3: {customer1.Equals(customer3)}");

Customer clonedCustomer = (Customer)customer1.Clone();

Console.WriteLine($"\nКлонированный объект: {clonedCustomer}");

clonedCustomer.AmountSpent = 500;

Console.WriteLine($"Измененный клонированный объект: {clonedCustomer}");

Console.WriteLine($"Оригинальный объект после изменения клона: {customer1}");

**4 Вывод**

В ходе лабораторной работы я изучила процесс реализации стандартных интерфейсов на языке C#

**Лабораторная работа №11**

**Коллекции. Параметризованные классы**

**1 Цель работы**

1.1 Изучить процесс создания и применения параметризованных классов коллекций (обобщенных списков и словарей) на языке C#.

**2 Контрольные вопросы**

2.1 Что такое «коллекция»?

Коллекция — это структура данных, предназначенная для хранения набора элементов и управления ими.

2.2 Какие классы описаны в пространстве имен System.Collections.Generic?

Пространство имен System.Collections.Generic содержит обобщенные коллекции, которые работают с типом Т

2.3 Что такое List?

Это динамический массив, который автоматически расширяется при добавлении элементов.

2.4 Как можно обратиться к элементу списка?

К элементам списка обращаются по индексу

2.5 Что такое Dictionary<TKey, TValue>?

Dictionary<TKey, TValue> — это коллекция, которая хранит пары ключ-значение.

2.6 Как можно обратиться к элементу словаря?

К элементам словаря обращаются по ключу

**3 Вывод**

В ходе лабораторной работы я изучила процесс создания и применения параметризованных классов коллекций (обобщенных списков и словарей) на языке C#.

**Лабораторная работа №12**

**Работа с типом данных структура**

**1 Цель работы**

1.1 Изучить процесс создания и применения структур на языке C#.

**2 Контрольные вопросы**

2.1 В какой области памяти хранятся типы значения?

Типы значений хранятся в стеке, если они объявлены как локальные переменные в методах, или внутри в куче, если это поля ссылочного типа

2.2 Что такое «структура»?

Структура — это тип значения, который может содержать поля, методы, свойства и конструкторы.

2.3 Чем отличается структура от класса?

Структуры являются типами значений и хранятся в стеке или внутри родительского объекта. Они копируются целиком при передаче или присваивании, не поддерживают наследование и не могут быть null. Классы размещаются в куче, при операциях копируется только ссылка на объект, поддерживают полное наследование и могут принимать значение null.

2.4 Что такое «перечисление»?

Перечисление — это тип, который определяет набор именованных констант.

2.5 Для чего используются перечисления?

Перечисления применяются для: улучшения читаемости кода, ограничения допустимых значений

2.6 Какова общая форма объявления перечисления в C#?

[модификаторы] enum ИмяПеречисления [: базовыйТип]

{

ИмяКонстанты1 [= значение],

ИмяКонстанты2 [= значение],

}

**3 Код**

5 заданий

В Customer.cs

public struct Customer

{

private string \_fullName;

private string \_address;

private double \_spentAmount;

public Category CustomerCategory;

public Customer(string fullName, string address, double spentAmount, Category category)

{

FullName = fullName;

Address = address;

SpentAmount = spentAmount >= 0 ? spentAmount : 0;

CustomerCategory = category;

}

public string FullName

{

get => \_fullName;

set

{

if (!string.IsNullOrEmpty(value))

\_fullName = value;

else

throw new ArgumentException("ФИО не может быть пустым.");

}

}

public string Address

{

get => \_address;

set

{

if (!string.IsNullOrEmpty(value))

\_address = value;

else

throw new ArgumentException("Адрес не может быть пустым.");

}

}

public double SpentAmount

{

get => \_spentAmount;

set

{

if (value >= 0)

\_spentAmount = value;

else

throw new ArgumentException("Сумма не может быть отрицательной.");

}

}

public void DisplayInfo()

{

Console.WriteLine($"ФИО: {\_fullName}, Адрес: {\_address}, Потраченная сумма: {\_spentAmount}, Категория: {CustomerCategory}");

}

}

В Program.cs

// 1

Customer customer1 = new();

Customer customer2 = new("Лебедева Марьяна Ростиславовна", "Москва, ул. Вологодская, д. 67", 2000, Category.Private);

customer1.DisplayInfo();

customer2.DisplayInfo();

Console.WriteLine();

// 2

try

{

customer1.FullName = "Носов Иван Васильевич";

customer1.Address = "Астрахань, ул. Мира, д. 10";

customer1.SpentAmount = 1500;

customer1.CustomerCategory = Category.Corporative;

customer1.DisplayInfo();

customer1.SpentAmount = -500;

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

Console.WriteLine();

// 4

Customer[] customers = new[]

{

new("Сидоров Сергей Сергеевич", "Геленджик, ул. Морская, д. 36", 2009, Category.Private),

customer2,

new("Алексеева Мария Александровна", "Махачкала, ул. Суворова, д. 16", 500, Category.Corporative)

};

var foundCustomers = Array.FindAll(customers, c => c.CustomerCategory == Category.Private);

Console.WriteLine("Найденные клиенты категории 'Частный':");

foreach (var customer in foundCustomers)

{

customer.DisplayInfo();

}

Console.WriteLine();

// 5

FileRecord file1 = new() { FileName = "archive.zip", FullPath = "C:\\files\\archive.zip", FileSize = 1012 };

FileRecord file2 = new() { FileName = "audio.mp3", FullPath = "C:\\files\\audio.mp3", FileSize = 567 };

FileRecord file3 = new() { FileName = "video.mov", FullPath = "C:\\files\\video.mov", FileSize = 237 };

FileRecord file4 = new() { FileName = "audio.mp3", FullPath = "C:\\files\\audio.mp3", FileSize = 567 };

Console.WriteLine(file1.ToString());

Console.WriteLine(file2.ToString());

Console.WriteLine(file3.ToString());

Console.WriteLine(file4.ToString());

Console.WriteLine($"Сравнение file1 и file2: {file1.Equals(file2)}");

Console.WriteLine($"Сравнение file2 и file4: {file2.Equals(file4)}");

if (file1 == file2)

Console.WriteLine("file1 и file2 равны");

else

Console.WriteLine("file1 и file2 не равны");

if (file2 == file4)

Console.WriteLine("file2 и file4 равны");

else

Console.WriteLine("file2 и file4 не равны");

В Category.cs

public enum Category

{

Corporative,

Private

}

В FileRecord.cs

record FileRecord

{

public string FileName { get; set; }

public string FullPath { get; set; }

public int FileSize { get; set; }

public override string ToString()

=> $"Имя файла: {FileName}, Путь: {FullPath}, Размер: {FileSize}";

}

**4 Вывод**

В ходе лабораторной работы я изучила процесс создания и применения структур на языке C#

**Лабораторная работа №13**

**Обработка и форматирование строк**

**1 Цель работы**

1.1 Изучить процесс обработки строк на языке C#;

1.2 Научиться применять стандартные методы классов String, StringBuilder и Char для обработки строковых и символьных данных в программах на языке C#.

**2 Контрольные вопросы**

2.1 К какому типу переменных относятся переменные типа string?

String — это ссылочный тип

2.2 Какие операции допустимы над строковыми данными?

Основные операции со строками: объединение, сравнение, поиск и извлечение подстрок, форматирование

2.3 В чем отличие переменной типа string от массива символов?

Отличие заключается в том, что любые операции модификации строки создают новый объект в памяти, в то время как массив символов позволяет напрямую изменять свои элементы. Строка предоставляет множество методов для работы с текстом, тогда как с массивом работают как с обычной последовательностью данных.

2.4 Что такое «интерполяция строк»?

Интерполяция строк — это способ вставки значений переменных в строку с помощью $ и фигурных скобок {}.

2.5 Изменяют ли методы класса String исходную строку?

Все методы класса String возвращают новую строку не изменяя исходную.

**3 Код**

5 заданий

Task 1

Console.Write("Введите строку: ");

string input = Console.ReadLine();

Console.Write("Введите символ для поиска: ");

char searchChar = Console.ReadKey().KeyChar;

Console.WriteLine();

int totalChars = input.Length;

int charsWithoutSpaces = input.Count(c => !char.IsWhiteSpace(c));

int letterCount = input.Count(c => char.IsLetter(c));

var positions = Enumerable.Range(0, input.Length)

.Where(i => input[i] == searchChar)

.ToArray();

Console.WriteLine($"Количество символов: {totalChars}");

Console.WriteLine($"Количество символов без пробелов: {charsWithoutSpaces}");

Console.WriteLine($"Количество букв: {letterCount}");

if (positions.Length > 0)

Console.WriteLine("Позиции символа: " + string.Join(", ", positions));

else

Console.WriteLine("Совпадений не найдено.");

Console.WriteLine();

Task 2

Console.Write("Введите строку: ");

string inputTwo = Console.ReadLine();

string trimmedString = Regex.Replace(inputTwo.Trim(), @"\s+", " ");

Console.WriteLine("Измененная строка: " + trimmedString);

Console.Write("Выберите регистр (верхний, нижний, инвертированный): ");

string caseOption = Console.ReadLine();

StringBuilder modifiedString = new StringBuilder();

foreach (char c in trimmedString)

{

if (caseOption == "верхний")

modifiedString.Append(char.ToUpper(c));

else if (caseOption == "нижний")

modifiedString.Append(char.ToLower(c));

else if (caseOption == "инвертированный")

modifiedString.Append(char.IsUpper(c) ? char.ToLower(c) : char.ToUpper(c));

else

modifiedString.Append(c);

}

Console.WriteLine("Результат с измененным регистром: " + modifiedString.ToString());

Console.WriteLine();

Task3

Console.Write("Введите текст: ");

string inputThree = Console.ReadLine();

var words = inputThree.Split([' ', ',', '.', '!', '?'], StringSplitOptions.RemoveEmptyEntries);

Console.WriteLine("Массив слов: " + string.Join(", ", words));

Console.WriteLine();

Task 4

В Patient.cs

class Patient

{

public string FIO { get; set; }

public string Policy { get; set; }

public DateTime BirthDate { get; set; }

public override string ToString()

{

return $"{FIO.ToUpper()};{Policy.PadLeft(9, '0')};{BirthDate:yyyy/MM/dd}";

}

}

В Program.cs

Patient patient = new Patient

{

FIO = "Мирная Анастасия Антоновна",

Policy = "12345",

BirthDate = new DateTime(2014, 2, 3)

};

Console.WriteLine(patient.ToString());

Console.WriteLine();

Task 5

string[] array = ["Привет", "мир!", "прививка", "кошка"];

Console.Write("Введите текст для поиска: ");

string searchText = Console.ReadLine();

var results = Array.FindAll(array, s => s.Contains(searchText, StringComparison.OrdinalIgnoreCase));

if (results.Length > 0)

Console.WriteLine("Найденные строки: " + string.Join(", ", results));

else

Console.WriteLine("Совпадений не найдено.");

**4 Вывод**

В ходе лабораторной работы я изучила процесс обработки строк и научилась применять стандартные методы классов String, StringBuilder и Char для обработки строковых и символьных данных в программах на языке C#.

**Лабораторная работа №14**

**Использование регулярных выражений**

**1 Цель работы**

1.1 Научиться составлять шаблоны регулярных выражений в программах на C#;

1.2 Научиться применять регулярные выражения для поиска и замены подстрок в программах на языке C#.

**2 Контрольные вопросы**

2.1 Что такое «регулярное выражение»?

Регулярное выражение — это специальный шаблон для поиска, проверки или замены текста.

2.2 Для чего используются регулярные выражения?

Основные применения: проверка формата, замена частей текста по заданному правилу, разбиение строк на части

2.3 Для чего используется класс Regex?

Класс Regex в C# предоставляет методы для работы с регулярными выражениями

2.4 Каков алгоритм поиска подстроки при помощи регулярного выражения?

1 Компиляция шаблона.

2 Сканирование текста.

3 Возврат результата

2.5 Каков алгоритм замены подстроки при помощи регулярного выражения?

1 Поиск совпадений

1. Замена

2.6 Для чего в регулярных выражениях применяются escape-символы?

Escape-символы позволяют: экранировать спецсимволы, задавать спецпоследовательности

2.7 Для чего в регулярных выражениях применяются классы символов?

Классы символов группируют символы для упрощения шаблонов

2.8 Для чего в регулярных выражениях применяются квантификаторы?

Квантификаторы задают количество повторений

**3 Код**

5 заданий

Task 1

string phonePattern = @"^(**\+**7**\(**9\d{2}**\)**\d{3}-\d{2}-\d{2}|8**\(**9\d{2}**\)**\d{3}-\d{2}-\d{2})$";

string numberPhone = "+7(921)283-83-81";

string numberPhone2 = "8(921)283-83-81";

bool isPhoneValid = Regex.IsMatch(numberPhone, phonePattern);

Console.WriteLine($"Номер телефона {numberPhone} корректен: {isPhoneValid}");

bool isPhoneValid2 = Regex.IsMatch(numberPhone2, phonePattern);

Console.WriteLine($"Номер телефона {numberPhone2} корректен: {isPhoneValid2}");

Task 2

string whitespacePattern = @"\s{2,}";

string textWithSpaces = "балерина капучина лелел тралаллело тралала";

string cleanedText = Regex.Replace(textWithSpaces, whitespacePattern, " ");

Console.WriteLine($"Текст с пробелами: {cleanedText}");

Task 3

string emailPattern = @"^[a-zA-Z0-9\_-]+@[a-zA-Z0-9]+(**\.**[a-zA-Z]{2,})$";

string email = "kuplinov@gmail.com";

bool isEmailValid = Regex.IsMatch(email, emailPattern, RegexOptions.IgnoreCase);

Console.WriteLine($"E-mail {email} корректен: {isEmailValid}");

Task 4

string passwordPattern = @"^(?=.\*[0-9])(?=.\*[A-Z])(?=.\*[a-z])(?=.\*[?!.])[A-Za-z0-9.?!]{6,}$";

string password;

do

{

Console.WriteLine("Введите пароль: ");

password = Console.ReadLine();

if (Regex.IsMatch(password, passwordPattern))

Console.WriteLine("Пароль надежен");

else

Console.WriteLine("Пароль не соответствует требованиям");

}

while (!Regex.IsMatch(password, passwordPattern));

Task 5

string datePattern = @"(?<day>(0?[1-9]|[12]\d|3[01]))[./](?<month>(0?[1-9]|1[0-2]))[./](?<year>\d{2,4})";

string replacementPattern = @"${year}-${month}-${day}";

string dates = "25.10.2024 и 02/08/52";

string formattedDates = Regex.Replace(dates, datePattern, replacementPattern);

Console.WriteLine($"Отформатированные даты: {formattedDates}");

**4 Вывод**

В ходе лабораторной работы я научилась составлять шаблоны регулярных выражений в программах и применять регулярные выражения для поиска и замены подстрок в программах на языке C#.

**Лабораторная работа №15**

**Разработка делегатов**

**1 Цель работы**

1.1 Научиться разрабатывать и применять делегаты на C#

**2 Контрольные вопросы**

2.1 Что такое «делегат» в C#?

Делегат — это тип, который представляет ссылку на метод с определенной сигнатурой

2.2 Как объявить делегат на C#?

Делегат объявляется с ключевым словом delegate, сигнатурой метода и именем

2.3 Какие способы вызова метода через делегат существуют?

Прямой вызов, явный вызов и асинхронный вызов

2.4 Какие встроенные делегаты имеются в C# и для чего они используются?

Action — для методов без возвращаемого значения, Func — для методов с возвращаемым значением, Predicate<T> — возвращает bool, EventHandler — для событий

**3 Вывод**

В ходе лабораторной работы я научилась разрабатывать и применять делегаты на C#

**Лабораторная работа №16**

**Разработка событий**

**1 Цель работы**

1.1 Научиться создавать, вызывать и обрабатывать события на C#.

**2 Контрольные вопросы**

2.1 Что такое «событие» в C#?

