**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**по МДК.01.01**

Студент: Матигоров Н.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 упорядоченная коллекция однотипных элементов, каждый из которых доступен по своему уникальному индексу.

2.2 Как описывается одномерный массив?

тип\_массива[] название\_массива;

2.3 Как обратиться к некоторому элементу одномерного массива?

название\_массива[индекс элемента]

2.4 Как можно задать одномерный массив?

тип\_массива[] название\_массива = new тип\_массива[размер\_массива];

2.5 Что такое «сортировка»?

Сортировка в программировании — это упорядочивание элементов массива или списка в определённом порядке, обычно по возрастанию или по убыванию.

2.6 Что такое «алгоритм сортировки»?

Алгоритм сортировки — это последовательность операций, которая упорядочивает элементы в массиве или списке по определённому признаку.

2.7 Какие виды сортировки массивов существуют?

Простые, продвинутые и гибридные.

**3 Вывод**

3.1 Я научился 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 операций, которая упорядочивает элементы в массиве или списке по определённому признаку.

2.2 Какие виды поиска элементов массивов существуют?

Линейный, двоичный, поиск прыжками, алгоритм Фибоначчиевого поиска и др.

2.3 В чем особенность алгоритма линейного поиска и какова его временная

сложность?

Особенность алгоритма линейного поиска (последовательного поиска) — последовательный просмотр каждого элемента списка до тех пор, пока не будет найден искомый элемент или не будет достигнут конец списка. Временная сложность O(n).

2.4 В чем особенность алгоритма двоичного поиска и какова его временная

сложность?

Особенность алгоритма двоичного поиска в том, что он делит заранее отсортированный массив данных пополам, чтобы обнаружить нужный элемент. Временная сложность O(log(N)).

2.5 В чем особенность алгоритма поиска прыжками и какова его временная

сложность?

Особенность алгоритма поиска прыжками — похож на двоичный поиск, но движение только вперёд. Такой поиск требует отсортированной коллекции Временная сложность O(sqrt(N)).

**3 Вывод**

3.1 Я научился 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 саму себя для решения задачи.

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

При реализации рекурсивных алгоритмов на электронных вычислительных машинах могут возникать проблемы, связанные с переполнением стека, неэффективностью и сложностью отладки.

2.3 Какое определение функции может быть названо рекурсивным? Привести

примеры.

Рекурсивной может быть названа функция, которая в процессе выполнения вызывает саму себя. К примеру, функция вычисления факториала.

2.4 Что такое «глубина рекурсии»?

Глубина рекурсии — это максимальное количество уровней рекурсивных вызовов, которые функция может выполнить до достижения базового случая или предела, установленного интерпретатором языка программирования.

2.5 Что такое «рекурсивный спуск»?

Рекурсивный спуск в программировании — это метод синтаксического анализа, при котором программа представляет себя в виде набора подпрограмм, каждая из которых читает и обрабатывает свою конструкцию, а также вызывает (в том числе рекурсивно) соответствующие подпрограммы для анализа вложенных конструкций.

2.6 Что такое «рекурсивный подъём»?

Это завершение работы рекурсивных подпрограмм, вплоть до самой первой, инициировавшей рекурсивные вызовы.

**3 Вывод**

3.1 Я смог научиться 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 — совокупность приёмов и методов, которые облегчают и упрощают решение познавательных, конструктивных и практических задач..

2.2 Зачем используются эвристические алгоритмы?

Эвристические алгоритмы используются, чтобы ускорить решение задачи в тех случаях, когда точное решение не может быть найдено.

2.3 Какие алгоритмы относятся к эвристическим?

Алгоритм поиска локального максимума, жадный алгоритм, метод ветвей и границ

**3 Вывод**

Я начал реализовывать, 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#?

Class имя\_класса

{

// тело класса

}.

2.2 На какие виды делятся данные класса в C#?

Поля, константы, события.

2.3 На какие виды делятся функции класса в C#?

Статические и экземплярные(зависящие от свойств определённого экземпляра).

2.4 Для чего применяются конструкторы классов?

Конструкторы классов применяются для инициализации объектов.

2.5 Что такое «цепочка конструкторов»?

Цепочка конструкторов — это механизм, с помощью которого конструктор класса вызывает другой конструктор того же класса.

2.6 Для чего применяются свойства классов?

Свойства классов (акссесоры) используются для доступа к внутреннему состояния объекта.

2.7 Когда используются автоматически реализуемые свойства классов?

В классах или структурах с модификатором record.

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

Член экземпляра класса относится к конкретному экземпляру класса. Статический член класса, напротив, относится к классу в целом, а не к конкретному экземпляру класса.

**3 Вывод**

3.1 Был изучен процесс разработки и применения классов на C#

3.2 Был изучен процесс 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»?

Это один из мощнейших механизмов языка, который позволяет определить несколько методов с одним и тем же именем, но с различными параметрами.

2.2 К какому виду полиморфизма относится перегрузка методов?

Статический.

2.3 Какие типы функций класса можно перегружать?

Любые, кроме некоторых операторов.

2.4 Какие требования предъявляются к сигнатуре перегружаемых функций класса?

Различный набор параметров.

2.5 Какие типы функций класса нельзя перегружать?

Некоторые оперторы: ^x, x = y, x.y, x?.y, c ? t : f, x ?? y, x ??= y, x..y, x->y, =>, f(x), as, await, checked, unchecked, default, delegate, is, nameof, new, sizeof, stackalloc, switch, typeof, with.

**3 Вывод**

3.1 Я изучил процесс выполнения перегрузки методов на языке C#;

3.2 Были изучены различные способы передачи параметров в метод.

**Лабораторная работа №7**

**Определение операций в классе**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Какое ключевое слово определяет операторный метод?

Operator.

2.2 Какие унарные операторы можно перегружать в C#?

+, -, !, ~, ++, --, true, false.

2.3 Какие бинарные операторы можно перегружать в C#?

x + y, x - y, x \* y, x / y, x % y, x & y, x | y, x ^ y, x << y, x >> y.

2.4 Какие операторы сравнения можно перегружать в C#?

x == y, x != y, x < y, x > y, x <= y, x >= y.

2.5 Какие операторы требуется перегружать в C# попарно?

x + y, x - y, x \* y, x / y, x % y, x & y, x | y, x ^ y, x << y, x >> y, x == y, x != y, x < y, x > y, x <= y, x >= y.

**3 Вывод**

3.1 Я изучил 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 принцип объектно-ориентированного программирования, который позволяет создавать иерархии классов, где один класс (подкласс) наследует свойства и методы другого класса.

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

Один.

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

base.

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

Переопределение метода в объектно-ориентированном программировании — это возможность заменить реализацию метода, уже определённого в родительском классе, на специфическую реализацию в классе-наследнике..

2.5 Что такое «абстрактный класс»?

Класс, который не может иметь своего экземпляра.

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

Virtual позволяет переопределять метод в классе наследнике.

Override используется для переопределение метода.

Abstract указывает на неполную или отсутствующую реализацию.

New явным образом скрывает члены, унаследованные от базового класса.

**3 Вывод**

1.1 Я изучил процесс разработки дочерних классов на языке C#;

1.2 Была изучена реализация 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»?

Контракт, который должен быть реализован в класс, исполняющий его, или может использоваться как тип данных.

2.2 Чем отличается интерфейс от абстрактного класса?

Интерфейс говорит какие должны быть методы и может дать базовую реализацию, а абстрактный класс может иметь поля и реализованные методы, но абстрактные методы, в классах наследниках обязаны иметь реализацию (чтобы класс наследник можно было создавать экземпляры класса).

2.3 Есть ли у класса ограничения по количеству реализуемых интерфейсов?

Нет.

2.4 Какова общая форма объявления интерфейсов и их элементов?

Interface имя\_интерфейса

{

тип\_возврата имя\_метода(параметры);

}

**3 Вывод**

3.1 Был изучен процесс разработки и реализации интерфейсов на языке C#.

**4 Ход работы**

4.4 Создание объектов типа интерфейса.

Создать переменную типа интерфейса IPrinter и затем:

- присвоить ей значение объекта из таблицы 1. Вызвать у переменной типа интерфейса метод Print();

- присвоить ей значение объекта из таблицы 2. Вызвать у переменной типа интерфейса метод Print().

Создать переменную типа IPrinter[], добавить в массив 4 объекта (2 из таблицы 1 и 2 из таблицы 2). Используя цикл:

- перебрать элементы массива и у каждого вызвать метод Print();

- если элемент поддерживает тип IFigure, вывести на экран название фигуры.

// Задание 4

Console.WriteLine("\nЗадание 4");

IPrinter applicant1 = new Applicant()

{

FullName = "Михалков Иван Иванович",

ClassNumber = 9,

AverageScore = 4.50

};

IPrinter rectangle1 = new Rectangle()

{

Height = 10,

Width = 2

};

applicant1.Print();

rectangle1.Print();

Console.WriteLine();

IPrinter[] printers = [

applicant1,

new Applicant()

{

FullName = "Тестов Тест Тесторович",

ClassNumber = 11,

AverageScore = 4.77

},

rectangle1,

new Rectangle()

{

Height = 8,

Width = 5

}];

foreach (var printer in printers)

{

if (printer is IFigure iFigure)

Console.WriteLine(iFigure.Name);

printer.Print();

**}**

Вывод:

Задание 4

ФИО: Михалков Иван Иванович; Номер класса: 9; Средний балл: 4,5

Высота: 10; Ширина: 2

ФИО: Михалков Иван Иванович; Номер класса: 9; Средний балл: 4,5

ФИО: Тестов Тест Тесторович; Номер класса: 11; Средний балл: 4,77

Прямоугольник

Высота: 10; Ширина: 2

Прямоугольник

Высота: 8; Ширина: 5

**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?

Используется для сортировки объектов данного типа.

2.2 Для чего используется интерфейс IEquatable?

Используется для определения равенства двух объектов.

2.3 В чем отличие между обобщенным и необобщенным интерфейсами?

Основное отличие между обобщённым и необобщённым интерфейсами заключается в том, что обобщённые интерфейсы работают с различными типами данных, а необобщённые — только с конкретными типами.

**3 Вывод**

3.1 Был изучен процесс реализации стандартных интерфейсов на языке C#.

**4 Ход работы**

4.5 Реализация стандартного интерфейса IComparer<T> (c обобщениями)

Реализовать интерфейс IComparer в классе DescendingComparer для того, чтобы сортировать по убыванию числового значения.

Реализовать интерфейс IComparer в классе StringComparer для того, чтобы сортировать по возрастанию первого строкового значения.

Для проверки в Program.Main создать массив объектов класса и отсортировать этот массив, вызвав Array.Sort(массив, new ИмяComparer()). Если все сделано корректно, массив будет отсортирован по выбранному компаратору. На экран должен быть выведен исходный и отсортированный массив.

public class DescendingComparer : IComparer<Applicant>

{

public int Compare(Applicant? applicant1, Applicant? applicant2)

=> -applicant1.CompareTo(applicant2);

}

public class StringComparer : IComparer<Applicant>

{

public int Compare(Applicant? applicant1, Applicant? applicant2)

=> applicant1.FullName.CompareTo(applicant2?.FullName);

}

// Задание 5

foreach (var applicant in applicants)

if (applicant is not null)

Console.WriteLine($"{applicant.FullName} закончил {applicant.ClassNumber} класс со средним баллом {applicant.AverageScore}.");

else

Console.WriteLine("null");

Console.WriteLine();

Array.Sort(applicants, new DescendingComparer());

foreach (var applicant in applicants)

if (applicant is not null)

Console.WriteLine($"{applicant.FullName} закончил {applicant.ClassNumber} класс со средним баллом {applicant.AverageScore}.");

else

Console.WriteLine("null");

Console.WriteLine();

Array.Sort(applicants,new StringComparer());

foreach (var applicant in applicants)

if (applicant is not null)

Console.WriteLine($"{applicant.FullName} закончил {applicant.ClassNumber} класс со средним баллом {applicant.AverageScore}.");

else

Console.WriteLine("null");

**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 в C# — это специализированный класс, который позволяет работать с группами объектов, выполнять операции над ними, такие как хранение, обновление, удаление, извлечение, поиск и сортировка. .

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

List<T>, Dictionary<TKey, Tvalue>, LinkedList<T>, Queue<T>, SortedSet<T>, SortedList<TKey, Tvalue>, SortedDictionary<TKey, Tvalue>, Stack<T>.

2.3 Что такое List?

Класс, представляющий последовательный список.

2.4 Как можно обратиться к элементу списка?

имя\_списка[индекс\_элемента].

2.5 Что такое Dictionary<TKey, TValue>?

Класс коллекции, хранящей наборы пар «ключ-значение».

2.6 Как можно обратиться к элементу словаря?

имя\_словаря[ключ].

**3 Вывод**

3.1 Я изучил 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?

В стеке.

2.2 Что такое «структура»?

Структура — это объект, который храниться в стеке.

2.3 Чем отличается структура от класса?

Принципами ООП, (зачастую структуры имеют противоположные свойства принципам ООП), а также тем, что структуры передают значение, а классы - ссылку.

2.4 Что такое «перечисление»?

Тип данных, который содержит набор констант, и каждой константе сопоставлено определённое числовое значение.

2.5 Для чего используются перечисления?

Перечисления используются для того, чтобы делать код более читаемым и безопасным. Вместо магических чисел или строк используются ясные и понятные имена.

2.6 Какова общая форма объявления перечисления в C#?

enum имя\_перечисления

{

имя\_константы1,

. имя\_константы2,

…

}

**3 Вывод**

3.1 Я изучил 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?

Это ссылочный тип данных.

2.2 Какие операции допустимы над строковыми данными?

Конкатенация, сравнение, поиск, замена, разделение, обрезка, извлечение подстрок, вставка.

2.3 В чем отличие переменной типа string от массива символов?

Массив изменяем, а строка - нет.

2.4 Что такое «интерполяция строк»?

Подстановка переменных в строку

2.5 Изменяют ли методы класса String исходную строку?

Нет, методы создают новую строку.

**3 Вывод**

3.1 Был изучен процесс обработки строк на языке C#;

3.2 Я научился 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»?

Регуля́рные выраже́ния — формальный язык, используемый в компьютерных программах, работающих с текстом, для поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов (символов-джокеров).

2.2 Для чего используются регулярные выражения?

Для работы со текстом.

2.3 Для чего используется класс Regex?

Класс Regex используется для работы с регулярными выражениями.

2.4 Каков алгоритм поиска подстроки при помощи регулярного выражения?

Алгоритм поиска подстроки с помощью регулярного выражения в C# заключается в сопоставлении шаблона (регулярного выражения) с строкой.

2.5 Каков алгоритм замены подстроки при помощи регулярного выражения?

Алгоритм замены подстроки с помощью регулярного выражения в C# заключается в том, что метод Regex.Replace() класса System.Text.RegularExpressions заменяет все строки, соответствующие указанному регулярному выражению, на указанную строку замены.

2.6 Для чего в регулярных выражениях применяются escape-символы?

Escape-символы в регулярных выражениях применяются, чтобы сделать специальные символы обычными.

2.7 Для чего в регулярных выражениях применяются классы символов?

Классы символов в регулярных выражениях применяются для упрощения поиска определённых типов символов в тексте.

2.8 Для чего в регулярных выражениях применяются квантификаторы?

Квантификаторы в регулярных выражениях применяются для указания количества повторений символа, группы или класса символов в строке.

**3 Вывод**

3.1 Я научился составлять шаблоны регулярных выражений в программах на C#;

3.2 Я научился применять регулярные выражения для поиска и замены подстрок в программах на языке C#.

**4 Ход работы**

4.1 Написать регулярное выражение, проверяющее, является ли введенная строка корректным номером телефона.

Требования к формату номера: +7(9хх)ххх-хх-хх или 8(9хх)ххх-хх-хх, где х — любая цифра.