Событие в C# — это механизм, позволяющий объекту уведомлять другие объекты о произошедшем действии

* 1. Как объявить событие на C#?

Событие объявляется с ключевым словом event и делегатом

2.3 Как создать обработчик события?

Создайте метод с сигнатурой, соответствующей делегату события и подпишите метод на событие через +=

2.4 Какой класс является родительским для всех классов данных события?

Базовый класс для данных события — EventArgs

2.5 Какие классы делегатов являются стандартными для создания событий в C#?

EventHandler, EventHandler<TEventArgs>

**3 Код**

Task1  
 В Program.cs  
  
static void Main(string[] args)

{

User user = new() { Login = "admin", Password = "asd" };

user.DataChanged += User\_DataChanged;

user.Login = "user";

user.Password = "dsa";

}

private static void User\_DataChanged(object? sender, EventArgs e)

{

User user = sender as User;

Console.WriteLine($"Изменены данные пользователя со следующим логином: {user.Login}");

}

В User.cs

private string \_login;

private string \_password;

public string Login

{

get => \_login;

set

{

if (\_login != value && !string.IsNullOrEmpty(value))

{

\_login = value;

OnDataChanged();

}

}

}

public string Password

{

get => \_password;

set

{

if (\_password != value)

{

\_password = value;

OnDataChanged();

}

}

}

public event EventHandler DataChanged;

public void OnDataChanged()

=> DataChanged?.Invoke(this, EventArgs.Empty);

Task3

В InfoEventArgs.cs

internal class InfoEventArgs : EventArgs

{

public string NameProperty { get; set; }

public string TextError { get; set; }

public DateTime DateChange { get; set; }

}

В Program.cs

static void Main(string[] args)

{

User user = new() { Login = "admin", Password = "asd" };

user.PropertyChanged += User\_PropertyChanged;

user.Login = "";

user.Password = "dsaaa";

}

private static void User\_PropertyChanged(object? sender, InfoEventArgs e)

{

Console.WriteLine($"{e.DateChange}: у пользователя {((User)sender).Login} изменено {e.NameProperty}.\n{e.TextError}");

}

В User.cs

internal class User : InfoEventArgs

{

private string \_login;

private string \_password;

public event EventHandler<InfoEventArgs> PropertyChanged;

public string Login

{

get => \_login;

set

{

if (\_login != value && !string.IsNullOrEmpty(value))

{

\_login = value;

}

OnPropertyLoginChanged(\_login);

}

}

public string Password

{

get => \_password;

set

{

if (\_password != value && value.Length > 5 && value.Length < 20)

{

\_password = value;

}

OnPropertyPasswordChanged(\_password);

}

}

public void OnPropertyLoginChanged(string value)

{

InfoEventArgs info = new();

info.NameProperty = "логин";

info.DateChange = DateTime.Now;

if (string.IsNullOrEmpty(value))

{

info.TextError = "Ошибка: логин не может быть пустым";

}

else

{

info.TextError = "";

}

PropertyChanged?.Invoke(this, info);

}

public void OnPropertyPasswordChanged(string value)

{

InfoEventArgs info = new()

{

NameProperty = "пароль",

DateChange = DateTime.Now

};

if (string.IsNullOrEmpty(value) || value.Length < 6 || value.Length > 20)

{

info.TextError = "Ошибка: пароль не может быть меньше 6 или больше 20";

}

else

{

info.TextError = "";

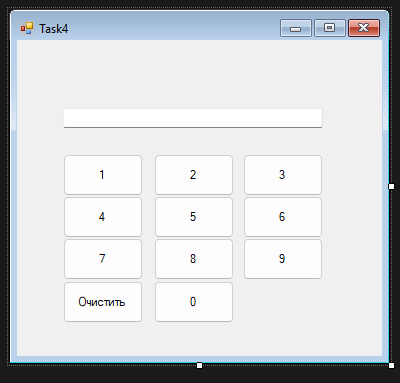
}

PropertyChanged?.Invoke(this, info);

}

}

Task4



public partial class NumbersForm : Form

{

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

{

Button button = (Button)sender;

numbersTextBox.Text += button.Text;

}

public NumbersForm()

{

InitializeComponent();

}

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

{

numbersTextBox.Clear();

}

}

**4 Вывод**

В ходе лабораторной работы научились создавать, вызывать и обрабатывать события на C#

**Лабораторная работа №17**

**Операции со списками**

**1 Цель работы**

1.1 Изучить процесс обработки списков стандартными методами на языке C#.

**2 Контрольные вопросы**

2.1 Что такое «LINQ»?

LINQ — это технология в C#, которая позволяет выполнять запросы к данным прямо в коде, используя единый синтаксис.

2.2 Что такое «анонимный тип»?

Анонимный тип — это тип данных, создаваемый компилятором автоматически без явного объявления.

* 1. Для чего используется var?

Var позволяет сократить код при работе с сложными типами

2.4 Для чего используется метод Select?

Метод Select преобразует элементы коллекции по заданному правилу.

2.5 Для чего используются методы OrderBy, OrderByDescending?

Эти методы сортируют коллекцию: OrderBy — по возрастанию, OrderByDescending — по убыванию

2.6 Для чего используются методы ThenBy, ThenByDescending?

Эти методы применяются после OrderBy для дополнительной сортировки по другим критериям

**3 Код**

var task31 = files

.Select(file => file.Extension)

.Distinct()

.Dump();

var task32 = files

.GroupBy(file => file.Extension)

.Select(group => new { Extension = group.Key, FilesCount = group.Count() })

.Dump();

var task4 = files

.Where(file => file.CreationTime.Date == DateTime.Today)

.OrderByDescending(file => file.CreationTime)

.Take(3)

.Dump();

**4 Вывод**

В ходе лабораторной работы я изучила процесс обработки списков стандартными методами на языке C#

**Лабораторная работа №18**

**Проектирование диаграммы классов**

**1 Цель работы**

1.1 Изучить процесс описания типов данных с применением диаграммы классов.

1. **Контрольные вопросы**

2.1 Для чего используется диаграмма классов?

Она используется для: визуализации классов, их свойств, методов и взаимосвязей, проектирования архитектуры приложения до написания кода, анализа взаимодействия между компонентами системы.

2.2 Из каких элементов состоит описание типа в диаграмме классов?

Из имени, атрибутов и методов

2.3 На какие группы делятся взаимосвязи в диаграмме классов?

На структурные, зависимости и анонимные связи

2.4 В чем отличие между взаимосвязью «наследование» и «реализация»?

Наследование обозначает отношение между классами, где дочерний класс наследует структуру и поведение родительского класса. Реализация же показывает, что класс реализует определенный интерфейс, то есть обязуется предоставить конкретную реализацию всех его методов.

2.5 В чем отличие между взаимосвязью «агрегация» и «композиция»?

Агрегация представляет более слабую связь, где часть может существовать независимо от целого. Композиция же показывает более сильную связь, где часть не может существовать без целого и уничтожается вместе с ним.

2.6 Что показывает взаимосвязь «ассоциация»?

Ассоциация — это простейшая связь между классами, указывающая на их взаимодействие.

2.7 Какие виды мощности отношений могут быть указаны в диаграмме классов?

Основные виды мощности включают: "1", "0..1", "" или "0..", "1..\*", конкретные диапазоны

**3 Вывод**

В ходе лабораторной работы я изучила процесс описания типов данных с применением диаграммы классов.

**Лабораторная работа №19**

**Использование шаблонов проектирования**

**1 Цель работы**

1.1 Научиться применять паттерны проектирования в разработке программ.

**2 Контрольные вопросы**

2.1 Для чего используются порождающие паттерны?

Порождающие паттерны решают задачи создания объектов, делая процесс более гибким и контролируемым

2.2 Какие паттерны относятся к порождающим?

Singleton, Factory Method, Abstract Factory, Builder

2.3 Для чего используются структурные паттерны?

Структурные паттерны отвечают за построение гибких и эффективных структур классов и объектов

2.4 Какие паттерны относятся к структурным?

Adapter, Decorator, Facade, Composite

2.5 Для чего используются поведенческие паттерны?

Поведенческие паттерны решают задачи взаимодействия между объектами и распределения ответственности

2.6 Какие паттерны относятся к поведенческим?

Observer, Strategy, Command, Iterator

**3 Код**

4 задания

Task 1

В Context.cs

private IStrategy \_strategy;

public Context(IStrategy strategy)

{

\_strategy = strategy;

}

public int ExecuteStrategy(int number1, int number2)

{

return \_strategy.DoOperation(number1, number2);

}

В IStrategy.cs

public interface IStrategy

{

public int DoOperation(int number1, int number2);

}

В OpertionAdd.cs

public class OperationAdd : IStrategy

{

public int DoOperation(int number1, int number2)

{

return number1 + number2;

}

}

В OperationDivision.cs

public class OperationDivision : IStrategy

{

public int DoOperation(int number1, int number2)

{

return number1 / number2;

}

}

В OpertionMultiply.cs

public class OperationMultiply : IStrategy

{

public int DoOperation(int number1, int number2)

{

return number1 \* number2;

}

}

В OperationSubstract.cs

public class OperationSubstract : IStrategy

{

public int DoOperation(int number1, int number2)

{

return number1 - number2;

}

}

В Program.cs

Context context = new(new OperationAdd());

Console.WriteLine("10 + 5 = " + context.ExecuteStrategy(10, 5));

context = new(new OperationSubstract());

Console.WriteLine("10 - 5 = " + context.ExecuteStrategy(10, 5));

context = new(new OperationMultiply());

Console.WriteLine("10 \* 5 = " + context.ExecuteStrategy(10, 5));

context = new(new OperationDivision());

Console.WriteLine("10 / 5 = " + context.ExecuteStrategy(10, 5));

Task 2

public interface IShape

{

void Draw();

}

public class Circle : IShape

{

public void Draw() => Console.WriteLine("Внутри Circle::draw() метод.");

}

public class Rectangle : IShape

{

public void Draw() => Console.WriteLine("Внутри Rectangle::draw() метод.");

}

public class Square : IShape

{

public void Draw() => Console.WriteLine("Внутри Square::draw() метод.");

}

public class Star : IShape

{

public void Draw() => Console.WriteLine("Внутри Star::draw() метод.");

}

public class ShapeFactory

{

public IShape? GetShape(string shapeType)

{

if (string.IsNullOrEmpty(shapeType))

return null;

if (shapeType.Equals("КРУГ", StringComparison.OrdinalIgnoreCase))

return new Circle();

if (shapeType.Equals("ПРЯМОУГОЛЬНИК", StringComparison.OrdinalIgnoreCase))

return new Rectangle();

if (shapeType.Equals("КВАДРАТ", StringComparison.OrdinalIgnoreCase))

return new Square();

if (shapeType.Equals("ЗВЕЗДА", StringComparison.OrdinalIgnoreCase))

return new Star();

return null;

}

}

ShapeFactory shapeFactory = new();

IShape shape1 = shapeFactory.GetShape("КРУГ");

shape1.Draw();

IShape shape2 = shapeFactory.GetShape("ПРЯМОУГОЛЬНИК");

shape2.Draw();

IShape shape3 = shapeFactory.GetShape("КВАДРАТ");

shape3.Draw();

IShape shape4 = shapeFactory.GetShape("ЗВЕЗДА");

shape4.Draw();

Task 3

public interface IShape

{

void Draw();

}

public class Circle : IShape

{

public void Draw()

{

Console.WriteLine("Фигура: Круг");

}

}

public class Rectangle : IShape

{

public void Draw()

{

Console.WriteLine("Фигура: Прямоугольник");

}

}

public abstract class ShapeDecorator : IShape

{

protected IShape decoratedShape;

public ShapeDecorator(IShape decoratedShape)

{

this.decoratedShape = decoratedShape;

}

public void Draw()

{

decoratedShape.Draw();

}

}

public class RedShapeDecorator : ShapeDecorator

{

public RedShapeDecorator(IShape decoratedShape) : base(decoratedShape) { }

public void Draw()

{

decoratedShape.Draw();

RedBorder(decoratedShape);

}

private void RedBorder(IShape DecoratedShape)

{

Console.WriteLine("Цвет окантовки: красный");

}

}

public class PurpleShapeDecorator : ShapeDecorator

{

public PurpleShapeDecorator(IShape decoratedShape) : base(decoratedShape) { }

public void Draw()

{

decoratedShape.Draw();

PurpleBorder(decoratedShape);

}

private void PurpleBorder(IShape DecoratedShape)

{

Console.WriteLine("Цвет окантовки: фиолетовый");

}

}

IShape circle = new Circle();

RedShapeDecorator RedCircle = new(new Circle());

RedShapeDecorator RedRectangle = new(new Rectangle());

PurpleShapeDecorator PurpleCircle = new(new Circle());

Console.WriteLine("Круг с обычной окантовкой");

circle.Draw();

Console.WriteLine("\nКруг с красной окантовкой");

RedCircle.Draw();

Console.WriteLine("\nПрямоугольник с красной окантовкой ");

RedRectangle.Draw();

Console.WriteLine("\nКруг с фиолетовой окантовкой");

PurpleCircle.Draw();

Task 4

public abstract class Observer

{

protected Subject \_subject;

public abstract void update();

}

public class BinaryObserver : Observer

{

public BinaryObserver(Subject subject)

{

\_subject = subject;

\_subject.attach(this);

}

public override void update()

{

Console.WriteLine("Binary String: " + Convert.ToString(\_subject.getState(),2));

}

}

public class HexaObserver : Observer

{

public HexaObserver(Subject subject)

{

\_subject = subject;

\_subject.attach(this);

}

public override void update()

{

Console.WriteLine("Hex String: " + Convert.ToString(\_subject.getState()).ToUpper());

}

}

public class OctalObserver : Observer

{

public OctalObserver(Subject subject)

{

\_subject = subject;

\_subject.attach(this);

}

public override void update()

{

Console.WriteLine("Octal String: " + \_subject.getState().ToString("X"));

}

}

public class Subject

{

private List<Observer> \_observers = new();

private int \_state;

public int getState()

{

return \_state;

}

public void setState(int state)

{

\_state = state;

notifyAllObservers();

}

public void attach(Observer observer)

{

\_observers.Add(observer);

}

public void notifyAllObservers()

{

foreach (var observer in \_observers)

{

observer.update();

}

}

}

Subject subject = new Subject();

new HexaObserver(subject);

new OctalObserver(subject);

new BinaryObserver(subject);

Console.WriteLine("First state change: 15");

subject.setState(15);

Console.WriteLine("Second state change: 10");

subject.setState(10);

**4 Вывод**

В ходе лабораторной работы я научилась применять паттерны проектирования в разработке программ.

**Лабораторная работа №20**

**Проектирование API**

**1 Цель работы**

1.1 Научиться проектировать API.

**2 Контрольные вопросы**

2.1 Что такое REST?

REST — это архитектурный стиль для создания веб-API, основанный на стандартных HTTP-методах и принципах

2.2 Для чего используется метод GET?

GET — запрос данных с сервера без изменения состояния ресурса.

2.3 Для чего используется метод POST?

POST — отправка данных на сервер для создания нового ресурса.

2.4 Для чего используется метод PUT?

PUT — полное обновление существующего ресурса (или создание, если его нет).

2.5 Для чего используется метод DELETE?

DELETE — удаление ресурса по указанному URI.

**3 Вывод**

В ходе лабораторной работы я научилась проектировать API.

**Лабораторная работа №21**

**Оптимизация кода**

**1 Цель работы**

1.1 Изучить методы оптимизации программного кода

**2 Контрольные вопросы**

2.1 Что такое «оптимизация программного кода»?

Оптимизация кода — это процесс улучшения программы для повышения её эффективности по одному или нескольким критериям

2.2 Какова цель оптимизации программного кода?

Ускорение работы программы, снижение нагрузки на систему, поддержка масштабируемости

2.3 Какие методы оптимизации программного кода применяются?

Замена линейного поиска на бинарный, использование хеш-таблиц вместо списков для частых поисков, применение кэширования для повторных вычислений

**3 Вывод**

В ходе лабораторной работы я изучила методы оптимизации программного кода

**Лабораторная работа №22**

**Асинхронное программирование**

**1 Цель работы**

1.1 Научиться реализовывать и запускать асинхронные операции на C#.

1.2 Научиться выполнять вычисления, используя асинхронные операции.

1.3 Научиться выполнять ввод и вывод данных, используя асинхронные операции.

**2 Контрольные вопросы**

2.1 Какие ключевые слова используются в C# для работы с асинхронными вызовами?

async, await, Task / Task<T>

2.2 Какие типы возвратамогут быть у асинхронныхметодов и для чего предназначен каждый из типов?

Task – для методов без возвращаемого значения, Task<T> - Для методов, возвращающих результат типа T, void - Только для обработчиков событий

2.3 Как вызвать метод в асинхронномрежиме?

Используйте await для ожидания завершения асинхронной операции

2.4 Как указать, что в методемогут быть асинхронные вызовы?

Пометьте метод ключевым словом async и используйте await для вызова других асинхронных операций

2.5 Как обработатьисключения, возникшие в асинхронных вызовах?

Исключения в асинхронных методах перехватываются через try-catch

**3 Код**

4 задания

Task 1

static async Task PowerAsync(double a, int x)

{

double result = Math.Pow(a, x);

Console.WriteLine($"{a} ^ {x} = {result}");

}

private static async Task Main(string[] args)

{

Console.WriteLine("Последовательный вызов: ");

await Task.Run(() => PowerAsync(2, 3));

await Task.Run(() => PowerAsync(4, 4));

await Task.Run(() => PowerAsync(5, 2));

Console.WriteLine("Параллельный вызов: ");

Task task1 = Task.Run(() => PowerAsync(2, 3));

Task task2 = Task.Run(() => PowerAsync(4, 4));

Task task3 = Task.Run(() => PowerAsync(5, 2));

await Task.WhenAll(task1, task2, task3);

}

Task 2

static async Task GetFileAsync(string fileName, int N)

{

Console.WriteLine($"Запись в файл {fileName} начата");

using (StreamWriter writer = new(fileName, false))

{

Random random = new Random();

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

{

await writer.WriteLineAsync($"Число {i}: {random.Next()}");

}

}

Console.WriteLine($"Запись в файл {fileName} закончена");

}

static async Task Main(string[] args)

{

await GetFileAsync("number.txt", 100000);

Console.WriteLine("конец");

}

Task 3

string[] fileNames = {

"C:\\Temp\\ispp31\\1.txt",

"C:\\Temp\\ispp31\\2.txt",

"C:\\Temp\\ispp31\\3.txt"

};

Task[] readTasks = new Task[fileNames.Length];

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

{

readTasks[i] = ReadFileAsync(fileNames[i]);

}

await Task.WhenAll(readTasks);

static async Task ReadFileAsync(string fileName)

{

Console.WriteLine($"чтение из файла {fileName} начато");

using (StreamReader reader = new(fileName))

{

string line;

while ((line = await reader.ReadLineAsync()) != null)

{

Console.WriteLine($"{fileName}: {line}");

}

}

Console.WriteLine($"чтение из файла {fileName} закончено");

}

Task4

string directoryPath = @"Y:\МДК.01.01";

var files = Directory.GetFiles(directoryPath);

foreach (var file in files)

{

var hashTask = CalculateFileHashAsync(file);

hashTask.Wait();

Console.WriteLine(hashTask.Result);

}

static string CalculateHash(byte[] data)

{

using var sha256 = SHA256.Create();

return Convert.ToHexString(sha256.ComputeHash(data));

}

static async Task<string> CalculateFileHashAsync(string filePath)

{

byte[] fileData = await File.ReadAllBytesAsync(filePath);

string hash = await Task.Run(() => CalculateHash(fileData));

return $"{Path.GetFileName(filePath)} : {hash}";

}

**4 Вывод**

В ходе лабораторной работы я научилась реализовывать и запускать асинхронные операции, выполнять вычисления, ввод и вывод данных, используя асинхронные операции.

**Лабораторная работа №23**

**Документирование кода**

**1 Цель работы**

1.1 Изучить процесс документирования программного кода

**2 Контрольные вопросы**

2.1 Что такое «XML-документация»?

XML-документация — это стандартизированный способ документирования кода в C# с помощью специальных комментариев в формате XML

2.2 Как сгенерировать XML-комментарий?

Установите курсор над методом или классом, введите /// и заполните нужные тэги

2.3 Какие действия нужно выполнить, чтобы XML-документация была видна при подключении библиотеки в стороннем решении?

Включите генерацию XML-файла и Добавьте XML-файл в NuGet-пакет

2.4 Что пишется в разделе summary?

**<**summary> содержит краткое описание функциональности

2.5 Что пишется в разделе param?

<param> описывает параметры метода

2.5 Что пишется в разделе returns?

**<**returns**>** используется для описания возвращаемого значения метода

**3 Вывод**

В ходе лабораторной работы я изучила процесс документирования программного кода.

**Лабораторная работа №24**

**Рефакторинг кода**

**1 Цель работы**

1.1 Изучить и применить техники рефакторинга программного кода.

**2 Контрольные вопросы**

2.1 Что такое «рефакторинг»?

Рефакторинг — это процесс улучшения структуры кода без изменения его внешнего поведения.

2.2 Какие группы техник рефакторинга существуют?

Рефакторинг методов, классов, работа с данными, упрощение вызовов

2.3 Как выполнить рефакторинг в Visual Studio?

Переименование – F2 или ПКМ -> Rename, извлечение метода – Выделите код -> ПКМ -> Quick Actions -> Extract Method, извлечение интерфейса – ПКМ на классе -> Quick Actions -> Extract Interface

**3 Код**

5 заданий

В Order.cs

public class Order : OrderService

{

public int Id { get; set; }

private double total;

public double Total

{

get => total;

set

{

if (value < 0)

throw new ArgumentException("Стоимость заказа не может быть отрицательной.");

total = value;

}

}

public bool IsExpress { get; set; }

private string address;

public string Address

{

get => address;

set

{

if (string.IsNullOrWhiteSpace(value))

throw new ArgumentException("Адрес не может быть пустым.");

address = value;

}

}

private const double TaxRate = 0.2;

private const double DiscountThreshold = 10000;

private const double DiscountRate = 0.1;

private ShippingService shippingService = new();

public double CalculateTotalPrice(Order order)

{

if (Address is null)

throw new OrderException("Адрес не указан.");

double discount = CalculateDiscountCost();

double deliveryCost = shippingService.CalculateDeliveryCost(Total, IsExpress);

double tax = CalculateTax();

return Total - discount + tax + deliveryCost;

}

private double CalculateTax()

{

return Total \* TaxRate;

}

private double CalculateDiscountCost()

{

return Total > DiscountThreshold ? Total \* DiscountRate : 0;

}

}

В OrderException.cs

public class OrderException : Exception

{

public OrderException(string message) : base(message) { }

}

В OrderService.cs

public class OrderService

{

private List<Order> orders { get; set; } = new List<Order>();

public void AddOrder(Order order)

{

orders.Add(order);

}

public void PrintOrderDetails(int orderId)

{

var order = orders.FirstOrDefault(order => order.Id == orderId);

if (order == null)

{

Console.WriteLine("Заказ не найден.");

return;

}

Console.WriteLine("Order Id: " + order.Id);

Console.WriteLine("Стоимость заказа: " + order.Total);

Console.WriteLine("Адрес: " + order.Address);

Console.WriteLine("Экспресс доставка: " + (order.IsExpress ? "Да" : "Нет"));

}

}

В Program.cs

Order orderService = new();

var order = new Order {Id = 2228, Total = 2700, IsExpress = false, Address = "Папанина, 24" };

orderService.AddOrder(order);

orderService.PrintOrderDetails(order.Id);

Console.WriteLine("Полная стоимость: " + orderService.CalculateTotalPrice(order));

В ShippingService.cs

public class ShippingService

{

private const double FreeShippingThreshold = 3000;

private const double FixedShippingThreshold = 1000;

private const double FixedOrderShippingCost = 230;

private const double ExpressShippingCostCoefficient = 2;

public double CalculateDeliveryCost(double total, bool isExpress)

{

double deliveryCost = CalculateBaseDeliveryCost(total);

return CalculateExpressDelivery(isExpress, deliveryCost);

}

private double CalculateBaseDeliveryCost(double total)

{

if (total < FixedShippingThreshold)

return total;

if (total < FreeShippingThreshold)

return FixedOrderShippingCost;

return 0;

}

private static double CalculateExpressDelivery(bool isExpress, double deliveryCost)

{

if (isExpress)

deliveryCost \*= ExpressShippingCostCoefficient;

return deliveryCost;

}

}

**4 Вывод**

В ходе лабораторной работы я изучила и применила техники рефакторинга программного кода

**Лабораторная работа №25**

**Работа с системой контроля версий**

**1 Цель работы**

1.1 Научиться применять систему контроля версий git в процессе разработки программного обеспечения

**2 Контрольные вопросы**

2.1 Что такое «репозиторий»?

Репозиторий — это хранилище кода и связанных с ним файлов

2.2 Что указывается в файле readme.md?

README.md — документация проекта.

2.3 Что указывается в файле .gitignore?

.gitignore — исключает ненужные файлы.

2.4 Какое программное обеспечение может применяться для управления git репозиторием?

Командная строка, GUI-клиенты, интеграция в IDE

2.5 Где может располагаться репозиторий?

Локально — на вашем компьютере или удаленно — на хостингах

2.6 Какой основной порядок выполнения команд при работе с репозиторием?

Клонирование, создание ветки, добавление изменений, фиксация изменений, отправка на удаленный сервер, обновление локальной копии

**3 Вывод**

В ходе лабораторной работы я научилась применять систему контроля версий git в процессе разработки программного обеспечения

**Лабораторная работа №26**

**Разработка интерфейса пользователя: компоновка элементов**

**1 Цель работы**

1.1 Изучить элементы-контейнеры, применяющиеся в приложениях WPF для компоновки.

**2 Контрольные вопросы**

2.1 Что такое «компоновка» в WPF?

Компоновка в WPF — это процесс автоматического размещения и упорядочивания элементов управления внутри контейнеров с учетом их размеров, выравнивания и внешних отступов.

2.2 Какой класс является родительским для всех элементов-контейнеров в WPF?

Базовый класс для всех контейнеров — Panel

2.3 Как выровнять элементы внутри контейнера по высоте и по ширине?

HorizontalAlignment или VerticalAlignment

2.4 В чем особенность компоновки с использованием следующих элементовконтейнеров: Grid и Canvas?

Grid: сетка со строками и столбцами, элементы размещаются с помощью Grid.Row и Grid.Column

Canvas: позиционирование через Canvas.Left и Canvas.Top, используется для точного размещения

2.5 Чем отличается компоновка с использованием StackPanel, DockPanel, WrapPanel?

StackPanel размещает дочерние элементы последовательно в одну линию — вертикально или горизонтально. DockPanel позволяет "прикреплять" элементы к определенным краям с помощью свойства DockPanel.Dock. WrapPanel автоматически переносит элементы на новую строку или столбец, когда они перестают помещаться в доступное пространство

2.6 В каких единицах измерения могут задаваться размеры элементов в приложениях WPF?

px, Auto, \*, относительные единицы

**3 Вывод**

В ходе лабораторной работы я изучила элементы-контейнеры, применяющиеся в приложениях WPF для компоновки.

**Лабораторная работа №27**

**Организация интерфейса пользователя**

**1 Цель работы**

1.1 Изучить процесс настройки интерфейса и организации переходов в приложениях WPF.

**2 Контрольные вопросы**

2.1 Для чего используется элемент управления Frame?

Frame в WPF — это контейнер, который позволяет отображать страницы и поддерживает навигацию между ними

2.2 Для чего используется элемент управления Page?

Page — это самостоятельная XAML-страница, которая используется вместе с Frame для навигации

2.3 Как перейти к определенной странице, используя фрейм?

Установить свойство Source или создать программный переход через метод Navigate()

2.4 Как проверить, что во фрейме можно вернуться к предыдущей странице?

Использовать свойство CanGoBack

2.5 Как перейти к предыдущей странице, используя фрейм?

Вызвать метод GoBack()

2.6 Какие элементы позволяют сгруппировать содержимое?

GroupBox, TabControl, Expander

**3 Код**

4 задания

В MainWindows.xaml

Title="{Binding ElementName=ContentFrame, Path=Content.Title}" Height="450" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Name="Cap" Height="100"/>

<RowDefinition/>

<RowDefinition Name="Basement" Height="50"/>

</Grid.RowDefinitions>

<Frame Name="ContentFrame" Grid.Row="1" ContentRendered="ContentFrame\_ContentRendered" NavigationUIVisibility="Hidden"/>

<Border Grid.Row="2" Background="DarkOliveGreen"/>

<Border Background="LightYellow"/>

<Image Source="C:\Users\Lenovo\Downloads\dd.jpg" HorizontalAlignment="Left"/>

<Button Name="GoBackButton" Content="Назад" Click="GoBackButton\_Click" HorizontalAlignment="Right" VerticalAlignment="Center"/>

</Grid>

В MainWindows.xaml.cs

public static Frame MainFrame { get; set; }

public MainWindow()

{

InitializeComponent();

ContentFrame.Navigate(new MainPage());

App.MainFrame = ContentFrame;

}

private void ContentFrame\_ContentRendered(object sender, EventArgs e)

{

if (App.MainFrame.CanGoBack)

GoBackButton.Visibility = Visibility.Visible;

else

GoBackButton.Visibility = Visibility.Collapsed;

}

private void GoBackButton\_Click(object sender, RoutedEventArgs e)

{

App.MainFrame.GoBack();

}

Папка Pages:

В AuthorizationPage.xaml

<StackPanel Background="White">

<Label Content="Логин"/>

<TextBox Name="LoginTextBox"/>

<Label Content="Пароль"/>

<TextBox Name="PasswordTextBox"/>

<Button Name="EnterButton" Content="Войти" Click="EnterButton\_Click"/>

<Button Name="RegistrationButton" Content="Зарегистрироваться" Click="RegistrationButton\_Click"/>

</StackPanel>

В AuthorizationPage.xaml.cs

private void RegistrationButton\_Click(object sender, RoutedEventArgs e)

{

App.MainFrame.Navigate(new RegistrationPage());

}

private void EnterButton\_Click(object sender, RoutedEventArgs e)

{

if (App.MainFrame.CanGoBack)

App.MainFrame.GoBack();

}

В MainPage.xaml

<StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">

<Button Name="Authorization" Content="Авторизация" Click="Authorization\_Click"/>

<Button Name="Registration" Content="Регистрация" Click="Registration\_Click"/>

<Button Name="Settings" Content="Настройки" Click="Settings\_Click"/>

</StackPanel>

В MainPage.xaml.cs

private void Authorization\_Click(object sender, RoutedEventArgs e)

{

App.MainFrame.Navigate(new AuthorizationPage());

}

private void Settings\_Click(object sender, RoutedEventArgs e)

{

App.MainFrame.Navigate(new SettingsPage());

}

private void Registration\_Click(object sender, RoutedEventArgs e)

{

App.MainFrame.Navigate(new RegistrationPage());

}

В RegistrationPage.xaml

<StackPanel Background="White">

<Label Content="Логин"/>

<TextBox Name="LoginTextBox"/>

<Label Content="Пароль"/>

<TextBox Name="PasswordTextBox"/>

<Label Content="Подтверждение пароля"/>

<TextBox Name="ConfirmPasswordTextBox"/>

<Button Name="RegistrationButton" Content="OK" Click="RegistrationButton\_Click"/>

</StackPanel>

В RegistrationPage.xaml.cs

private void RegistrationButton\_Click(object sender, RoutedEventArgs e)

{

if (App.MainFrame.CanGoBack)

App.MainFrame.GoBack();

}

В SettingsPage.xaml

<TabControl>

<TabItem Header="Категории">

<StackPanel>

<GroupBox>

<Label Content="О программе"/>

</GroupBox>

<GroupBox>

<StackPanel>

<Label Content="Цветовая тема"/>

<RadioButton Content="Тёмная"/>

<RadioButton Content="Светлая"/>

</StackPanel>

</GroupBox>

<GroupBox>

<StackPanel>

<Label Content="Размер текста"/>

<RadioButton Content="Большой"/>

<RadioButton Content="Средний"/>

<RadioButton Content="Мелкий"/>

</StackPanel>

</GroupBox>

</StackPanel>

</TabItem>

<TabItem Header="Категории">

<StackPanel>

<Expander Header="О программе">

</Expander>

<Expander Header="Цветовая Тема">

<StackPanel>

<RadioButton Content="Тёмная"/>

<RadioButton Content="Светлая"/>

</StackPanel>

</Expander>

<Expander Header="Размер текста">

<StackPanel>

<RadioButton Content="Большой"/>

<RadioButton Content="Средний"/>

<RadioButton Content="Мелкий"/>

</StackPanel>

</Expander>

</StackPanel>

</TabItem>

</TabControl>

В SettingsPage.xaml.cs

public SettingsPage()

{

InitializeComponent();

CheckBox checkBox;

Expander expander;

StackPanel stackPanel;

}

**4 Вывод**

В ходе лабораторной работы я изучила процесс настройки интерфейса и организации переходов в приложениях WPF.