// Задание 1

using System.Text.RegularExpressions;

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

Console.WriteLine("Введите номер телефона, чтобы соответствовал формату +7(9хх)ххх-хх-хх или 8(9хх)ххх-хх-хх, где х — любая цифра:");

string numberPhone = Console.ReadLine();

if (Regex.IsMatch(numberPhone, pattern))

Console.WriteLine("Введёный номер телефона соответствует формату.");

else

Console.WriteLine("Введёный номен телефона НЕ соответствует формату.");

4.2 Написать регулярное выражение, заменяющее в исходной строке два и более подряд идущих пробельных символа на пробел.

// Задание 2

using System.Text.RegularExpressions;

Console.Write("Введите строку: ");

string inputString = Console.ReadLine();

Regex regex = new(@"\s+");

string target = " ";

Console.WriteLine(regex.Replace(inputString, target));

4.3 Написать регулярное выражение, проверяющее, является ли введенная строка корректным e-mail адресом.

Требования к адресу: логин@доменноеИмя.

Логин может состоять из символов латинского алфавита, цифр, тире, нижнего подчеркивания.

Доменное имя должно состоять из двух или более доменов, причем домен первого уровня может состоять только из букв (например: ru, com, org и т.д.), остальные — из букв и цифр, домены разделяются точкой.

Поиск должен быть регистронезависимым.

// Задание 3

using System.Text.RegularExpressions;

Regex regex = new(@"^[a-z1-9\_**\-\.**]+**\@**([a-z0-9]+**\.**){0,2}[a-z]+", RegexOptions.IgnoreCase);

Console.Write("Введите адрес электронной почты: ");

string input = Console.ReadLine();

if (regex.IsMatch(input))

Console.WriteLine("Введён верный адрес электронной почты.");

else

Console.WriteLine("Введён неверный адрес электронной почты.");

4.4 Написать регулярное выражение, проверяющее, является ли введенная строка надежным паролем.

Требования к паролю: длина от 6 символов, обязательна цифра, строчная и прописная латинская буква, знак препинания (.?!).

Если требования не выполнены, запрашивать повторный ввод, пока строка не будет соответствовать требованиям.

// Задание 4

using System.Text.RegularExpressions;

Regex regex = new(@"^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[.?!]).{6,}$");

while(true)

{

Console.Write("Введите пароль: ");

string input = Console.ReadLine();

if (regex.IsMatch(input))

break;

else

Console.WriteLine("Введён ненадёжный пароль.\n");

}

Console.WriteLine("Введён надёжный пароль.");

3.5 Написать регулярное выражение, заменяющее в исходной строке все даты в формате дд.мм.гггг и дд/мм/гггг на даты в формате гггг-мм-дд.

В качестве дня (дд) и месяца (мм) могут быть указаны от 1 до 2 цифр, в качестве года (гггг) — 2 или 4 цифры. Месяц может быть числом от 01 до 12, день — от 01 до 31, год — любое число.

При реализации использовать именованные группы для дня, месяца и года.

// Задание 5

using System.Text.RegularExpressions;

Regex regex = new(@"^\d{2}[./]\d{2}[./]\d{4}$");

Console.Write("Введите дату: ");

string input = Console.ReadLine();

if (!regex.IsMatch(input))

{

Console.WriteLine("Некорректный формат ввода даты.");

return;

}

Regex pattern = new(@"^(?<day>\d{2})[./](?<month>\d{2})[./](?<year>\d{4})$");

string formatedDate = pattern.Replace(input, "${year}-${month}-${day}");

Console.WriteLine(formatedDate);

**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 данных, который представляет собой ссылку на метод с заданным списком параметров и типом возвращаемого значения.

2.2 Как объявить делегат на C#?

delegate возвращаемый\_тип имя\_делегата (параметры).

2.3 Какие способы вызова метода через делегат существуют?

имя\_делегата(); имя\_делегата.Invoke(); имя\_делегата?.Invoke();.

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

Action, Predicate, Func.

**3 Вывод**

3.1 Я научился 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#?

Это сущность, которая позволяет классу уведомлять другие объекты о возникновении каких-либо ситуаций, а также реагировать на эти уведомления.

2.2 Как объявить событие на C#?

event тип\_делегата имя\_события;

2.3 Как создать обработчик события?

имя\_события += (нажать Tab).

2.4 Какой класс является родительским для всех классов данных события?

EventArgs.

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

Action, Predicate, Func.

**3 Вывод**

3.1 Я научился 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, которая позволяет выполнять запросы к различным источникам данных, таким как коллекции, базы данных, XML-документы, и другим, используя единый синтаксис.

2.2 Что такое «анонимный тип»?

Анонимный тип — это тип данных, который создается и инициализируется без объявления явного класса. Он позволяет создавать временные объекты с набором свойств, но без определения типа на этапе компиляции.

2.3 Для чего используется var?

var используется для неявной типизации локальных переменных. Оно указывает компилятору, что тип переменной должен быть выведен из выражения, находящегося с правой стороны оператора инициализации.

2.4 Для чего используется метод Select?

Используется для преобразования элементов последовательности в элементы другого типа.

2.5 Для чего используются методы OrderBy, OrderByDescending?

Используются для сортировки элементов коллекции на основе заданных критериев.

2.6 Для чего используются методы ThenBy, ThenByDescending?

Используются для вторичной сортировки данных после первичной сортировки, выполненной с помощью методов OrderBy или OrderByDescending.

**3 Вывод**

3.1 Я изучил процесс обработки списков стандартными методами на языке C#.

**4 Ход работы**

4.3 Написать LINQ-запросы, возвращающие:

- список расширений файлов в директории (без повторений), используя Distinct(),

- список расширений файлов в директории и количество файлов каждого

расширения, используя GroupBy() и Count().

DirectoryInfo directory = new DirectoryInfo(@"C:\temp\ispp31");

FileInfo[] files = directory.GetFiles("\*", SearchOption.AllDirectories);

var sortedFiles = files

.OrderBy(f => f.Name)

.ThenByDescending(f => f.CreationTime)

.Select(f => new { f.Name, f.DirectoryName, f.Length, f.CreationTime })

.ToList();

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

string search = Console.ReadLine();

var searchedFiles = files.Where(f => f.Name.Contains(search)).ToList();

var countSearchedFiles = files.Count(f => f.Name.Contains(search));

Console.WriteLine(countSearchedFiles);

4.4 Написать LINQ-запрос, возвращающий список последних 10 файлов, созданных за текущий день, используя Where(), OrderByDescending() и Take().

var extension = files.Select(f => f.Extension).Distinct()

.ToList();

var extensionCount = files.GroupBy(f => f.Extension)

.Select(f => new { f.Key, Count = f.Count() })

.ToList();

4.5 Написать LINQ-запрос, возвращающий имя файла, его расширение, путь к

файлу, его размер с указанием единицы измерения, используя Select().

Размер файла — целое число. Единица измерения:

- Б (если размер меньше 1КБ),

- КБ (если размер меньше 1МБ),

- МБ (в остальных случаях).

Для получения размера можно использовать побитовый сдвиг влево на 2сдвиг: размер >> сдвиг (для КБ сдвиг 10, для МБ сдвиг 20 и т.д.).

var todayFiles = files.Where(f => f.CreationTime.Date == DateTime.Today)

.OrderByDescending(f => f.CreationTime)

.Take(10)

.ToList();

**Лабораторная работа №18**

**Проектирование диаграммы классов**

**1 Цель работы**

1.1 Изучить процесс описания типов данных с применением диаграммы классов.

**2 Контрольные вопросы**

2.1 Для чего используется диаграмма классов?

Диаграмма классов используется для визуального представления структуры и взаимосвязей классов внутри системы.

2.2 Из каких элементов состоит описание типа в диаграмме классов?

Описание типа в диаграмме классов включает следующие элементы: Имя класса, атрибуты класса, классовые операции (методы).

2.3 На какие группы делятся взаимосвязи в диаграмме классов?

Взаимосвязи в диаграмме классов делятся на группы: зависимость, реализация, наследование, ассоциация, агрегация, композиция.

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

Основное отличие между взаимосвязью «наследование» и «реализация» заключается в уровне описания отношений между объектами.

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

Композиция предполагает жёсткую зависимость, а агрегация — слабую.

2.6 Что показывает взаимосвязь «ассоциация»?

Взаимосвязь «ассоциация» показывает, что объекты одной сущности связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому.