**Лабораторная работа №28**

**Разработка интерфейса пользователя: настройка стилей**

**1 Цель работы**

1.1 Изучить процесс настройки интерфейса с использованием стилей в

приложениях WPF.

**2 Контрольные вопросы**

2.1 Для чего используются стили в приложениях WPF?

Стили упрощают единообразное оформление элементов

2.2 Какова общая форма локального определения стиля элемента управления?

<Style TargetType=" Тип\_элемента">

<Setter Property=" Свойство" Value="Значение"/>

</Style>

2.3 Какова общая форма определения стиля приложения?

Стиль приложения определяется в App.xaml

<Application.Resources>

<Style TargetType="Тип\_элемента" x:Key="Ключ\_стиля">

<Setter Property="Свойство" Value="Значение"/>

</Style>

</Application.Resources>

2.4 Как указать явное использование стилей?

<Control Style="{StaticResource Ключ\_стиля}"/>

2.5 Как указать наследование стиля?

Использовать свойство BasedOn="{StaticResource BaseStyle}"

2.6 Как добавить новую тему в приложение?

Создать файл ресурсов и добавить тему в App.xaml, подключить Source

2.7 Как выполнить переключение между темами?

Создать несколько файлов ресурсов, далее в xaml

<Button Content="Темная тема" Click="SwitchToDarkTheme"/>

<Button Content="Светлая тема" Click="SwitchToLightTheme"/>

Далее

private void SwitchToDarkTheme(object sender, RoutedEventArgs e)

{

ApplyTheme("Themes/DarkTheme.xaml");

}

**3 Вывод**

В ходе лабораторной работы я изучила процесс настройки интерфейса с использованием стилей в приложениях WPF.

**Лабораторная работа №29**

**Разработка интерфейса пользователя: применение триггеров**

**1 Цель работы**

1.1 Изучить процесс применения триггеров в приложениях WPF,

1.2 Закрепить навык применения стилей в приложениях на WPF.

**2 Контрольные вопросы**

2.1 Что позволяют делать триггеры в приложениях WPF?

Триггеры автоматически изменяют свойства элементов при определённых условиях

2.2 Какие виды триггеров можно разработать в приложениях WPF?

Виды триггеров: триггер свойств, триггер данных, триггер событий

2.3 Для чего используется и когда срабатывает триггер свойств?

Триггер свойств срабатывает при изменении свойства, например IsMouseOver

2.4. Для чего используется и когда срабатывает триггер данных?

Триггер данных реагирует на изменение данных, например текста

2.5. Для чего используется и когда срабатывает триггер событий?

Триггер событий срабатывает при наступлении события, например нажатие

**3 Вывод**

В ходе лабораторной работы я изучила процесс применения триггеров и закрепила навык применения стилей в приложениях на WPF

**Лабораторная работа №30**

**Изучение особенностей элементов выбора в приложениях WPF**

**1 Цель работы**

1.1 Изучить свойства и процесс обработки событий элементов выбора в

приложениях WPF.

**2 Контрольные вопросы**

2.1. Что такое CheckBox и для чего он используется?

CheckBox — флажок для выбора одного или нескольких вариантов

2.2. Что такое RadioButton и для чего он используется?

RadioButton — переключатель для выбора одного варианта из группы

вариантов

2.3. Как проверить, что флажок или переключатель выбран?

Состояние элемента проверяется через свойство IsChecked, должно быть

true

2.4. Какое событие срабатывает при выборе флажка или переключателя?

Событие Checked срабатывает при выборе флажка или переключателя

2.5. Какое событие срабатывает при снятии выбора флажка или переключателя?

Событие Unchecked срабатывает при снятии выбора флажка или

переключателя

2.6. Какие значения могут принимать флажки и переключатели?

Возможные значения: true, false, null

**3 Вывод**

В ходе лабораторной работы я изучила свойства и процесс обработки событий элементов выбора в приложениях WPF.

**Лабораторная работа №31**

**Разработка приложения с использованием текстовых компонентов**

**1 Цель работы**

1.1 Изучить различные типы полей ввода, применяющихся в приложениях WPF,

1.2 Изучить свойства полей ввода и процесс обработки событий полей ввода.

**2 Контрольные вопросы**

2.1. Как задать имя элементам управления в WPF?

x:Name="Имя"

2.2. Как создать обработчик события в WPF?

В XAML: Click="Button\_Click"

В коде: private void Button\_Click(...).

2.3. Для чего используется Slider в WPF?

Slider позволяет выбрать числовое значение в диапазоне

2.4. Для чего используется TextBox в WPF?

TextBox предназначен для ввода текста.

2.5. Для чего используется TextBlock в WPF и чем он отличается от Label?

TextBlock отображает текст, а Label можно привязать к элементу

2.6. Для чего используется Calendar в WPF?

Calendar отображает даты

2.7. Для чего используется DatePicker в WPF?

DatePicker позволяет выбрать дату из выпадающего календаря

2.8. Для чего используется PasswordBox в WPF?

PasswordBox используется для ввода пароля

**3 Код**

Task 2

В MainWindow.xaml

<StackPanel>

<Label Content="Дата рождения:"/>

<DatePicker Name="DatePickerBirthday"

DisplayDateStart="01/01/1900"

SelectedDateChanged="DatePickerBirthday\_SelectedDateChanged" />

<TextBlock Name="AgeTextBlock"/>

</StackPanel>

В MainWindow.xaml.cs

public MainWindow()

{

InitializeComponent();

DatePickerBirthday.DisplayDateEnd = DateTime.Now;

}

private void DatePickerBirthday\_SelectedDateChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

if (DatePickerBirthday.SelectedDate.HasValue)

{

DateTime birthDate = DatePickerBirthday.SelectedDate.Value;

int age = CalculateAge(birthDate);

AgeTextBlock.Text = $"Возраст: {age} лет";

}

}

private int CalculateAge(DateTime birthDate)

{

int currentYear = DateTime.Now.Year;

int age = currentYear - birthDate.Year;

if (birthDate > new DateTime(currentYear, DateTime.Now.Month, DateTime.Now.Day))

age--;

return age;

}

Task3

В MainWindow.xaml

<StackPanel>

<Label Content="Стаж:"/>

<Slider Name="ExperienceSlider"

Minimum="0"

Maximum="50"

TickFrequency="5"

LargeChange="5"

SmallChange="1"

IsSnapToTickEnabled="True"

ValueChanged="ExperienceSlider\_ValueChanged"/>

<TextBox Name="ExperienceTextBox" IsReadOnly="True"/>

</StackPanel>

В MainWindow.xaml.cs

private void ExperienceSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

int experience = (int)e.NewValue;

ExperienceTextBox.Text = $"Стаж: {experience} {GetExperienceWord(experience)}";

}

private string GetExperienceWord(int experience)

{

if (experience % 10 == 1 && experience % 100 != 11)

{

return "год";

}

else if ((experience % 10 >= 2 && experience % 10 <= 4) && (experience % 100 < 10 || experience % 100 >= 20))

{

return "года";

}

else

{

return "лет";

}

}

**4 Вывод**

В ходе лабораторной работы я изучила различные типы полей ввода, свойства полей ввода и процесс обработки событий полей ввода, применяющихся в приложениях WPF

**Лабораторная работа №32**

**Разработка приложения с использованием элементов отображения списков**

**1 Цель работы**

1.1 Изучить свойства и процесс обработки событий элементов отображения  
списков в приложениях WPF.

**2 Контрольные вопросы**

2.1. Что такое ComboBox и для чего он используется?

ComboBox — выпадающий список для выбора одного варианта

2.2. Что такое ListBox и для чего он используется?

ListBox — список с возможностью выбора нескольких элементов

2.3. Какое событие срабатывает при выборе элемента в селекторе?

Событие SelectionChanged срабатывает при выборе элемента.

2.4. В каком свойстве хранятся элементы селекторов?

Элементы хранятся в свойстве Items

2.5. Какого типа элементы могут быть в селекторе?

Элементы могут быть любого типа

2.6. Какое свойство позволяет привязать селектор к набору данных?

Свойство ItemsSource связывает список с данными

2.7. Для чего используется свойство DisplayMemberPath в селекторе?

DisplayMemberPath указывает, какое поле объекта отображать

**3 Код**

5 заданий

В MainWindow.xaml

<Window.Resources>

<ItemsPanelTemplate x:Key="ListTemplate">

<VirtualizingStackPanel Orientation="Vertical"/>

</ItemsPanelTemplate>

<ItemsPanelTemplate x:Key="TileTemplate">

<WrapPanel Orientation="Horizontal"/>

</ItemsPanelTemplate>

<DataTemplate x:Key="ListDataTemplate">

<StackPanel Orientation="Horizontal" Margin="5">

<Image Source="C:\Temp\ispp34\LabWork32\LabWork32\bin\Debug\net8.0-windows\plant.png" Width="50" Height="50"/>

<TextBlock Text="{Binding Name}" Margin="10" VerticalAlignment="Center"/>

<TextBlock Text="{Binding Price, StringFormat={}{0:C}}" Margin="10" VerticalAlignment="Center"/>

<Button Content="Купить" Margin="10" Click="AddToCart\_Click"/>

</StackPanel>

</DataTemplate>

<DataTemplate x:Key="TileDataTemplate">

<Border BorderBrush="Gray" BorderThickness="1" Margin="5" Padding="5">

<StackPanel Width="150">

<Image Source="C:\Temp\ispp34\LabWork32\LabWork32\bin\Debug\net8.0-windows\plant.png" Width="100" Height="100" HorizontalAlignment="Center"/>

<TextBlock Text="{Binding Name}" HorizontalAlignment="Center" TextWrapping="Wrap"/>

<TextBlock Text="{Binding Price, StringFormat={}{0:C}}" HorizontalAlignment="Center"/>

<Button Content="Купить" Margin="5" Click="AddToCart\_Click"/>

</StackPanel>

</Border>

</DataTemplate>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="200"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Column="0" Grid.RowSpan="3" Margin="5">

<TextBlock Text="Категории:" FontWeight="Bold"/>

<ListBox Name="CategoryListBox" ItemsSource="{Binding Categories}" SelectionMode="Single"

ScrollViewer.HorizontalScrollBarVisibility="Auto">

<ListBox.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ListBox.ItemsPanel>

</ListBox>

<TextBlock Text="Наличие:" FontWeight="Bold" Margin="0,10,0,0"/>

<ListBox Name="AvailabilityListBox" SelectionMode="Single">

<CheckBox Content="В наличии" IsChecked="True"/>

<CheckBox Content="Под заказ: сегодня" IsChecked="True"/>

<CheckBox Content="Под заказ: завтра" IsChecked="True"/>

<CheckBox Content="Под заказ: позже" IsChecked="True"/>

<CheckBox Content="Нет в продаже"/>

</ListBox>

<Button Content="Применить" Margin="0,5" Click="ApplyAvailabilityFilter\_Click"/>

<TextBlock Name="SortText" Margin="0,10" Text="{Binding SelectedSort}"/>

<TextBlock Name="CategoryText" Text="{Binding SelectedCategory}"/>

<TextBlock Name="AvailabilityText"/>

</StackPanel>

<StackPanel Grid.Column="1" Grid.RowSpan="3" Margin="5">

<StackPanel Orientation="Horizontal">

<ComboBox Name="SortComboBox" Width="150" SelectedIndex="0">

<ComboBoxItem IsSelected="True">Сначала недорогие</ComboBoxItem>

<ComboBoxItem>Сначала дорогие</ComboBoxItem>

<ComboBoxItem>По новинкам</ComboBoxItem>

<ComboBoxItem>По скидке</ComboBoxItem>

<ComboBoxItem>По количеству отзывов</ComboBoxItem>

<ComboBoxItem>Сначала с лучшей оценкой</ComboBoxItem>

</ComboBox>

<ToggleButton Name="ViewToggle" Content="Плитка" Margin="10,0" Click="ViewToggle\_Click"/>

</StackPanel>

<ListView Name="ProductsListView" ItemsSource="{Binding Products}"

ItemTemplate="{StaticResource ListDataTemplate}"

ItemsPanel="{StaticResource ListTemplate}"/>

</StackPanel>

<StackPanel Grid.Column="2" Grid.RowSpan="3" Margin="5">

<TextBlock Text="Корзина:" FontWeight="Bold"/>

<ListBox Name="CartListBox" ItemsSource="{Binding Cart}"

SelectionMode="Extended" DisplayMemberPath="Name"

Height="300"/>

<Button Content="Удалить" Click="RemoveFromCart\_Click"/>

</StackPanel>

</Grid>

В MainWindow.xaml.cs

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Category { get; set; }

public decimal Price { get; set; }

}

public partial class MainWindow : Window

{

public ObservableCollection<Product> Products { get; set; }

public ObservableCollection<string> Categories { get; set; }

public ObservableCollection<Product> Cart { get; set; }

public MainWindow()

{

InitializeComponent();

Products = new ObservableCollection<Product>

{

new Product { Id = 1, Name = "Ноутбук", Category = "Электроника", Price = 50000 },

new Product { Id = 2, Name = "Смартфон", Category = "Электроника", Price = 30000 },

new Product { Id = 3, Name = "Книга", Category = "Книги", Price = 500 },

new Product { Id = 4, Name = "Наушники", Category = "Электроника", Price = 4000 },

new Product { Id = 5, Name = "Футболка", Category = "Одежда", Price = 1500 }

};

Categories = new ObservableCollection<string>

{

"Все категории", "Электроника", "Книги", "Одежда", "Мебель",

"Игрушки", "Спорт", "Красота", "Авто", "Еда"

};

Cart = new ObservableCollection<Product>();

DataContext = this;

CategoryListBox.SelectionChanged += (sender, e) =>

{

if (CategoryListBox.SelectedItem != null)

{

CategoryText.Text = $"Категория: {CategoryListBox.SelectedItem}";

}

};

SortComboBox.SelectionChanged += (sender, e) =>

{

if (SortComboBox.SelectedItem != null)

{

SortText.Text = $"Сортировка: {((ComboBoxItem)SortComboBox.SelectedItem).Content}";

}

};

}

private void AddToCart\_Click(object sender, RoutedEventArgs e)

{

if (((Button)sender).DataContext is Product product)

{

Cart.Add(product);

}

}

private void RemoveFromCart\_Click(object sender, RoutedEventArgs e)

{

var selectedItems = CartListBox.SelectedItems.Cast<Product>().ToList();

foreach (var item in selectedItems)

{

Cart.Remove(item);

}

}

private void ApplyAvailabilityFilter\_Click(object sender, RoutedEventArgs e)

{

var selectedOptions = new List<string>();

foreach (var item in AvailabilityListBox.Items)

{

if (item is CheckBox checkBox && checkBox.IsChecked == true)

{

selectedOptions.Add(checkBox.Content.ToString());

}

}

AvailabilityText.Text = "Наличие: " + string.Join(", ", selectedOptions);

}

private void ViewToggle\_Click(object sender, RoutedEventArgs e)

{

if (ViewToggle.IsChecked == true)

{

ProductsListView.ItemsPanel = (ItemsPanelTemplate)FindResource("TileTemplate");

ProductsListView.ItemTemplate = (DataTemplate)FindResource("TileDataTemplate");

ViewToggle.Content = "Список";

}

else

{

ProductsListView.ItemsPanel = (ItemsPanelTemplate)FindResource("ListTemplate");

ProductsListView.ItemTemplate = (DataTemplate)FindResource("ListDataTemplate");

ViewToggle.Content = "Плитка";

}

}

**4 Вывод**

В ходе лабораторной работы я изучила свойства и процесс обработки событий элементов отображения списков в приложениях WPF.