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

В диаграмме классов могут быть указаны следующие виды мощности отношений:

* 1 — в точности одна связь;
* N — неограниченное число (0 или больше);
* 0..N — ноль или больше;
* 1..N — одна или больше;
* 3..7 — указанный интервал.
* **3 Вывод**

3.1 Я изучил 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 используются для решения проблем, связанных с созданием объектов и экземпляров классов.

2.2 Какие паттерны относятся к порождающим?

К порождающим паттернам (Creational patterns) относятся:

* Фабричный метод;
* Абстрактная фабрика;
* Одиночка;
* Строитель;
* Прототип.

2.3 Для чего используются структурные паттерны?

Структурные паттерны используются для организации классов и объектов в крупные системы.

2.4 Какие паттерны относятся к структурным?

К структурным паттернам относятся:

* Адаптер;
* Мост;
* Компоновщик;
* Декоратор;
* Фасад;
* Приспособленец;
* Заместитель.

2.5 Для чего используются поведенческие паттерны?

Поведенческие паттерны в C# используются для эффективного взаимодействия между объектами, распространения информации между ними и координации их работы.

2.6 Какие паттерны относятся к поведенческим?

Некоторые поведенческие паттерны:

* Цепочка обязанностей;
* Команда;
* Посредник;
* Наблюдатель;
* Состояние;
* Стратегия.

**3 Вывод**

3.1 Я смог научиться применять паттерны проектирования в разработке программ.

**4 Ход работы**

4.2 Реализовать и объяснить порождающий паттерн «Фабричный метод» согласно

UML-диаграмме на рисунке 2.

Добавить еще одну фигуру к описанному интерфейсу.

https://www.tutorialspoint.com/design\_pattern/factory\_pattern.htm

Для сравнения строк без учета регистра вместо equalsIgnoreCase используется:

строка1.Equals(строка2, StringComparison.OrdinalIgnoreCase)

public interface IShape

{

void Draw();

}

public class Circle : IShape

{

public void Draw()

=> Console.WriteLine("Inside Circle::draw() method.");

}

public class Square : IShape

{

public void Draw()

=> Console.WriteLine("Inside Square::draw() method.");

}

public class Rectangle : IShape

{

public void Draw()

=> Console.WriteLine("Inside Rectangle::draw() method.");

}

public class Pentagon : IShape

{

public void Draw()

=> Console.WriteLine("Inside Pentagon::draw() method.");

}

public class ShapeFactory

{

public IShape GetShape(string shapeType)

{

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

return new Circle();

else if (shapeType.Equals("RECTANGLE", StringComparison.OrdinalIgnoreCase))

return new Rectangle();

else if (shapeType.Equals("SQUARE", StringComparison.OrdinalIgnoreCase))

return new Square();

else if (shapeType.Equals("PENTAGON", StringComparison.OrdinalIgnoreCase))

return new Pentagon();

return null;

}

}

ShapeFactory shapeFactory = new();

IShape shape = shapeFactory.GetShape("CIRCLE");

shape.Draw();

shape = shapeFactory.GetShape("RECTANGLE");

shape.Draw();

shape = shapeFactory.GetShape("SQUARE");

shape.Draw();

shape = shapeFactory.GetShape("PENTAGON");

shape.Draw();

4.3 Реализовать и объяснить структурный паттерн «Декоратор» согласно UMLдиаграмме на рисунке 3.

Добавить еще один декоратор к описанному интерфейсу.

<https://www.tutorialspoint.com/design_pattern/decorator_pattern.htm>

public interface IShape

{

void Draw();

}

public class Circle : IShape

{

public void Draw()

=> Console.WriteLine("Shape: Circle");

}

public class Rectangle : IShape

{

public void Draw()

=> Console.WriteLine("Shape: Rectangle");

}

public class Square : IShape

{

public void Draw()

=> Console.WriteLine("Shape: Square");

}

public class RedShapeDecorator : ShapeDecorator

{

public RedShapeDecorator(IShape decoratedShape) : base(decoratedShape)

{

}

public void Draw()

{

\_decoratedShape.Draw();

SetRedBorder(\_decoratedShape);

}

private void SetRedBorder(IShape decoratedShape)

{

Console.WriteLine("Border Color: Red");

}

}

public abstract class ShapeDecorator : IShape

{

protected IShape \_decoratedShape;

public ShapeDecorator(IShape decoratedShape)

=> \_decoratedShape = decoratedShape;

public void Draw()

=> \_decoratedShape.Draw();

}

IShape circle = new Circle();

Console.WriteLine("Circle with normal border");

circle.Draw();

IShape redCircle = new RedShapeDecorator(new Circle());

Console.WriteLine("\nCircle of red border");

redCircle.Draw();

IShape redRectangle = new RedShapeDecorator(new Rectangle());

Console.WriteLine("\nRectangle of red border");

redRectangle.Draw();

IShape redSquare = new RedShapeDecorator(new Square());

Console.WriteLine("\nSquare of red border");

redSquare.Draw();

4.4 Реализовать и объяснить поведенческий паттерн «Наблюдатель» согласно UMLдиаграмме на рисунке 4. Вместо списка наблюдателей использовать модель событий.

Добавить еще одного наблюдателя, отображающего число в десятичной системе.

https://www.tutorialspoint.com/design\_pattern/observer\_pattern.htm

Для перевода в систему счисления по основанию 2/8/10/16 вместо Integer.toОснованиеString используется:

Convert.ToString(число, основание)

public abstract class Observer

{

protected Observer(Subject subject)

{

\_subject = subject;

\_subject.NotifyObservers += Update;

}

protected Subject \_subject;

public abstract void Update();

}

public class BinaryObserver : Observer

{

public BinaryObserver(Subject subject) : base(subject) { }

public override void Update()

=> Console.WriteLine($"Binary String: {Convert.ToString(\_subject.GetState(), 2)}");

}

public class OctalObserver : Observer

{

public OctalObserver(Subject subject) : base(subject) { }

public override void Update()

=> Console.WriteLine($"Octal String: {Convert.ToString(\_subject.GetState(), 8)}");

}

public class HexaObserver : Observer

{

public HexaObserver(Subject subject) : base(subject) { }

public override void Update()

=> Console.WriteLine($"Hex String: {Convert.ToString(\_subject.GetState(), 16).ToUpper()}");

}

public class Subject

{

private int \_state;

public event Action NotifyObservers;

public int GetState()

=> \_state;

public void SetState(int state)

{

\_state = state;

NotifyObservers?.Invoke();

}

}

Subject subject = new();

Observer BinObserver = new BinaryObserver(subject);

Observer OctObserver = new OctalObserver(subject);

Observer HexObserver = new HexaObserver(subject);

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

subject.SetState(15);

Console.WriteLine();

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

subject.SetState(10);

**Лабораторная работа №20**

**Проектирование API**

**1 Цель работы**

1.1 Научиться проектировать API.

**2 Контрольные вопросы**

2.1 Что такое REST?

Это архитектурный стиль взаимодействия между клиентом и сервером через HTTP протокол.

2.2 Для чего используется метод GET?

Для запроса информации о ресурсе.

2.3 Для чего используется метод POST?

Для создания нового ресурса на сервере.

2.4 Для чего используется метод PUT?

Для обновления существующих ресурсов или создания новых, если они не существуют.

2.5 Для чего используется метод DELETE?

Для удаления ресурса с сервера.

**3 Вывод**

3.1 Я научился проектировать API.

**4 Ход работы**

4.1 Создать конечные точки, возвращающие:

- список всех котов,

- кота по id (если требуемого нет – возвращать 404).

[ApiController]

[Route("[controller]")]

public class CatController : ControllerBase

{

private static List<Cat> \_cats = new()

{

new Cat { Id = 1, Name = "Усатик", Breed = "Персидская", Color = "Белый", Age = 5 },

new Cat { Id = 2, Name = "Варежка", Breed = "Сиамская", Color = "Сил-пойнт", Age = 3 },

new Cat { Id = 3, Name = "Дымок", Breed = "Британская короткошерстная", Color = "Серый", Age = 7 },

new Cat { Id = 4, Name = "Белла", Breed = "Мейн-кун", Color = "Рыжий", Age = 4 },

new Cat { Id = 5, Name = "Макс", Breed = "Рэгдолл", Color = "Сил-миттед", Age = 2 },

new Cat { Id = 6, Name = "Луна", Breed = "Персидская", Color = "Трехцветный", Age = 6 },

new Cat { Id = 7, Name = "Оливер", Breed = "Британская короткошерстная", Color = "Черный", Age = 8 },

new Cat { Id = 8, Name = "Клео", Breed = "Сиамская", Color = "Коричневый", Age = 3 },

new Cat { Id = 9, Name = "Симба", Breed = "Мейн-кун", Color = "Рыжий", Age = 5 },

new Cat { Id = 10, Name = "Ромашка", Breed = "Персидская", Color = "Белый", Age = 2 },

new Cat { Id = 11, Name = "Тест", Breed = "Персидская", Color = "Белый", Age = 2 },

new Cat { Id = 12, Name = "Ромашка", Breed = "Персидская", Color = "Белый", Age = 2 }

};

[HttpGet("/cats")]

public ActionResult<List<Cat>> GetList(int page = 1, int pageSize = int.MaxValue)

{

var cats = \_cats.Skip((page - 1) \* pageSize).Take(pageSize).ToList();

return Ok(cats);

}

[HttpGet("/cats/{id:int}")]

public ActionResult<Cat> GetById(int id)

{

var cat = \_cats.FirstOrDefault(c => c.Id == id);

if (cat == null)

return NotFound();

return Ok(cat);

}

}

4.2 Создать конечную точку, возвращающую всех котов определенной породы, используя параметр конечной точки.

[HttpGet("/cats/breed")]

public ActionResult<List<Cat>> GetByBreed(string breed)

=> Ok(\_cats.Where(c => c.Breed == breed));

[HttpPost]

public ActionResult<Cat> Create([FromBody]Cat cat)

{

try

{

cat.Id = \_cats.Count() + 1;

\_cats.Add(cat);

return Created();

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

4.3 Изменить код конечной точки из п.3.1, чтобы можно было передать номер страницы и размер страницы. Для этого добавить параметры в метод конечной точки.

[HttpGet("/cats")]

public ActionResult<List<Cat>> GetList(int page = 1, int pageSize = int.MaxValue)

{

var cats = \_cats.Skip((page - 1) \* pageSize).Take(pageSize).ToList();

return Ok(cats);

}

4.4 Создать конечную точку, удаляющую кота по id. Возвращать 204 в случае успешного удаления, иначе – соответствующий код ошибки.

[HttpDelete("{id:int}")]

public ActionResult Delete(int id)

{

var cat = \_cats.FirstOrDefault(\_c => \_c.Id == id);

if (cat is not null)

{

\_cats.Remove(cat);

return NoContent();

}

return NotFound();

}

4.5 Создать конечную точку, добавляющую нового кота. Возвращать 201 в случае успешного добавления, иначе – соответствующий код ошибки.

[HttpPost]

public ActionResult<Cat> Create([FromBody]Cat cat)

{

try

{

cat.Id = \_cats.Count() + 1;

\_cats.Add(cat);

return Created();

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

**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»?

Это процесс улучшения существующего кода для повышения его производительности и эффективности.

2.2 Какова цель оптимизации программного кода?

Снижение ресурсов и времени на выполнение операций.

2.3 Какие методы оптимизации программного кода применяются?

Оптимизация циклов, устранение избыточных операций, использование подходящих структур данных.

**3 Вывод**

3.1 Были изучены 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.

2.2 Какие типы возврата могут быть у асинхронных методов и для чего предназначен каждый из типов?

Task - используется, если метод не возвращает значение.

Task<TResult> - Применяется, если метод возвращает значение типа TResul.

void - Используется в асинхронных обработчиках событий.

2.3 Как вызвать метод в асинхронном режиме?

async тип\_возвращаемого\_значения название\_метода (параметры)

2.4 Как указать, что в методе могут быть асинхронные вызовы?

В языке C# указать, что в методе могут быть асинхронные вызовы, можно с помощью ключевого слова async.

2.5 Как обработать исключения, возникшие в асинхронных вызовах?

Чтобы обработать исключения, можно использовать блок try-catch.

**3 Вывод**

3.1 Я научился реализовывать и запускать асинхронные операции на C#.

3.2 Я научился выполнять вычисления, используя асинхронные операции.

3.3 Я научился выполнять ввод и вывод данных, используя асинхронные операции.

**4 Ход работы**

4.3 Оптимизация условий

4.3.1 Вынести инвариантные условия из цикла

public void FindSum(int[] numbers)

{

int sum = 0

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

{

if (numbers.Length == 0)

throw new InvalidOperationException("в массиве нет чисел");

sum += numbers[i];

}

Console.WriteLine($"sum = {sum}");

}

void FindSum(int[] numbers)

{

if (numbers.Length == 0)

throw new InvalidOperationException("В массиве нет чисел.");

int sum = 0;

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

sum += numbers[i];

Console.WriteLine($"sum = {sum}");

}

5.3.2 Выполнить оптимизацию программного кода, уменьшив количество

проверок и оптимизировав порядок условий:

private int GetDaysCount(int month, int year)

{

switch (month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

return 31;

case 4:

case 6:

case 9:

case 11:

return 30;

case 2:

return (year % 400 ==0 || year % 100 != 0 && year % 4 == 0) ? 29 : 28;

default:

return -1;

}

}

int GetDaysCount(int month, int year)

{

if (month < 1 || month > 12 )

return -1;

switch (month)

{

case 2:

return (year % 400 == 0 || year % 100 != 0 && year % 4 == 0) ? 29 : 28;

case 4:

case 6:

case 9:

case 11:

return 30;

default:

return 31;

}

}

4.4 Оптимизация при работе со строками

Преобразовать все данные списка cats (из файла LINQ.txt) в одну строку с данными

в следующем формате:

- в начале названия свойств, отделенные точкой с запятой,

- затем каждая отдельная строка – значения свойств отдельного объекта из списка.

Пример отображения информации:

Id;Name;Age;

1;Мурка;10;

2;Барсик;3;

Полученная строка должна быть выведена на экран.

Для оптимизации вместо конкатенации и сложения строк использовать методы

StringBuilder.

public class Cat

{

public int Id { get; set; }

public string Name { get; set; }

public string Breed { get; set; }

public string Color { get; set; }

public int Age { get; set; }

}

List<Cat> cats = new List<Cat>

{

new Cat { Id = 1, Name = "Усатик", Breed = "Персидская", Color = "Белый", Age = 5 },

new Cat { Id = 2, Name = "Варежка", Breed = "Сиамская", Color = "Сил-пойнт", Age = 3 },

new Cat { Id = 3, Name = "Дымок", Breed = "Британская короткошерстная", Color = "Серый", Age = 7 },

new Cat { Id = 4, Name = "Белла", Breed = "Мейн-кун", Color = "Рыжий", Age = 4 },

new Cat { Id = 5, Name = "Макс", Breed = "Рэгдолл", Color = "Сил-миттед", Age = 2 },

new Cat { Id = 6, Name = "Луна", Breed = "Персидская", Color = "Трехцветный", Age = 6 },

new Cat { Id = 7, Name = "Оливер", Breed = "Британская короткошерстная", Color = "Черный", Age = 8 },

new Cat { Id = 8, Name = "Клео", Breed = "Сиамская", Color = "Коричневый", Age = 3 },

new Cat { Id = 9, Name = "Симба", Breed = "Мейн-кун", Color = "Рыжий", Age = 5 },

new Cat { Id = 10, Name = "Ромашка", Breed = "Персидская", Color = "Белый", Age = 2 }

};

StringBuilder stringBuilder = new("Id;Name;Age;Breed;Color;\n");

foreach (var cat in cats)

stringBuilder.AppendLine($"{cat.Id};{cat.Name};{cat.Age};{cat.Breed};{cat.Color};");

Console.WriteLine(stringBuilder);

4.5 Сохранение промежуточных результатов

4.5.1 Написать и протестировать рекурсивную функцию вычисления факториала.