**Лабораторная работа №33**

**Разработка приложения для отображения данных в табличном виде**

**1 Цель работы**

1.1 Изучить свойства и процесс настройки внешнего вида элемента DataGrid и ListView в приложениях WPF.

**2 Контрольные вопросы**

2.1. Что такое DataGrid и для чего он используется?

DataGrid — таблица для отображения и редактирования данных

2.2. Какие типы столбцов поддерживаются в DataGrid?

Поддерживаемые типы столбцов: текстовые, выпадающие списки, кнопки и др

2.3. Как добавить кнопку в строки DataGrid?

Кнопка добавляется через DataGridTemplateColumn

2.4. Как указать источник данных для DataGrid?

Источник данных задаётся через ItemsSource

2.5. Как указать источник данных для выпадающего списка DataGrid?

Указать ItemsSource в столбце.

**3 Вывод**

В ходе лабораторной работы я изучила свойства и процесс настройки внешнего вида элемента DataGrid и ListView в приложениях WPF.

**Лабораторная работа №34**

**Разработка приложения с меню и панелью инструментов**

**1 Цель работы**

1.1 Изучить свойства и процесс настройки внешнего вида меню и панели инструментов в приложениях WPF.

**2 Контрольные вопросы**

2.1. Что такое Menu и для чего он используется?

Menu — главное меню приложения

2.2. Что такое ContextMenu и для чего он используется?

ContextMenu — контекстное меню

2.3. Что такое ToolBar и для чего он используется?

ToolBar — панель с кнопками для быстрого доступа

2.4. Что такое StatusBar и для чего он используется?

StatusBar — строка состояния

**3 Вывод**

В ходе лабораторной работы я изучила свойства и процесс настройки внешнего вида меню и панели инструментов в приложениях WPF.

**Лабораторная работа №35**

**Создание проекта с использованием компонентов стандартных диалогов**

**1 Цель работы**

1.1 Изучить процесс создания и применения стандартных диалоговых окон в  
приложениях WPF.

**2 Контрольные вопросы**

2.1. Что такое диалоговое окно?

Диалоговое окно — это вспомогательное окно для взаимодействия с пользователем

2.2. Для чего используется OpenFileDialog?

OpenFileDialog — для выбора файла

2.3. Для чего используется SaveFileDialog?

SaveFileDialog — для сохранения файла

2.4. Что такое MessageBox и какие настройки можно ему задать?

MessageBox — окно с сообщением

2.5. Для чего используется PrintDialog?

PrintDialog — для отправки данных на печать

**3 Код**

5 заданий

Task 1-3

В MainWindows.xaml

<DockPanel>

<Button Background="White" x:Name="OpenButton" Content="Открыть" Click="OpenButton\_Click" DockPanel.Dock="Top"/>

<Button Background="LightBlue" x:Name="SaveButton" Content="Сохранить" Click="SaveButton\_Click" DockPanel.Dock="Top"/>

<TextBox Background="Aquamarine"

x:Name="FileTextBox"

AcceptsReturn="True"

TextWrapping="Wrap"

VerticalScrollBarVisibility="Auto" Height="395"

/>

</DockPanel>

В MainWindows.xaml.cs

private void OpenButton\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog dialog = new();

dialog.Filter = "текстовые файлы|\*.txt;\*.cs;\*.html;\*.css;\*.js;\*.sql";

if (dialog.ShowDialog() == true)

{

FileTextBox.Text = File.ReadAllText(dialog.FileName);

this.Title = dialog.FileName;

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog dialog = new();

dialog.Filter = "текстовые файлы|\*.txt;\*.cs;\*.html;\*.css;\*.js;\*.sql";

if (dialog.ShowDialog() == true)

{

File.WriteAllText(dialog.FileName, FileTextBox.Text);

MessageBox.Show("вы сохранили файл",

"успех",

MessageBoxButton.OK,

MessageBoxImage.Asterisk);

}

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if (MessageBox.Show("вы точно хотите закрыть приложение?",

"успех",

MessageBoxButton.YesNo,

MessageBoxImage.Question) == MessageBoxResult.No)

e.Cancel = true;

}

Task 4

В MainWindows.xaml

<DockPanel>

<Button x:Name="OpenButton" Content="открыть картинки" Click="OpenButton\_Click" DockPanel.Dock="Top"/>

<ListView x:Name="ImageListView" ScrollViewer.HorizontalScrollBarVisibility="Disabled">

<ListView.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel/>

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<Image Source="{Binding}" Height="100"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</DockPanel>

В MainWindows.xaml.cs

private void OpenButton\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog dialog = new();

dialog.Filter = "изображения|\*.png;\*.jpg;\*.jpeg;\*.bmp";

dialog.Multiselect = true;

if (dialog.ShowDialog() == true)

{

ImageListView.ItemsSource = dialog.FileNames;

}

}

Task 5

В MainWindows.xaml

<DockPanel>

<Button x:Name="PrintButton" Content="Печать" DockPanel.Dock="Top" Click="PrintButton\_Click"/>

<Canvas x:Name="PrintCanvas">

<Ellipse Canvas.Top="200" Width="200" Height="200" Fill="DarkGreen" HorizontalAlignment="Left" Canvas.Left="123" VerticalAlignment="Center"/>

<Rectangle Canvas.Right="10" Canvas.Top="200" Width="200" Height="200" Fill="DarkCyan" HorizontalAlignment="Left" Canvas.Left="463" VerticalAlignment="Center"/>

<TextBlock Canvas.Left="200" Text="это можно напечатать" FontSize="30"/>

</Canvas>

</DockPanel>

В MainWindows.xaml.cs

private void PrintButton\_Click(object sender, RoutedEventArgs e)

{

PrintDialog printDialog = new();

if (printDialog.ShowDialog() == true)

{

printDialog.PrintVisual(PrintCanvas, "");

}

}

**4 Вывод**

В ходе лабораторной работы я изучила процесс создания и применения стандартных диалоговых окон в приложениях WPF.

**Лабораторная работа №36**

**Разработка приложения с несколькими формами**

**1 Цель работы**

1.1 Изучить процесс создания и применения пользовательских окон в приложениях WPF.

**2 Контрольные вопросы**

2.1. Как сделать доступными данные пользовательского диалогового окна вызывающим его окнам?

Данные диалогового окна делаются доступными через публичные свойства

2.2. Какие значения может принимать переменная DialogResult?

DialogResult может принимать значения true или false

2.3. Как открыть окно в диалоговом режиме?

Используется ShowDialog()

2.4. Как открыть окно в недиалоговом режиме?

Используется Show()

2.5. В чем отличие между диалоговым и недиалоговым режимами работы?

Диалоговое окно блокирует родительское, недиалоговое — нет

**3 Код**

5 заданий

Task 1-3

В MainWindow.xaml

<DockPanel>

<Menu DockPanel.Dock="Top">

<MenuItem Header="Файл">

<MenuItem Header="Новое окно" Click="MenuItem\_Click"/>

</MenuItem>

</Menu>

<TextBox TextWrapping="Wrap" AcceptsReturn="True"/>

</DockPanel>

В MainWindow.xaml.cs

rivate void MenuItem\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

CancelDialog dialog = new CancelDialog();

dialog.Owner = this;

if(dialog.ShowDialog()!=true)

e.Cancel = true;

}

В CancelDialog.xaml

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="Question.png" Grid.RowSpan="2"/>

<Label Content="Вы хотите закрыть приложение?" Grid.Column="1" Grid.ColumnSpan="2" FontSize="15" VerticalAlignment="Bottom"/>

<Button x:Name="YesButton" Content="Да" Grid.Column="1" Grid.Row="2"

Height="30" Width="70" Background="LightGreen" IsDefault="True" Click="YesButton\_Click"/>

<Button x:Name="NoButton" Content="Нет" Grid.Column="2" Grid.Row="2"

Height="30" Width="70" Background="Pink" IsCancel="True" Click="NoButton\_Click"/>

</Grid>

В CancelDialog.xaml.cs

private void YesButton\_Click(object sender, RoutedEventArgs e)

{

DialogResult = true;

}

private void NoButton\_Click(object sender, RoutedEventArgs e)

{

Close();

}

Task 4

В AdminWindow.xaml

<Grid>

<Button x:Name="BackButton" Content="Назад" Width="50" VerticalAlignment="Top" HorizontalAlignment="Left" Click="BackButton\_Click"/>

</Grid>

В AdminWindow.xaml.cs

private void BackButton\_Click(object sender, RoutedEventArgs e)

{

Close();

}

В ManagerWindow.xaml

<Grid>

<Button x:Name="BackButton" Content="Назад" Width="50" VerticalAlignment="Top" HorizontalAlignment="Left" Click="BackButton\_Click"/>

</Grid>

В ManagerWindow.xaml.cs

private void BackButton\_Click(object sender, RoutedEventArgs e)

{

Close();

}

В AuthorizationWindow.xaml

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label Content="Логин"/>

<TextBox x:Name="LoginTextBox" Grid.Column="1"/>

<Label Content="Пароль" Grid.Row="1"/>

<PasswordBox x:Name="PasswordBox" Grid.Column="1" Grid.Row="1"/>

<StackPanel Grid.Row="2">

<Button x:Name="EnterButton" Content="Войти" Click="EnterButton\_Click"/>

<Button x:Name="CancelButton" Content="Отмена" Click="CancelButton\_Click"/>

</StackPanel>

</Grid>

В AuthorizationWindow.xaml.cs

private void EnterButton\_Click(object sender, RoutedEventArgs e)

{

if(LoginTextBox.Text=="admin" && PasswordBox.Password=="qwerty")

{

AdminWindow adminWindow = new AdminWindow();

Hide();

adminWindow.ShowDialog();

ShowDialog();

return;

}

if (LoginTextBox.Text == "manager" && PasswordBox.Password == "12345")

{

ManagerWindow managerWindow = new ManagerWindow();

Hide();

managerWindow.ShowDialog();

ShowDialog();

return;

}

MessageBox.Show("Пользователя не существует", "Некорректные данные", MessageBoxButton.OK, MessageBoxImage.Warning);

}

private void CancelButton\_Click(object sender, RoutedEventArgs e)

{

Close();

}

В MainWindow.xaml

<Grid>

<Button x:Name="AuthorizationButton" Content="Авторизоваться"

VerticalAlignment="Top" Click="AuthorizationButton\_Click"/>

</Grid>

В MainWindow.xaml.cs

private void AuthorizationButton\_Click(object sender, RoutedEventArgs e)

{

AuthorizationWindow authorizationWindow = new AuthorizationWindow();

Hide();

authorizationWindow.ShowDialog();

Show();

}

Task 5

В ColorDialog.xaml

<StackPanel Slider.ValueChanged="StackPanel\_ValueChanged">

<DockPanel>

<Label Content="Red" Width="45"/>

<Slider x:Name="RedSlider" Minimum="0" Maximum="255" />

</DockPanel>

<DockPanel>

<Label Content="Green" Width="45"/>

<Slider x:Name="GreenSlider" Minimum="0" Maximum="255"/>

</DockPanel>

<DockPanel>

<Label Content="Blue" Width="45"/>

<Slider x:Name="BlueSlider" Minimum="0" Maximum="255"/>

</DockPanel>

<WrapPanel>

<Button x:Name="OKButton" Content="OK" Width="50" IsDefault="True" Click="OKButton\_Click"/>

<Button x:Name="CancelButton" Content="Отмена" Width="50" IsCancel="True"/>

</WrapPanel>

<Rectangle x:Name="Rectangle" Width="70" Height="50" StrokeThickness="1" Stroke="Black"/>

</StackPanel>

В ColorDialog.xaml.cs

public SolidColorBrush ColorValue => new(Color.FromRgb((byte)RedSlider.Value, (byte)GreenSlider.Value, (byte)BlueSlider.Value));

public ColorDialog()

{

InitializeComponent();

}

private void OKButton\_Click(object sender, RoutedEventArgs e)

{

DialogResult = true;

}

private void StackPanel\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

Rectangle.Fill = ColorValue;

}

В MainWindow.xaml

<Grid>

<Button x:Name="ColorButton" Content="Color" Click="ColorButton\_Click" VerticalAlignment="Top"/>

</Grid>

В MainWindow.xaml.cs

private void ColorButton\_Click(object sender, RoutedEventArgs e)

{

ColorDialog colorDialog = new ColorDialog();

Hide();

if (colorDialog.ShowDialog() == true)

Background = colorDialog.ColorValue;

Show();

}

**4 Вывод**

В ходе лабораторной работы я изучила процесс создания и применения пользовательских окон в приложениях WPF.

**Лабораторная работа №37**

**Реализация фильтрации данных**

**1 Цель работы**

1.1 Научиться применять LINQ-запросы для фильтрации данных по одному критерию и набору критериев.

**2 Контрольные вопросы**

2.1. Для чего используется метод Where?

Where фильтрует данные по условию

2.2. Какие логические операторы могут применяться при составлении условий?

Логические операторы: И, ИЛИ, НЕ

2.3. Как проверить, что строка начинается с определенного текста?

Для проверки начала строки используется StartsWith

2.4. Как проверить, что строка содержит определенный текст?

 Для проверки содержания подстроки — Contains

2.5. Как составить LINQ-запрос для фильтрации по нескольким критериям?

Применять Where() несколько раз или объединять условия через И и ИЛИ

**3 Вывод**

В ходе лабораторной работы я научилась применять LINQ-запросы для фильтрации данных по одному критерию и набору критериев.

**Лабораторная работа №38**

**Реализация постраничного вывода информации**

**1 Цель работы**

1.1 Научиться применять LINQ-запросы для постраничного вывода данных.

**2 Контрольные вопросы**

2.1. Почему может потребоваться выводить данные постранично?

 Постраничный вывод уменьшает нагрузку при работе с большими данными

2.2. Что такое «пагинация»?

Пагинация — это разбиение данных на страницы

2.3. Для чего используется метод Take?

Take выбирает указанное количество элементов

2.4. Для чего используется метод Skip?

Skip пропускает указанное количество элементов

2.5. Для чего используется метод TakeWhile?

TakeWhile выбирает элементы, пока выполняется условие

2.6. Для чего используется метод SkipWhile?

SkipWhile пропускает элементы, пока выполняется условие

2.7. Каким должен быть набор данных, чтобы можно было использовать Take и Skip?