Для оптимизации использовать мемоизацию (сохранять ранее вычисленные

значения в словаре Dictionary<int, long>, где ключ – число, значение – факториал

числа). При вычислении проверять. Есть ли значение в словаре: если есть, сразу

возвращать, если нет – вычислять и добавлять в словарь.

class Factorial

{

public static Dictionary<uint, long> factorialCash = new();

public static long GetFactorial(uint number)

{

if (number <= 1)

return 1;

if (factorialCash.ContainsKey(number))

return factorialCash[number];

factorialCash[number] = number \* GetFactorial(number - 1);

return factorialCash[number];

}

}

Console.WriteLine($"Факториал 5: {Factorial.GetFactorial(5)}\n");

Console.WriteLine($"Факториал 6: {Factorial.GetFactorial(6)}\n");

4.5.2 Написать и протестировать рекурсивную функцию быстрого вычисления xn

,

где n неотрицательное целое, используя возведение в квадрат.

Для ускорения работы рекурсия должна вызываться в ветке алгоритма не более

одного раза.

Пример использования алгоритма быстрого возведения в степень:



Для некорректных данных возвращать -1. Стандартный метод возведения в

степень не использовать.

Console.WriteLine($"Возведение 2 в 15 степень: {Exponential(2, 15)}");

double Exponential(double x, int n)

{

if (n < 0)

return -1;

if (n == 0)

return 1;

double temp = Exponential(x, n / 2);

if (n % 2 != 0)

return x \* temp \* temp;

return temp \* temp;

}

**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 описание назначения класса и его полей, свойств, методов, событий и кода с помощью специальных тегов XML.

2.2 Как сгенерировать XML-комментарий?

В Visual Studio есть функция автоматической генерации структуры XML-комментариев для элементов кода, таких как методы, классы и свойства.

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

* Указать ссылку на файл документации в проекте, чтобы Visual Studio поняла, что это файл комментариев от библиотеки. Для этого в свойствах проекта нужно указать путь до файла документации.

2.4 Что пишется в разделе summary?

В разделе summary тега в документации XML пишется краткое описание элемента.

2.5 Что пишется в разделе param?

В разделе «param» указывается имя и описание конкретного параметра.

2.6 Что пишется в разделе returns?

В разделе «returns» пишется описание возвращаемого значения метода.

**3 Вывод**

3.1 Был изучен 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 — процесс изменения внутренней структуры программы, который не затрагивает её внешнего поведения.

2.2 Какие группы техник рефакторинга существуют?

Составление методов, перемещение функций между объектами, организация данных, решение задач обобщения.

2.3 Как выполнить рефакторинг в Visual Studio?

Некоторые команды рефакторинга, доступные в контекстном меню Visual Studio:

* Extract Method — извлечь сегмент кода в отдельный метод.
* Encapsulate Field — инкапсулировать переменную в существующий класс.
* Extract Interface — выделить основные методы в виде интерфейса.
* Reorder Parameters — переупорядочить параметры.

**3 Вывод**

3.1 Я изучил и применил 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 — место, где хранятся и поддерживаются какие-либо данные.

2.2 Что указывается в файле readme.md?

Файл README.md содержит вводное руководство по проекту.

2.3 Что указывается в файле .gitignore?

В файле .gitignore указывается список файлов и папок проекта, которые Git должен игнорировать и не отслеживать.

2.4 Какое программное обеспечение может применяться для управления git репозиторием?

Для управления Git-репозиторием можно использовать графическое программное обеспечение (GUI) или инструменты командной строки (CLI).

2.5 Где может располагаться репозиторий?

Репозиторий может располагаться в разных местах в зависимости от его типа:

-Централизованный репозиторий.

-Локальный репозиторий.

-Распределенный репозиторий.

-Сетевые репозитории.

2.6 Какой основной порядок выполнения команд при работе с репозиторием?

Основной порядок выполнения команд при работе с репозиторием в Git включает следующие шаги:

1) Инициализация репозитория;

2) Добавление файлов;

3) Создание коммита;

4) Отправка изменений;

5) Получение изменений из удалённого репозитория.

**3 Вывод**

3.1 Я смог научиться 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 размещения элементов внутри контейнера при проектировании пользовательского интерфейса приложения.

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

System.Windows.Controls.Panel — родительский класс для всех элементов-контейнеров в WPF..

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

В WPF элементы внутри контейнера выравниваются по горизонтали и по вертикали с помощью свойств: HorizontalAlignment и VerticalAlignment.

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

Grid располагает элементы в невидимой сетке строк и столбцов.

Canvas поддерживает абсолютное позиционирование элементов с использованием точных координат.

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

StackPanel упорядочивает дочерние элементы в одну строку, которая может быть ориентирована по горизонтали или вертикали.

DockPanel позволяет прикреплять дочерние элементы к краям панели сверху, снизу, слева или справа.

WrapPanel располагает элементы последовательным образом слева направо или сверху вниз в зависимости от значения свойства Orientation.

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

Пиксели, Дюймы, Сантиметры, Точки.

**3 Вывод**

3.1 Я изучил 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?

Элемент управления Frame в WPF используется для размещения другого содержимого XAML внутри контейнера.

2.2 Для чего используется элемент управления Page?

Элемент управления Page в WPF используется для создания страничных интерфейсов и поддержки навигации.

2.3 Как перейти к определенной странице, используя фрейм?

Чтобы перейти к определённой странице в приложении WPF с использованием фрейма, нужно установить свойство Source элемента Frame в значение, соответствующее URI желаемой страницы.

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

Чтобы проверить, можно ли во фрейме WPF вернуться к предыдущей странице, используют свойство CanGoBack.

2.5 Как перейти к предыдущей странице, используя фрейм?

Необходимо вызвать метод GoBack фрейма, чтобы осуществить возврат.

2.6 Какие элементы позволяют сгруппировать содержимое?

В WPF для группировки содержимого используются элементы GroupBox и Expander.

**3 Вывод**

3.1 Был изучен 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 в приложениях WPF используются для создания единообразного внешнего вида для набора элементов управления.

2.2 Какова общая форма локального определения стиля элемента управления?

Стиль элемента управления в WPF определяется с помощью класса Style.

2.3 Какова общая форма определения стиля приложения?

Стиль в WPF — это набор настроек DependencyProperty, которые при применении к целевому объекту изменяют его поведение.

2.4 Как указать явное использование стилей?

Чтобы указать явное использование стилей в WPF, нужно явно сослаться на стиль в элементе. Для этого используется свойство Style элемента, которому присваивается значение, определенное в стиле.

2.5 Как указать наследование стиля?

В WPF для указания наследования стиля используется свойство BasedOn класса Style.

2.6 Как добавить новую тему в приложение?

Чтобы добавить новую тему в приложение WPF, нужно определить стили в словаре ресурсов и применить их к элементам приложения.

2.7 Как выполнить переключение между темами?

Переключение между темами в WPF осуществляется через изменение словаря ресурсов (ResourceDictionary). Этот подход позволяет динамически применять разные наборы стилей к элементам управления во время выполнения.

**3 Вывод**

3.1 Был изучен процесс настройки интерфейса с использованием стилей в приложениях WPF.

**4 Ход работы**

4.2 Создание стиля для контейнера (элемента компоновки или окна).

4.2.1 Создать WPF-приложение для авторизации. Настроить окно авторизации

аналогично настройкам окна авторизации из п.5.1 (включая папки с ресурсами).

У элементов убрать из тэгов все атрибуты кроме подписи и указания строки и

столбца размещения в Grid. У StackPanel задать горизонтальную ориентацию.

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Content="Логин"/>

<TextBox Grid.Column="1"/>

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

<TextBox Grid.Column="1" Grid.Row="1"/>

<StackPanel Grid.Column="1" Grid.Row="2" Orientation="Horizontal">

<Button Content="ОК"/>

<Button Content="Отмена"/>

</StackPanel>

<Image Grid.Row="2" HorizontalAlignment="Left" VerticalAlignment="Center" Width="100" Height="100" Source="Images/galaxy.jpg"/>

</Grid>

4.2.2 Настроить стили дочерних элементов управления в контейнере:

- у меток задать цвет текста и шрифт, отличный от стандартного,

- у полей ввода задать максимальную длину строки (MaxLength) — 10, размер

текста — 20 и тот же шрифт, что у меток,

- у кнопок задать высоту 20, ширину – 100, вертикальное выравнивание по верхнему

краю, цвет фона отличный от стандартного, отступы (Margin) – 5 для всех кроме отступа

слева (значение: 0 5 5 5).

<Application.Resources>

<Style TargetType="Label">

<Setter Property="Foreground" Value="DarkBlue"/>

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="TextBox">

<Setter Property="MaxLength" Value="10"/>

<Setter Property="FontSize" Value="20"/>

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="Button">

<Setter Property="Height" Value="20"/>

<Setter Property="Width" Value="100"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Background" Value="Lavender"/>

<Setter Property="Margin" Value=" 0 5 5 5"/>

</Style>

</Application.Resources>

4.3 Создание стиля приложения

4.3.1 Создать WPF-приложение аналогично описанию из п.5.2.1 (включая папки с ресурсами).

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Content="Логин"/>

<TextBox Grid.Column="1"/>

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

<TextBox Grid.Column="1" Grid.Row="1"/>

<StackPanel Grid.Column="1" Grid.Row="2" Orientation="Horizontal">

<Button Content="ОК"/>

<Button Content="Отмена"/>

</StackPanel>

<Image Grid.Row="2" HorizontalAlignment="Left" VerticalAlignment="Center" Width="100" Height="100" Source="Images/galaxy.jpg"/>

</Grid>

4.3.2 Настроить стили приложения для выполнения требований из п.5.2.2.

<Application.Resources>

<Style TargetType="Label">

<Setter Property="Foreground" Value="DarkBlue"/>

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="TextBox">

<Setter Property="MaxLength" Value="10"/>

<Setter Property="FontSize" Value="20"/>

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="Button">

<Setter Property="Height" Value="20"/>

<Setter Property="Width" Value="100"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Background" Value="Lavender"/>

<Setter Property="Margin" Value=" 0 5 5 5"/>

</Style>

</Application.Resources>

4.4 Наследование стиля

4.4.1 Создать WPF-приложение аналогично описанию из п.5.2.1 (включая папки с ресурсами).

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Content="Логин"/>

<TextBox Grid.Column="1"/>

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

<TextBox Grid.Column="1" Grid.Row="1"/>

<StackPanel Grid.Column="1" Grid.Row="2" Orientation="Horizontal">

<Button Content="ОК"/>

<Button Style="{StaticResource CancleButton}" Content="Отмена"/>

<Button x:Name="ThemeChangeButton" Content="Поменять тему" Click="ThemeChangeButton\_Click"/>

</StackPanel>

<Image Grid.Row="2" HorizontalAlignment="Left" VerticalAlignment="Center" Width="100" Height="100" Source="Images/galaxy.jpg"/>

</Grid>

4.4.2 Настроить стили приложения для Control, чтобы у всех элементов управления был одинаковый шрифт (не из стандартных, можно выбрать из папки Fonts), размер текста, цвет текста, отступы. Имя стиля не указывать.

5.4.3 Создать стили остальных используемых в приложении элементов управления на основе стиля для Control.

5.4.4 Добавить наследование исходного стиля кнопок для кнопок Отмена (цвет кнопок должен быть красным). Указать имя стиля и задать его явное использование для кнопок Отмена в окнах авторизации и регистрации.

<Style TargetType="Control">

<Setter Property="Background" Value="DarkBlue"/>

<Setter Property="Foreground" Value="Lavender"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="Margin" Value=" 0 5 5 5"/>

<Setter Property="FontFamily" Value="./Fonts/#Roboto Thin"/>

</Style>

<Style TargetType="Label" BasedOn="{StaticResource {x:Type Control}}"/>

<Style TargetType="TextBox" BasedOn="{StaticResource {x:Type Control}}"/>

<Style TargetType="Button" BasedOn="{StaticResource {x:Type Control}}">

<Setter Property="Height" Value="20"/>

<Setter Property="Width" Value="100"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Background" Value="DarkBlue"/>

</Style>

<Style x:Key="CancleButton" TargetType="Button" BasedOn="{StaticResource {x:Type Button}}">

<Setter Property="Background" Value="Red"/>

</Style>

**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 в приложениях WPF позволяют динамически изменять внешний вид и поведение элемента управления без необходимости создавать новый.

2.2 Какие виды триггеров можно разработать в приложениях WPF?

Простой триггер (Trigger); Триггер данных (DataTrigger); Триггер события (EventTrigger); Множественный триггер (MultiTrigger).

2.3 Для чего используется и когда срабатывает триггер свойств?

Триггер свойств в WPF используется для изменения внешнего вида элемента управления в зависимости от состояния другого свойства.

2.4 Для чего используется и когда срабатывает триггер данных?

Триггер данных (DataTrigger) в WPF используется для изменения внешнего вида элемента интерфейса в зависимости от значения связанного свойства.

2.5 Для чего используется и когда срабатывает триггер событий?

Триггер событий (EventTrigger) в WPF используется для изменения внешнего вида или поведения элемента управления при возникновении конкретного события.

**3 Вывод**

3.1 Я изучил процесс применения триггеров в приложениях WPF и закрепил 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 в WPF — это элемент управления, который позволяет пользователю выбирать один или несколько вариантов из набора.

2.2 Что такое RadioButton и для чего он используется?

RadioButton — элемент управления, который позволяет пользователю выбрать одну опцию из группы взаимоисключающих вариантов.

2.3 Как проверить, что флажок или переключатель выбран?

Проверить, выбран ли флажок или переключатель, можно с помощью свойства IsChecked или события Checked.

2.4 Какое событие срабатывает при выборе флажка или переключателя?

Событие Checked срабатывает при выборе флажка.

2.5 Какое событие срабатывает при снятии выбора флажка или переключателя?

Событие Unchecked срабатывает при снятии флажка.

2.6 Какие значения могут принимать флажки и переключатели?

True; False; Null.

**3 Вывод**

3.1 Я изучил 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?

Имя задаётся с помощью атрибута x:Name.

2.2 Как создать обработчик события в WPF?

Для создания обработчика событий в XAML нужно добавить атрибут события к элементу управления.

2.3 Для чего используется Slider в WPF?

Slider в WPF используется для выбора числового значения из диапазона путём перемещения указателя вдоль горизонтальной или вертикальной линии.

2.4 Для чего используется TextBox в WPF?

TextBox в WPF используется для ввода и редактирования текста.

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

Label обычно используется для отображения статического текста, а TextBlock — для работы с динамическим контентом и многострочным текстом.

2.6 Для чего используется Calendar в WPF?

Элемент управления Calendar в WPF используется для выбора даты с помощью визуального отображения календаря.

2.7 Для чего используется DatePicker в WPF?

DatePicker в WPF используется для того, чтобы пользователь мог выбрать дату, введя её в текстовое поле или используя раскрывающийся список календаря.

2.8 Для чего используется PasswordBox в WPF?

PasswordBox в WPF используется для ввода парольной информации.

**3 Вывод**

3.1 Я смог изучить различные типы полей ввода, применяющихся в приложениях WPF и свойства полей ввода и процесс обработки событий полей ввода.

**4 Ход работы**

4.2 Работа с полями ввода дат

4.2.1 Добавить на экран профиля пользователя DatePicker для ввода даты рождения.

Настроить в разметке свойство DisplayDateStart (минимальную дату) на 01.01.1900.

Настроить программно свойство DisplayDateEnd (максимальную дату) на текущую дату.

<DatePicker x:Name="BirthDateDatePicker"

DisplayDateStart="01.01.1900"

SelectedDateChanged="BirthDateDatePicker\_SelectedDateChanged"/>

BirthDateDatePicker.DisplayDateEnd = DateTime.Now;

PasswordLengthTextBox.Text = $"Длина вашего пароля: {PasswordBox.Password.Length}";

4.2.2 При указании и изменении даты (событие SelectedDateChanged, свойство SelectedDate) выводить возраст в TextBlock. Для вычисления возраста: - найти разницу между текущим годом и годом рождения, - проверить, если дата рождения больше текущей даты, из которой вычли разницу между годами, вычесть из разницы ещё один год.

private void BirthDateDatePicker\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

int age = DateTime.Now.Year - BirthDateDatePicker.SelectedDate.Value.Year;

if (DateTime.Now.DayOfYear < BirthDateDatePicker.SelectedDate.Value.DayOfYear)

--age;

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

}

4.3 Работа со слайдером

4.3.1 Добавить на экран профиля пользователя Slider для ввода стажа (от 0 до 50 лет, реализовать отображение делений с шагом 5).