 Данные должны быть упорядочены

**3 Код**

Task 2-4

В MainWindow.xaml

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBox x:Name="FilterBox" Grid.Row="0" Margin="2" TextChanged="FilterBox\_TextCanged"/>

<ListView x:Name="FilesListView" Grid.Row="1">

<ListView.View>

<GridView>

<GridViewColumn Header="Имя файла" DisplayMemberBinding="{Binding Name}" Width="auto"/>

<GridViewColumn Header="Размер (KB)" DisplayMemberBinding="{Binding Size}" Width="auto"/>

<GridViewColumn Header="Дата создания" Width="auto">

<GridViewColumn.DisplayMemberBinding>

<Binding Path="Created" StringFormat="dd.MM.yyyy HH:mm"/>

</GridViewColumn.DisplayMemberBinding>

</GridViewColumn>

</GridView>

</ListView.View>

</ListView>

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" Margin="10">

<Button x:Name="FirstPageButton" Content="|&lt;" Width="30" Margin="2" Click="FirstPageButton\_Click"/>

<Button x:Name="PrevPageButton" Content="&lt;" Width="30" Margin="2" Click="PrevPageButton\_Click"/>

<TextBox x:Name="PageNumberTextBox" Width="30" Margin="2" VerticalAlignment="Center"

Text="{Binding CurrentPage}" KeyDown="PageNumberTextBox\_KeyDown"/>

<Button x:Name="NextPageButton" Content="&gt;" Width="30" Margin="2" Click="NextPageButton\_Click"/>

<Button x:Name="LastPageButton" Content="&gt;|" Width="30" Margin="2" Click="LastPageButton\_Click"/>

<TextBlock x:Name="PageInfoText" VerticalAlignment="Stretch"/>

</StackPanel>

</Grid>

В MainWindow.xaml.cs

public int PageSize { get; set; } = 5;

public int CurrentPage

{

get => \_currentPage;

set

{

\_currentPage = value;

PageNumberTextBox.Text = value.ToString();

UpDateNavigation();

ShowFiles();

}

}

public int \_currentPage = 1;

public int PagesCount => (int)Math.Ceiling((double)filteredFiles.Count / PageSize);

private FileInfo[] allFiles;

private List<FileInfo> filteredFiles;

public MainWindow()

{

InitializeComponent();

LoadFiles();

CurrentPage = 1;

}

private void LoadFiles()

{

var path = "Y:\\МДК.01.01\\Images";

var directory = new DirectoryInfo(path);

allFiles = directory.GetFiles()

.OrderBy(f => f.Name)

.ToArray();

filteredFiles = allFiles.ToList();

}

private void ShowFiles()

{

if (filteredFiles == null || filteredFiles.Count == 0)

{

FilesListView.ItemsSource = null;

return;

}

var filesToShow = filteredFiles

.Skip((CurrentPage - 1) \* PageSize)

.Take(PageSize)

.Select(f => new

{

f.Name,

Size = $"{f.Length / 1024.0:F2} KB",

Created = f.CreationTime

});

FilesListView.ItemsSource = filesToShow;

}

private void UpDateNavigation()

{

FirstPageButton.IsEnabled = PrevPageButton.IsEnabled = CurrentPage > 1;

NextPageButton.IsEnabled = LastPageButton.IsEnabled = CurrentPage < PagesCount;

PageInfoText.Text = $"Показано {CurrentPage} из {PagesCount} страниц";

}

private void FirstPageButton\_Click(object sender, RoutedEventArgs e)

{

CurrentPage = 1;

ShowFiles();

}

private void PrevPageButton\_Click(object sender, RoutedEventArgs e)

{

if (CurrentPage > 1)

{

CurrentPage--;

ShowFiles();

}

}

private void NextPageButton\_Click(object sender, RoutedEventArgs e)

{

if (CurrentPage < PagesCount)

{

CurrentPage++;

ShowFiles();

}

}

private void LastPageButton\_Click(object sender, RoutedEventArgs e)

{

CurrentPage = PagesCount;

ShowFiles();

}

private void PageNumberTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

if (int.TryParse(PageNumberTextBox.Text, out int pageNumber))

{

if (pageNumber >= 1 && pageNumber <= PagesCount)

{

CurrentPage = pageNumber;

ShowFiles();

}

else

{

MessageBox.Show($"Введите номер страницы от 1 до {PagesCount}");

}

}

else

{

MessageBox.Show("Введите корректный номер страницы");

}

}

}

private void FilterBox\_TextCanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

var filter = FilterBox.Text.ToLower();

filteredFiles = allFiles

.Where(f => f.Name.ToLower().Contains(filter))

.ToList();

CurrentPage = 1;

UpDateNavigation();

ShowFiles();

}

**4 Вывод**

В ходе лабораторной работы я научилась применять LINQ-запросы для постраничного вывода данных

**Лабораторная работа №39**

**Реализация группировки и соединения данных**

**1 Цель работы**

1.1 Научиться применять LINQ-запросы для группировки и соединения данных.

**2 Контрольные вопросы**

2.1. Какие агрегатные функции поддерживаются в LINQ?

Агрегатные функции: Sum, Average, Count, Min, Max

2.2. Что возвращает метод Distinct?

Distinct возвращает уникальные элементы

2.3. Для чего используется метод GroupBy?

GroupBy группирует данные по ключу

2.4. Для чего используется метод Join?

Join объединяет данные из двух коллекций по общему полю

2.5. Для чего используется метод GroupJoin?

GroupJoin группирует элементы одной коллекции по другой

2.6. В чем отличие результатов Concat, Union, Except, Intersect?

Concat — объединяет две коллекции в одну

Union — объединяет две коллекции, удаляя дубликаты

Except — озвращает элементы первой коллекции, которых нет во второй

Intersect — возвращает общие элементы

**3 Код**

5 заданий

Task 1

В xaml

<ListView x:Name="listView">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<VerticalStackLayout Padding="10">

<Label Text="{Binding Extension}" FontAttributes="Bold"/>

<Label Text="{Binding TotalSize, StringFormat='Общий размер: {0} байт'}"/>

<Label Text="{Binding MinSize, StringFormat='Минимальный размер: {0} байт'}"/>

<Label Text="{Binding MaxSize, StringFormat='Максимальный размер: {0} байт'}"/>

<Label Text="{Binding FileCount, StringFormat='Количество файлов: {0}'}"/>

</VerticalStackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

В xaml.cs

public Task1Page()

{

InitializeComponent();

var query = Files

.GroupBy(f => f.Extension.ToLower())

.Select(g => new {

Extension = string.IsNullOrEmpty(g.Key) ? "Без расширения" : g.Key,

TotalSize = g.Sum(f => f.Length),

MinSize = g.Min(f => f.Length),

MaxSize = g.Max(f => f.Length),

FileCount = g.Count()

})

.OrderByDescending(x => x.FileCount);

listView.ItemsSource = query.ToList();

}

Task 2

В xaml

<ListView x:Name="listView">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<VerticalStackLayout Padding="10">

<Label Text="{Binding Year, StringFormat='Год: {0}'}" FontAttributes="Bold"/>

<Label Text="{Binding Month, StringFormat='Месяц: {0}'}"/>

<Label Text="{Binding FileCount, StringFormat='Файлов: {0}'}"/>

</VerticalStackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

В xaml.cs

public Task2Page()

{

InitializeComponent();

var query = Files

.GroupBy(f => new {

f.CreationTime.Year,

f.CreationTime.Month

})

.Select(g => new {

g.Key.Year,

g.Key.Month,

FileCount = g.Count()

})

.OrderBy(x => x.Year)

.ThenBy(x => x.Month);

listView.ItemsSource = query.ToList();

}

Task 3

В xaml

<ListView x:Name="listView">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<VerticalStackLayout Padding="10">

<Label Text="{Binding Name}" FontAttributes="Bold"/>

<Label Text="{Binding Path}"/>

<Label Text="{Binding CreationDate, StringFormat='Создан: {0:dd.MM.yyyy HH:mm}'}"/>

</VerticalStackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

В xaml.cs

public Task3Page()

{

InitializeComponent();

var filesInfo = Files.Select(f => new {

f.Name,

Path = f.DirectoryName,

CreationDate = f.CreationTime

});

var directoriesInfo = Directories.Select(d => new {

d.Name,

Path = d.FullName,

CreationDate = d.CreationTime

});

var query = filesInfo.Union(directoriesInfo)

.OrderBy(x => x.CreationDate);

listView.ItemsSource = query.ToList();

}

Task 4

В xaml

<ListView x:Name="listView">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<Label Text="{Binding .}" Padding="10"/>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

В xaml.cs

public Task4Page()

{

InitializeComponent();

var today = DateTime.Today;

var query = Directories

.Where(d => d.CreationTime.Date == today)

.Join(Files,

d => d.FullName,

f => f.DirectoryName,

(d, f) => f.FullName)

.Distinct();

listView.ItemsSource = query.ToList();

}

Task 5

В xaml

<ListView x:Name="listView">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<VerticalStackLayout Padding="10">

<Label Text="{Binding FolderPath}" FontAttributes="Bold"/>

<Label Text="{Binding FileCount, StringFormat='Файлов: {0}'}"/>

</VerticalStackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

В xaml.cs

public Task5Page()

{

InitializeComponent();

var query = Directories

.GroupJoin(Files,

d => d.FullName,

f => f.DirectoryName,

(d, files) => new {

FolderPath = d.FullName,

FileCount = files.Count()

})

.OrderByDescending(x => x.FileCount);

listView.ItemsSource = query.ToList();

}

**4 Вывод**

В ходе лабораторной работы я научилась применять LINQ-запросы для группировки и соединения данных.

**Лабораторная работа №40**

**Разработка приложения для работы с графикой**

**1 Цель работы**

1.1 Изучить процесс рисования и трансформации объектов в приложениях WPF.

**2 Контрольные вопросы**

2.1. Какие графические примитивы доступны в приложениях WPF?

Графические примитивы: линии, эллипсы, прямоугольники, пути, полигоны

2.2. Для чего используется Path?

Path используется для рисования сложных фигур

2.3. Какие виды трансформаций объектов доступны в WPF?

Виды трансформаций: поворот, масштабирование, сдвиг, наклон

2.4. Как указать заливку и контур графических объектов в WPF?

 Заливка и контур задаются через свойства Fill и Stroke

**3 Вывод**

В ходе лабораторной работы я изучила процесс рисования и трансформации объектов в приложениях WPF

**Лабораторная работа №41**

**Разработка приложения с анимацией**

**1 Цель работы**

1.1 Изучить процесс анимации объектов в приложениях WPF.

**2 Контрольные вопросы**

2.1. Как задать анимацию размеров в WPF?

Анимация размеров задаётся через DoubleAnimation

2.2. Как задать анимацию цвета в WPF?

Анимация цвета — через ColorAnimation

2.3. Чем отличается покадровая анимация от плавной анимации в WPF?

Покадровая анимация задаёт ключевые кадры, плавная — изменяет значения по времени

2.4. Какие свойства управляют анимацией и зачем они нужны?

Свойства: RepeatBehavior - повтор, Duration - длительность, AutoReverse - возврат к началу

**3 Вывод**

В ходе лабораторной работы я изучила процесс анимации объектов в приложениях WPF

**Лабораторная работа №42**

**Разработка мультимедиа-приложения**

**1 Цель работы**

1.1 Изучить процесс создания мультимедиа-приложений на WPF.

**2 Контрольные вопросы**

2.1. Что такое «мультимедиа»?

 Мультимедиа — это работа с видео, аудио и изображениями

2.2. Какие элементы отображают изображения в WPF?

 Элементы для изображений: Image, InkCanvas

2.3. Какие элементы воспроизводят аудио в WPF?

Элементы для аудио: MediaElement, SoundPlayer

2.4. Какие элементы воспроизводят видео в WPF?

 Элемент для видео: MediaElement

**3 Вывод**

В ходе лабораторной работы я изучила процесс создания мультимедиа-приложений на WPF.

**Лабораторная работа №43**

**Разработка игрового приложения**

**1 Цель работы**

1.1 Изучить процесс разработки игровых приложений WPF, использующих  
графику, обработчики событий и таймеры.

**2 Контрольные вопросы**

2.1. Как подключить таймер к приложению на WPF?

Для подключения таймера используется DispatcherTimer

2.2. Как создать обработчик события для таймера в приложении WPF?

Обработчик события таймера привязывается к Tick

2.3. Как изменить интервал таймера в приложении WPF?

 Интервал таймера изменяется через свойство Interval

**3 Вывод**

В ходе лабораторной работы я изучила процесс разработки игровых приложений WPF, использующих графику, обработчики событий и таймеры

**Лабораторная работа №44**

**Создание БД**

**1 Цель работы**

1.1 Изучить процесс создания таблиц и связей между ними в реляционной СУБД.

**2 Контрольные вопросы**

2.1. Что такое «система управления базами данных»?

 СУБД — это программа для управления базами данных

2.2. Что такое «база данных»?

 База данных — это структурированный набор данных

2.3. Какие СУБД являются реляционными (примеры)?

Примеры реляционных СУБД: MSSQL, SQLite, MySQL

**3 Код**

Task 3

SqliteConnectionStringBuilder sqliteStringBuilder = new()

{

DataSource = @"gamesdb.sqlite"

};

var connectionString = sqliteStringBuilder.ConnectionString;

SqliteConnection connection = new(connectionString);

connection.Open();

var createGameTable = @"CREATE TABLE Game (

Id INTEGER PRIMARY KEY AUTOINCREMENT,

Title TEXT NOT NULL,

Description TEXT NOT NULL,

PublicationYear INTEGER NOT NULL,

Price REAL NOT NULL

)";

SqliteCommand gameCommand = new(createGameTable, connection);

gameCommand.ExecuteNonQuery();

var createReviewTable = @"

CREATE TABLE Review (

Id INTEGER PRIMARY KEY AUTOINCREMENT,

GameId INTEGER NOT NULL,

User TEXT NOT NULL,

Comment TEXT NOT NULL,

PublicationDate TEXT NOT NULL,

FOREIGN KEY (GameId) REFERENCES Game(Id)

)";

SqliteCommand reviewCommand = new(createReviewTable, connection);

reviewCommand.ExecuteNonQuery();

var insertGamesQuery = @"INSERT INTO Game VALUES

(1, 'Cyberpunk 2077', 'Игра в открытом мире', 2020, 59.99),

(2, 'God of War', 'Экшен-приключенческая игра', 2018, 49.99),

(3, 'The Witcher 3', 'Открытый RPG', 2015, 39.99),

(4, 'Dark Souls III', 'Экшен-RPG', 2016, 39.99),

(5, 'Hollow Knight', 'Метроидвания', 2017, 14.99

)";

SqliteCommand insertGames = new(insertGamesQuery, connection);

insertGames.ExecuteNonQuery();

var insertReviewsQuery = @"INSERT INTO Review (GameId, User, Comment, PublicationDate) VALUES

(1, 'Пользователь1', 'Удивительная игра с потрясающей графикой!', '2025-05-01'),

(1, 'Пользователь2', 'Сложная, но вознаграждающая игра.', '2025-05-02'),

(2, 'Пользователь3', 'Шедевр повествования.', '2025-05-03'),

(2, 'Пользователь4', 'Боевая система плавная и увлекательная.', '2025-05-04'),

(3, 'Пользователь5', 'Лучшая RPG, в которую я когда-либо играл!', '2025-05-05'),

(3, 'Пользователь6', 'Мир настолько погружающий.', '2025-05-06'),

(4, 'Пользователь7', 'Очень сложно, но стоит того.', '2025-05-07'),

(4, 'Пользователь8', 'Мифология невероятно глубокая.', '2025-05-08'),

(5, 'Пользователь9', 'Красивый художественный стиль и отличный игровой процесс.', '2025-05-09'),

(5, 'Пользователь10', 'Обязательно к игре для фанатов инди-игр.', '2025-05-10')";

SqliteCommand insertReviews = new(insertReviewsQuery, connection);

insertReviews.ExecuteNonQuery();

Task 4

SqlConnectionStringBuilder builder = new()

{

DataSource = "mssql",

UserID = "ispp3110",

Password = "3110",

InitialCatalog = "ispp3110",

TrustServerCertificate = true

};

var connectionString = builder.ConnectionString;

Console.WriteLine(connectionString);

using SqlConnection connection = new(connectionString);

connection.Open();

var rolesQuery = @"

CREATE TABLE Roles (

Id INT IDENTITY(1,1) PRIMARY KEY,

Name NVARCHAR(255) NOT NULL

)";

SqlCommand rolesCommand = new(rolesQuery, connection);

rolesCommand.ExecuteNonQuery();

var usersQuery = @"

CREATE TABLE Users (

Id INT IDENTITY(1,1) PRIMARY KEY,

Login NVARCHAR(255) NOT NULL,

PASSWORD NVARCHAR(255),

FOREIGN KEY (RoleId) REFERENCES Roles(Id)

)";

SqlCommand usersCommand = new(usersQuery, connection);

usersCommand.ExecuteNonQuery();

connection.Close();

**4 Вывод**

В ходе лабораторной работы я изучила процесс создания таблиц и связей между ними в реляционной СУБД.

**Лабораторная работа №45**

**Создание приложения с БД для чтения данных**

**1 Цель работы**

1.1 Научиться создавать приложения для чтения данных из БД.

**2 Контрольные вопросы**

2.1. Каково назначение элемента SqlConnection?

SqlConnection устанавливает соединение с БД

2.2. Как считать из БД одно значение?

 Для чтения одного значения используется ExecuteScalar()

2.3. Каково назначение элемента SqlDataReader?

SqlDataReader читает данные построчно

2.4. Какие пространства имён подключать для MS SQL?

Пространства имён: System.Data, System.Data.SqlClient

**3 Вывод**

В ходе лабораторной работы я научилась создавать приложения для чтения данных из БД

**Лабораторная работа №46**

**Создание приложения с БД для записи данных**

**1 Цель работы**

1.1 Научиться создавать приложения для чтения данных из БД.

**2 Контрольные вопросы**

2.1. Как изменить данные в БД, используя SqlCommand?

Для изменения данных используется ExecuteNonQuery()

2.2. Как передать параметры в SqlCommand?

Параметры передаются через command.Parameters.AddWithValue()

2.3. Для чего используются параметры в SqlCommand?

Параметры используются чтобы сделать запросы гибкими.

2.4. Как изменить данные в БД, используя SqlDataAdapter?

SqlDataAdapter изменяет данные через DataTable

2.5. Для чего применяется SqlCommandBuilder?

SqlCommandBuilder автоматически генерирует SQL-команды для SqlDataAdapter.

2.6. Как связать SqlCommandBuilder и SqlDataAdapter?

Создать CommandBuilder и передать в него адаптер: new SqlCommandBuilder(adapter).

**3 Код**

3 задания

В IDatabase.cs

public interface IDatabase

{

int ExecuteQuery(string query);

public bool UpdateGame(int id, string newTitle, decimal newPrice)

{

string query = $"UPDATE Games SET Title = '{newTitle.Replace("'", "''")}', " +

$"Price = {newPrice.ToString(CultureInfo.InvariantCulture)} " +

$"WHERE Id = {id}";

int rowsAffected = ExecuteQuery(query);

return rowsAffected == 1;

}

}

В SqlDatabase.cs

public class SqlDatabase : IDatabase

{

private readonly string \_connectionString;

public SqlDatabase(string server, string database, string username, string password)

{

var builder = new SqlConnectionStringBuilder

{

DataSource = server,

InitialCatalog = database,

UserID = username,

Password = password,

TrustServerCertificate = true

};

\_connectionString = builder.ConnectionString;

}

public int ExecuteQuery(string query)

{

using var connection = new SqlConnection(\_connectionString);

connection.Open();

using var command = new SqlCommand(query, connection);

return command.ExecuteNonQuery();

}

public bool UpdateGame(int id, string newTitle, decimal newPrice)

{

string query = $"UPDATE Game SET Title = '{newTitle}', Price = {newPrice} WHERE Id = {id}";

return ExecuteQuery(query) == 1;

}

}

В SqliteDatabase.cs

public class SqliteDatabase : IDatabase

{

private readonly string \_connectionString;

public SqliteDatabase(string filePath)

{

var builder = new SqliteConnectionStringBuilder

{

DataSource = filePath

};

\_connectionString = builder.ConnectionString;

}

public int ExecuteQuery(string query)

{

using var connection = new SqliteConnection(\_connectionString);

connection.Open();

using var command = new SqliteCommand(query, connection);

return command.ExecuteNonQuery();

}

public bool UpdateGame(int id, string newTitle, double newPrice)

{

using var connection = new SqliteConnection(\_connectionString);

connection.Open();

var command = new SqliteCommand(

"UPDATE Game SET Title = @Title, Price = @Price WHERE Id = @Id",

connection);

command.Parameters.AddWithValue("@Title", newTitle);

command.Parameters.AddWithValue("@Price", newPrice);

command.Parameters.AddWithValue("@Id", id);

return command.ExecuteNonQuery() == 1;

}

}

В Form1.cs

public partial class Form1 : Form

{

private IDatabase \_database;

public Form1()

{

InitializeComponent();

\_database = new SqlDatabase("mssql", "ispp3110", "ispp3110", "3110");

//\_database = new SqliteDatabase("C:\Temp\ispp31\GamesStore.sqlite");

Debug.WriteLine(\_database.ToString());

}

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

{

try

{

string query = "UPDATE Game SET Price = Price + 1";

MessageBox.Show($"Изменено: {\_database.ExecuteQuery(query)}");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

}

}

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

{

try

{

int id = int.Parse(textBoxId.Text);

string newtitle = textBoxName.Text;

decimal newprice = decimal.Parse(textBoxPrice.Text);

bool success = \_database.UpdateGame(id, newtitle, newprice);

MessageBox.Show(success ? "Запись обновлена!" : "Запись не найдена");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

**4 Вывод**

В ходе лабораторной работы я научилась создавать приложения для чтения данных из БД

**Лабораторная работа №47**

**Создание запросов к БД**

**1 Цель работы**

1.1 Научиться выполнять запросы к БД из клиентского приложения,

1.2 Научиться передавать параметры в запросы.

**2 Контрольные вопросы**

2.1. Для чего используются классы репозитория при работе с БД?

Классы репозитория изолируют работу с данными

2.2. Для чего используются классы сервиса при работе с БД?

Классы сервиса используют репозитории для доступа к данным.

2.3. Для чего используются ORM при работе с БД?

 ORM упрощает работу с БД, кроссплатформенная работа с разными СУБД

2.4. Какие свойства можно указать у параметра?

 Параметры запроса могут иметь имя, тип, значение

**3 Код**

3 задания

DatabaseLayer

В DatabaseFactory.cs

public interface IDbConnectionFactory

{

IDbConnection CreateConnection();

}

public class MsSqlFactory(string connectionString) : IDbConnectionFactory

{

public IDbConnection CreateConnection()

=> new SqlConnection(connectionString);

}

public class SqliteFactory(string connectionString) : IDbConnectionFactory

{

public IDbConnection CreateConnection()

=> new SqliteConnection(connectionString);

}

В Repositories->GamesRepositry.cs

public class GamesRepository(IDbConnectionFactory factory)

{

private readonly IDbConnection \_connection = factory.CreateConnection();

}

В Models->Game.cs

public class Game

{

public int Id { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public int PublicationYear { get; set; }

public decimal Price { get; set; }

}

В Repositories->GamesRepositry.cs

public class GamesRepository(IDbConnectionFactory factory)

{

private readonly IDbConnection \_connection = factory.CreateConnection();

public IEnumerable<Game> GetAll()

=> \_connection.Query<Game>("SELECT \* FROM Game");

public Game? GetById(int id)

=> \_connection.QueryFirstOrDefault<Game>("SELECT \* FROM Game WHERE Id=@id", new { id });

public void Create(Game game)

=> \_connection.Execute("INSERT INTO Game (Title, Description, PublicationYear, Price) VALUES (@Title, @Description, @PublicationYear, @Price)", game);

public void Update(Game game)

=> \_connection.Execute("UPDATE Game SET Title = @Title, Description = @Description, PublicationYear = @PublicationYear, Price = @Price WHERE Id = @Id", game);

public void Delete(int id)

=> \_connection.Execute("DELETE FROM Game WHERE Id=@Id", new {id});

}

WebApp -> GamesController.cs

private readonly GamesRepository \_repository = repository;

// GET: api/<GamesController>

[HttpGet]

public IActionResult Get()

=> Ok(\_repository.GetAll());

// GET api/<ReviewsController>/5

[HttpGet("{id}")]

public IActionResult Get(int id)

=> Ok(\_repository.GetById(id));

// POST api/<GamesController>

[HttpPost]

public IActionResult Post([FromBody] Game game)

{

\_repository.Create(game);

return Created();

}

// PUT api/<GamesController>/5

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Game game)

{

if (id != game.Id)

return NotFound();

\_repository.Update(game);

return NoContent();

}

// DELETE api/<GamesController>/5

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

\_repository.Delete(id);

return NoContent();

}

ConsoleApp -> Program.cs

using DatabaseLayer;

using DatabaseLayer.Models;

using DatabaseLayer.Repositories;

string connectionString = "Data Source=mssql;Initial Catalog=ispp3110;User ID=ispp3110;Password=3110;Trust Server Certificate=True";

IDbConnectionFactory factory = new MsSqlFactory(connectionString);

var repository = new GamesRepository(factory);

var games = repository.GetAll();

var game = repository.GetById(5);

game.Description = "игра не очень";

repository.Update(game);

game.Description = "топ 1 игра";

game.Title = "Бэтмен";

game.PublicationYear = 2005;

game.Price = 1000;

repository.Create(game);

repository.Delete(10);

Console.WriteLine();

В ConsoleClient -> Program.cs

using ApiServices;

var httpClient = new HttpClient()

{

BaseAddress = new Uri("http://localhost:5026/api/")

};

GamesServices service = new(httpClient);

try

{

var games = await service.GetGamesAsync();

var game = await service.GetGameAsync(2);

game.Title = "Человек паук";

game.Description = "супер игра";

game.PublicationYear = 2000;

game.Price = 100;

await service.UpdateGamesAsync(game);

var coolGame = await service.GetGameAsync(5);

coolGame.Description = "игра н1";

coolGame.Title = "ЗОНДБИ ЛАБОРАТОРИЯ";

coolGame.PublicationYear = 2005;

coolGame.Price = 1000;

await service.CreateGameAsync(coolGame);

await service.DeleteGamesAsync(11);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Console.ReadLine();

**4 Вывод**

В ходе лабораторной работы я научилась выполнять запросы к БД из клиентского приложения и передавать параметры в запросы.

**Лабораторная работа №48**

**Создание сервисов**

**1 Цель работы**

1.1 Научиться выполнять запросы к БД из клиентского приложения,

1.2 Научиться передавать параметры в запросы.

**2 Контрольные вопросы**

2.1. Для чего используются классы репозитория при работе с БД?

Репозитории выполняют CRUD-операции

2.2. Для чего используются классы сервиса при работе с БД?

Упрощение тестирования, кэширование и оптимизация запросов

2.3. Для чего используются ORM при работе с БД?

 ORM упрощает работу с БД, кроссплатформенная работа с разными СУБД

2.4. Какие свойства можно указать у параметра?

 Параметры запроса могут иметь имя, тип, значение

**3 Вывод**

В ходе лабораторной работы я научилась выполнять запросы к БД из клиентского приложения и передавать параметры в запросы.

**Практическая работа №1**

**Разработка приложений для обработки файлов**

**1 Цель работы**

1.1 Научиться применять классы для работы с файлами в приложениях на C#.

**2 Контрольные вопросы**

2.1. В чём отличие между классами Directory и DirectoryInfo?

Directory — это статический класс, подходит для быстрого доступа ;

DirectoryInfo — создаётся как объект, подходит для работы с конкретной папкой.

2.2. В чём отличие между классами File и FileInfo?

File — статический, FileInfo — создаётся как объект и даёт работать с его свойствами.

2.3. Как получить список файлов и папок определённого каталога?

Список файлов и папок получается через Directory.GetFiles() и Directory.GetDirectories()

2.4. Какие свойства класса FileInfo позволяют получить информацию о файле?

FileInfo предоставляет свойства: Name, Length, CreationTime, Extension

**3 Код**

Task 1

Console.Write("Введите имя файла: ");

string fileName = Console.ReadLine();

if (File.Exists(fileName))

{

try

{

string fileContent = File.ReadAllText(fileName);

Console.WriteLine("Содержимое файла:");

Console.WriteLine(fileContent);

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при чтении файла: " + ex.Message);

}

}

else

{

Console.WriteLine("Файл не существует.");

}

Task 2

Console.Write("введите имя файла: ");

string fileName = Console.ReadLine();

if (File.Exists(fileName))

{

Console.WriteLine("Файл открыт на дозапись.");

}

else

{

Console.WriteLine("Файл с указанным названием будет создан.");

}

Console.WriteLine("Введите строки для записи в файл (введите 'end' для завершения):");

string input;

StringBuilder stringBuilder = new();

while (true)

{

input = Console.ReadLine();

if (input.Equals("end", StringComparison.OrdinalIgnoreCase))

{

break;

}

stringBuilder.AppendLine(input);

}

File.AppendAllText(fileName, stringBuilder.ToString());

Console.WriteLine($"Запись в файл '{fileName}' завершена.");

Task 3

Console.WriteLine("Введите текст для поиска:");

string searchText = Console.ReadLine();

Console.WriteLine("Введите имя текстового файла:");

string fileName = Console.ReadLine();

if (File.Exists(fileName))

{

string[] lines = File.ReadAllLines(fileName);

bool isFound = false;

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

{

if (lines[i].Contains(searchText, StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine($"Строка {i + 1}: {lines[i]}");

isFound = true;

}

}

if (!isFound)

{

Console.WriteLine("Текст не найден в файле.");

}

}

else

{

Console.WriteLine("Файл не существует.");

}

Task 4

Console.Write("Введите имя папки: ");

string folderPath = Console.ReadLine();

Console.Write("Введите часть имени файла: ");

string filePart = Console.ReadLine();

if (Directory.Exists(folderPath))

{

var files = Directory.GetFiles(folderPath, "\*.\*", SearchOption.AllDirectories);

Console.WriteLine("Найденные файлы:");

foreach (var file in files)

{

if (Path.GetFileName(file).Contains(filePart))

{

FileInfo fileInfo = new FileInfo(file);

Console.WriteLine($"Имя: {fileInfo.FullName}, Размер: {fileInfo.Length} байт");

}

}

}

else

{

Console.WriteLine("Указанная папка не существует.");

}

Task 5

Console.Write("Введите имя папки: ");

string sourceFolderPath = Console.ReadLine();

if (Directory.Exists(sourceFolderPath))

{

var files = Directory.GetFiles(sourceFolderPath);

foreach (var file in files)

{

FileInfo fileInfo = new FileInfo(file);

DateTime lastModified = fileInfo.LastWriteTime;

string targetFolderPath = Path.Combine(sourceFolderPath, lastModified.Year.ToString(),

lastModified.Month.ToString("D2"),

lastModified.Day.ToString("D2"));

if (!Directory.Exists(targetFolderPath))

{

Directory.CreateDirectory(targetFolderPath);

}

string targetFilePath = Path.Combine(targetFolderPath, fileInfo.Name);

File.Move(file, targetFilePath);

Console.WriteLine($"Файл {fileInfo.Name} перемещен в {targetFilePath}");

}

}

else

{

Console.WriteLine("Указанная папка не существует.");

}

**4 Вывод**

В ходе практической работы я научилась применять классы для работы с файлами в приложениях на C#

**Практическая работа №2**

**Работа с табличными файлами**

**1 Цель работы**

1.1 Научиться выполнять создание и редактирование табличных документовна C#.

**2 Контрольные вопросы**

2.1. Какое пространство имён требуется подключить для работы с Excel?

Пространство имён: Microsoft.Office.Interop.Excel

2.2. Как создать объект типа «приложение Excel» в программе на C#?

 Объект Excel: var excel = new Excel.Application();

2.3. Что такое Workbooks?

Workbooks — коллекция открытых Excel-файлов

2.4. Что такое Worksheets?

Worksheets — коллекция листов в книге

2.5. Что такое Range?

Range — диапазон ячеек

2.6. Что такое Cells?

Cells — доступ к ячейкам по индексам по строкам и столбцам.

2.7. Как получить доступ к значению ячейки и диапазона?

Значение ячейки получается через worksheet.Cells[1,1].Value

**3 Вывод**

В ходе практической работы я научилась выполнять создание и редактирование табличных документовна C#

**Практическая работа №3**

**Работа с текстовыми файлами**

**1 Цель работы**

1.1 Научиться выполнять создание и редактирование текстовых документов на C#.

**2 Контрольные вопросы**

2.1. Какое пространство имён требуется подключить для работы с Word?

Пространство имён: Microsoft.Office.Interop.Word

2.2. Как создать объект типа «приложение Word» в программе на C#?

Объект World: var word = new Word.Application();

2.3. Что такое Documents?

Documents — коллекция открытых документов

2.4. Что такое Range?

Range — участок текста

2.5. Что такое Selection?

Selection — выделенный текст пользователем или программой

2.6. Что такое Paragraphs?

Paragraphs — коллекция абзацев

2.7. Что такое Tables?

Tables — коллекция таблиц

**3 Вывод**

В ходе практической работы я научилась выполнять создание и редактирование текстовых документов на C#

**Практическая работа №4**

**Привязка данных**

**1 Цель работы**

1.1 Научиться выполнять привязку данных в приложении WPF.

**2 Контрольные вопросы**

2.1. Что такое «привязка данных»?

Привязка данных связывает элементы управления с источниками данных

2.2. Каков шаблон настройки привязки данных?

Указание пути к свойству и источнику

2.3. Какой интерфейс надо реализовать для создания конвертера значений?

 Для создания конвертера значений реализуется IValueConverter.

2.4. Какой интерфейс надо реализовать для валидации данных при привязке?

 Для валидации данных реализуется IDataErrorInfo или INotifyDataErrorInfo

**3 Вывод**

Входе практической работы я научилась выполнять привязку данных в приложении WPF

**Практическая работа №5**

**Сохранение настроек приложения**

**1 Цель работы**

1.1 Научиться сохранять настройки в клиентском приложении на C#.

**2 Контрольные вопросы**

2.1 Как добавить настройки в приложение на C#?

Настройки добавляются в файл конфигурации

2.2 Как программно считать значение параметра из настроек?

var mySetting = Properties.Settings.Default.MySetting;

2.3 Как программно изменить значение параметра в настройках?

Properties.Settings.Default.MySetting = newValue;

2.4 Как выполнить сохранение значений параметров в настройках?

Properties.Settings.Default.Save();

**3 Код**

Task 1

В AuthWindow.xaml

<StackPanel>

<Label Content="Логин:"/>

<TextBox x:Name="LoginTextBox"/>

<Label Content="Пароль:"/>

<PasswordBox x:Name="PasswordTextBox"/>

<Button x:Name="AuthButton" Content="Войти"

Click="AuthButton\_Click"

Width="100" Height="30"/>

</StackPanel>

В AuthWindow.xaml.cs

private string HashPassword(string password)

{

byte[] bytes = Encoding.UTF8.GetBytes(password);

byte[] hashBytes = SHA256.HashData(bytes);

return Convert.ToHexString(hashBytes);

}

private void AuthButton\_Click(object sender, RoutedEventArgs e)

{

string login = LoginTextBox.Text;

string password = HashPassword(PasswordTextBox.Password);

if (login == Properties.Settings.Default.Login &&

password == Properties.Settings.Default.Password)

{

SettingsWindow settingsWindow = new();

settingsWindow.Owner = this.Owner;

settingsWindow.Show();

this.Close();

}

else

{

MessageBox.Show("Данные введены неверно.", "Попробуй ещё раз.",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

В MainWindow.xaml

<StackPanel>

<Button Content="Открыть авторизацию"

Click="OpenAuth\_Click"

Width="150" Height="30"/>

</StackPanel>

В MainWindow.xaml.cs

private void OpenAuth\_Click(object sender, RoutedEventArgs e)

{

AuthWindow authWindow = new();

DataContext = this;

authWindow.ShowDialog();

}

В SettingsWindow.xaml

<StackPanel>

<Label Content="Логин:"/>

<TextBox x:Name="LoginTextBox"/>

<Label Content="Пароль:"/>

<PasswordBox x:Name="PasswordTextBox"/>

<Label Content="Email:"/>

<TextBox x:Name="EmailTextBox"/>

<Button x:Name="SaveButton" Content="Сохранить"

Click="SaveButton\_Click"

Width="100" Height="30"/>

</StackPanel>

В SettingsWindow.xaml.cs

public SettingsWindow()

{

InitializeComponent();

LoginTextBox.Text = Properties.Settings.Default.Login;

//PasswordTextBox.Password = Properties.Settings.Default.Password;

EmailTextBox.Text = Properties.Settings.Default.Email;

}

private string HashPassword(string password)

{

byte[] bytes = Encoding.UTF8.GetBytes(password);

byte[] hashBytes = SHA256.HashData(bytes);

return Convert.ToHexString(hashBytes);

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

Properties.Settings.Default.Login = LoginTextBox.Text;

if (PasswordTextBox.Password.Length > 0)

Properties.Settings.Default.Password = HashPassword(PasswordTextBox.Password);

Properties.Settings.Default.Email = EmailTextBox.Text;

Properties.Settings.Default.Save();

MessageBox.Show("Мы сохранили ваши настройки.", "Но не спешите радоваться...",

MessageBoxButton.OK, MessageBoxImage.Information);

MainWindow mainWindow = new();

mainWindow.Show();

Close();

}

Task 2

В AuthWindow.xaml.cs

private void AuthButton\_Click(object sender, RoutedEventArgs e)

{

if (LoginTextBox.Text == ConfigurationManager.AppSettings["Login"] &&

PasswordTextBox.Password == ConfigurationManager.AppSettings["Password"])

{

SettingsWindow settingsWindow = new();

settingsWindow.Show();

}

else

{

MessageBox.Show("Данные введены неверно.", "Попробуй ещё раз.",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

В MainWindow.xaml.cs

private void AuthButton\_Click(object sender, RoutedEventArgs e)

{

AuthWindow authWindow = new();

DataContext = this;

authWindow.ShowDialog();

}

В SettingsWindow.xaml.cs

public SettingsWindow()

{

InitializeComponent();

LoginTextBox.Text = ConfigurationManager.AppSettings["Login"];

PasswordTextBox.Password = ConfigurationManager.AppSettings["Password"];

EmailTextBox.Text = ConfigurationManager.AppSettings["Email"];

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);

config.AppSettings.Settings["Login"].Value = LoginTextBox.Text;

config.AppSettings.Settings["Password"].Value = PasswordTextBox.Password;

config.AppSettings.Settings["Email"].Value = EmailTextBox.Text;

MessageBox.Show("Мы сохранили ваши настройки.", "Но не спешите радоваться...",

MessageBoxButton.OK, MessageBoxImage.Information);

config.Save();

config.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection("appSettings");

MainWindow mainWindow = new();

mainWindow.Show();

Close();

}

App.config

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

<configuration>

<appSettings>

<add key="Login" value="admin"/>

<add key="Email" value="admin@gmail.com"/>

<add key="Password" value="qwerty"/>

</appSettings>

</configuration>

Task 3

В AuthWindow.xaml.cs

private void AuthButton\_Click(object sender, RoutedEventArgs e)

{

string[] strings = System.IO.File.ReadAllLines("userData.txt");

string login = strings[0];

string password = strings[1];

if (LoginTextBox.Text == login && PasswordTextBox.Password == password )

{

SettingsWindow window = new();

window.Show();

}

else

{

MessageBox.Show("Данные введены неверно.", "Попробуй ещё раз.",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

В MainWindow.xaml.cs

private void AuthButton\_Click(object sender, RoutedEventArgs e)

{

AuthWindow authWindow = new();

DataContext = this;

authWindow.ShowDialog();

}

В SettingsWindow.xaml.cs

public SettingsWindow()

{

InitializeComponent();

string[] strings = System.IO.File.ReadAllLines("userData.txt");

LoginTextBox.Text = strings[0];

PasswordTextBox.Password = strings[1];

EmailTextBox.Text = strings[2];

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

string[] strings = [LoginTextBox.Text, PasswordTextBox.Password, EmailTextBox.Text];

System.IO.File.WriteAllLines("userData.txt", strings);

MessageBox.Show("Мы сохранили ваши настройки.", "Но не спешите радоваться...",

MessageBoxButton.OK, MessageBoxImage.Information);

MainWindow mainWindow = new();

mainWindow.Show();

Close();

}

**4 Вывод**

В ходе практической работы я научилась сохранять настройки в клиентском приложении на C#

**Практическая работа №6**

**Создание пользовательских элементов управления**

**1 Цель работы**

1.1 Научиться создавать пользовательские элементы управления в приложении WPF.

**2 Контрольные вопросы**

2.1. Для чего применяется ControlTemplate?

ControlTemplate изменяет внешний вид элемента.

2.2. Где может быть описан ControlTemplate?

ControlTemplate может быть описан в ресурсах

2.3. Для чего применяется UserControl?

UserControl создаётся для повторного использования компонентов

2.4. Каков алгоритм создания пользовательского элемента управления?

Создаём UserControl, добавляем элементы, логику и подключаем к проекту.

2.5. Как программно создать обработчик события?

элемент.Событие += Метод

2.6. Для чего используется свойство зависимости?

 Свойство зависимости используется для привязки данных

**3 Код**

4 задания

В MainWindow.xaml

<Window.Resources>

<ControlTemplate x:Key="RadioButtonTemplate" TargetType="{x:Type RadioButton}">

<Border x:Name="border" CornerRadius="10" Background="{TemplateBinding Background}"

BorderThickness="1" BorderBrush="Gray" Padding="10,5">

<ContentPresenter x:Name="content"

Content="{TemplateBinding Content}"

HorizontalAlignment="Center"

VerticalAlignment="Center"/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsChecked" Value="True">

<Setter TargetName="border" Property="Background" Value="LightBlue"/>

</Trigger>

<Trigger Property="IsEnabled" Value="False">

<Setter TargetName="border" Property="Background" Value="LightGray"/>

<Setter TargetName="content" Property="TextElement.Foreground" Value="Gray"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Window.Resources>

<StackPanel Width="200" Background="#EEE" Margin="1,0,599,0">

<RadioButton x:Name="MenuItem1" Content="Главная" Template="{StaticResource RadioButtonTemplate}"

Background="White" GroupName="Menu" Checked="RadioButton\_Checked" IsEnabled="False"/>

<RadioButton x:Name="MenuItem2" Content="Продукты" Template="{StaticResource RadioButtonTemplate}"

Background="White" GroupName="Menu" Checked="RadioButton\_Checked"/>

<RadioButton x:Name="MenuItem3" Content="Услуги" Template="{StaticResource RadioButtonTemplate}"

Background="White" GroupName="Menu" Checked="RadioButton\_Checked"/>

<RadioButton x:Name="MenuItem4" Content="О нас" Template="{StaticResource RadioButtonTemplate}"

Background="White" GroupName="Menu" Checked="RadioButton\_Checked"/>

<RadioButton x:Name="MenuItem5" Content="Контакты" Template="{StaticResource RadioButtonTemplate}"

Background="White" GroupName="Menu" Checked="RadioButton\_Checked"/>

<local:TimeControl/>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Цена:" VerticalAlignment="Center" Width="60"/>

<TextBox x:Name="PriceTextBox" Width="100" Text="100"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Количество:" VerticalAlignment="Center"/>

<local:NumericUpDownControl x:Name="NumericControl" MaxValue="5"

ValueChanged="NumericControl\_ValueChanged"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Итого:" VerticalAlignment="Center" Width="60"/>

<TextBlock x:Name="TotalTextBlock" VerticalAlignment="Center" Width="100"/>

</StackPanel>

</StackPanel>

В MainWindow.xaml.cs

public MainWindow()

{

InitializeComponent();

NumericControl.MaxValue = 5;

PriceTextBox.TextChanged += (s, e) => CalculateTotal();

CalculateTotal();

}

private void RadioButton\_Checked(object sender, RoutedEventArgs e)

{

if (sender is RadioButton radioButton)

{

this.Title = radioButton.Content.ToString();

}

}

private void NumericControl\_ValueChanged(object sender, RoutedEventArgs e)

{

CalculateTotal();

}

private void CalculateTotal()

{

if (decimal.TryParse(PriceTextBox.Text, out decimal price))

{

decimal total = price \* NumericControl.Value;

TotalTextBlock.Text = total.ToString("C2");

}

else

{

TotalTextBlock.Text = "Не бывает такой цены";

}

}

В NumericUpDownControl.xaml

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*" MinWidth="30"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Button x:Name="MinusButton" Grid.Column="0" Content="-" Width="30"/>

<TextBox x:Name="ValueTextBox" Grid.Column="1" Text="0" TextAlignment="Center"/>

<Button x:Name="PlusButton" Grid.Column="2" Content="+" Width="30"/>

</Grid>

В NumericUpDownControl.xaml.cs

public partial class NumericUpDownControl : UserControl

{

public static readonly DependencyProperty MaxValueProperty =

DependencyProperty.Register("MaxValue", typeof(int), typeof(NumericUpDownControl),

new PropertyMetadata(int.MaxValue, OnMaxValueChanged));

public static readonly DependencyProperty ValueProperty =

DependencyProperty.Register("Value", typeof(int), typeof(NumericUpDownControl),

new PropertyMetadata(0, OnValueChanged, CoerceValue));

public static readonly RoutedEvent ValueChangedEvent =

EventManager.RegisterRoutedEvent("ValueChanged", RoutingStrategy.Bubble,

typeof(RoutedEventHandler), typeof(NumericUpDownControl));

public NumericUpDownControl()

{

InitializeComponent();

PlusButton.Click += (s, e) => Value++;

MinusButton.Click += (s, e) => Value--;

ValueTextBox.TextChanged += ValueTextBox\_TextChanged;

UpdateButtonsState();

}

public int MaxValue

{

get => (int)GetValue(MaxValueProperty);

set => SetValue(MaxValueProperty, value);

}

public int Value

{

get => (int)GetValue(ValueProperty);

set => SetValue(ValueProperty, value);

}

public event RoutedEventHandler ValueChanged

{

add => AddHandler(ValueChangedEvent, value);

remove => RemoveHandler(ValueChangedEvent, value);

}

private static void OnMaxValueChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

{

var control = (NumericUpDownControl)d;

control.CoerceValue(ValueProperty);

control.UpdateButtonsState();

}

private static void OnValueChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

{

var control = (NumericUpDownControl)d;

control.ValueTextBox.Text = control.Value.ToString();

control.UpdateButtonsState();

control.RaiseEvent(new RoutedEventArgs(ValueChangedEvent));

}

private static object CoerceValue(DependencyObject d, object baseValue)

{

var control = (NumericUpDownControl)d;

int value = (int)baseValue;

if (value < 0)

return 0;

if (value > control.MaxValue)

return control.MaxValue;

return value;

}

private void ValueTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

if (int.TryParse(ValueTextBox.Text, out int newValue))

{

Value = newValue;

}

else

{

ValueTextBox.Text = Value.ToString();

}

}

private void UpdateButtonsState()

{

MinusButton.IsEnabled = Value > 0;

PlusButton.IsEnabled = Value < MaxValue;

}

}

В TimeControl.xaml

<UserControl x:Class="PractWork6.TimeControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<Label x:Name="TimeLabel" Content="00:00:00" FontSize="16"/>

</UserControl>

В TimeControl.xaml.cs

private DispatcherTimer timer = new DispatcherTimer(DispatcherPriority.Render);

public TimeControl()

{

InitializeComponent();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += Timer\_Tick;

timer.Start();

}

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

{

TimeLabel.Content = DateTime.Now.ToString("HH:mm:ss");

}

**4 Вывод**

В ходе практической работы я научилась создавать пользовательские элементы управления в приложении WPF