<Slider x:Name="ExperienceSlider" TickFrequency="5"

Minimum="0"

Maximum="50"

4.3.2 При изменении значения в слайдере выводить информацию о стаже в TextBox с учётом правил русского языка: Стаж: N год/года/лет N - целое число, которое берется из значения слайдера.

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

{

var experience = Math.Ceiling(ExperienceSlider.Value);

string yearTitle;

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

}

**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 — это элемент управления, который позволяет отображать несколько элементов в раскрывающемся списке и выбирать один из них за раз.

2.2 Что такое ListBox и для чего он используется?

ListBox — это элемент управления, который предоставляет список элементов для выбора пользователем.

2.3 Какое событие срабатывает при выборе элемента в селекторе?

SelectionChanged — событие, которое срабатывает при выборе элемента в селекторе.

2.4 В каком свойстве хранятся элементы селекторов?

SelectedItem, SelectedIndex, SelectedValue.

2.5 Какого типа элементы могут быть в селекторе?

Селектор может содержать коллекцию объектов любого типа.

2.6 Какое свойство позволяет привязать селектор к набору данных?

Свойство ItemsSource.

2.7 Для чего используется свойство DisplayMemberPath в селекторе?

Свойство DisplayMemberPath в селекторе в WPF используется для указания свойства объекта, значение которого будет отображаться в списке.

**3 Вывод**

3.1 Были изучены свойства и процесс обработки событий элементов отображения списков в приложениях WPF.

**4 Ход работы**

4.3 Настройка отображения элементов в виде карточек

4.3.1 Создать класс Product с автосвойствами Id, Name, Category, Price.

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Category { get; set; }

public double Price { get; set; }

}

4.3.2 Добавить в разметку приложения ListView. Добавить в приложение список List<Product> из пяти товаров и использовать его как ItemsSource в ListView.

<ListView x:Name="ProductsListView" ItemTemplate="{StaticResource VirtualizingStackPanel}"/>

List<Product> products = [

new Product

{

Id = 0,

Name = "Ока(Капсула смерти)",

Category = "Транспорт",

Price = 200\_000

},

new Product

{

Id = 1,

Name = "Стул из ИКЕИ",

Category = "Мебель",

Price = 1

},

new Product

{

Id = 2,

Name = "Револьвер",

Category = "Игрушки",

Price = 20

},

new Product

{

Id = 3,

Name = "Аккумулятор",

Category = "Электроника",

Price = 2000

},

new Product

{

Id = 0,

Name = "Нива",

Category = "Транспорт",

Price = 1000000

}

];

ProductsListView.ItemsSource = products;

4.3.3 Настроить у него отображение элементов в виде карточек, используя

ItemTemplate. Добавить в ресурсы приложения картинку, которая будет использоваться как заглушка для изображения всех товаров.

<DataTemplate x:Key="VirtualizingStackPanel">

<StackPanel Orientation="Horizontal">

<Image Source="/404.png" Width="30" Height="30"/>

<TextBlock Text="{Binding Name}" Width="100" VerticalAlignment="Center" Margin="10"/>

<StackPanel Width="100">

<TextBlock Text="{Binding Price}" />

<Button x:Name="BuyButton" Content="Купить" Click="BuyButton\_Click"/>

</StackPanel>

</StackPanel>

</DataTemplate>

4.4 Изменение содержимого списков

4.4.1 Добавить в разметку приложения ListBox для отображения названий товаров в корзине. Настроить у него DisplayMemberPath и разрешить выделение нескольких элементов с нажатием Shift/Control.

<ListBox x:Name="ConsumerBasketListBox" DisplayMemberPath="Name" SelectionMode="Multiple" Height="200"/>

5.4.2 При нажатии на кнопку «Купить» товар должен добавляться в корзину.

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

{

var product = (e.Source as Button).DataContext as Product;

ConsumerBasketListBox.Items.Add(product);

}

5.4.3 При нажатии на кнопку «Удалить» выделенные в корзине товары должны удаляться из нее (может потребоваться удаление элементов с конца списка).

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

{

Product[] products = new Product[ConsumerBasketListBox.SelectedItems.Count];

ConsumerBasketListBox.SelectedItems.CopyTo(products, 0);

foreach (var item in products)

ConsumerBasketListBox.Items.Remove(item);

}

5.5 Смена стиля отображения элементов списка

5.5.1 Добавить в ресурсы окна шаблоны панели элементов ItemsPanelTemplate и применить шаблон для списка в разметке:

- WrapPanel (для отображения элементов в виде плитки)

- VirtualizingStackPanel (для отображения элементов в виде списка)

<DataTemplate x:Key="VirtualizingStackPanel">

<StackPanel Orientation="Horizontal">

<Image Source="/404.png" Width="30" Height="30"/>

<TextBlock Text="{Binding Name}" Width="100" VerticalAlignment="Center" Margin="10"/>

<StackPanel Width="100">

<TextBlock Text="{Binding Price}" />

<Button x:Name="BuyButton" Content="Купить" Click="BuyButton\_Click"/>

</StackPanel>

</StackPanel>

</DataTemplate>

<DataTemplate x:Key="WrapPanel">

<WrapPanel Orientation="Vertical">

<Image Source="/404.png" Width="30" Height="30" HorizontalAlignment="Center"/>

<TextBlock Text="{Binding Name}" Width="100" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="30,5,0,5"/>

<StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding Price}" HorizontalAlignment="Left" VerticalAlignment="Center" Width="100" />

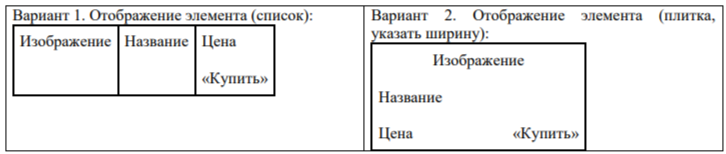
<Button x:Name="BuyButton" Content="Купить" Click="BuyButton\_Click" HorizontalAlignment="Right"/>

</StackPanel>

</WrapPanel>

</DataTemplate>

5.5.2 Добавить в ресурсы окна шаблоны элементов списка DataTemplate согласно макету и применить шаблон для списка в разметке:



<ListView x:Name="ProductsListView" ItemTemplate="{StaticResource VirtualizingStackPanel}"/>

**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?

Это элемент управления WPF для отображения данных в табличном формате.

2.2 Какие типы столбцов поддерживаются в DataGrid?

Текстовые данные, булевы значения, выпадающие списки и ссылки.

2.3 Как добавить кнопку в строки DataGrid?

Для добавления кнопки в строки используется DataGridTemplateColumn с шаблоном содержимого.

2.4 Как указать источник данных для DataGrid?

Источник данных указывается через свойство ItemSource.

2.5 Как указать источник данных для выпадающего списка DataGrid?

Источник данных для выпадающего списка задаётся через свойство ItemsSource.

**3 Вывод**

3.1 Я изучил свойства и процесс настройки внешнего вида элемента DataGrid и ListView в приложении WPF.

**4 Ход работы**

5.5 Привязка выбранного элемента к выпадающему списку

5.5.1 В конструкторе окна из п.5.2 создать список категорий с тремя строковыми значениями (email / БД / сайт).

List<string> categories = ["email", "сайт", "БД"];

5.5.2 Добавить в DataGrid столбец типа DataGridComboBoxColumn, который связать с категорией и заполнить значениями из коллекции строк с категориями.

<DataGridComboBoxColumn x:Name="CategoryDGComboBoxColumn" TextBinding="{Binding Category}" Header="Категория"/>

**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 - это элемент интерфейса пользователя, представляющий собой выпадающий список опций и команд. Используется для организации доступа к функциям программы через структурированное меню.

2.2 Что такое ContextMenu и для чего он используется?

ContextMenu - это всплывающее меню, которое появляется при правом клике мышью. Используется для быстрого доступа к операциям, связанным с текущим элементом или областью экрана.

2.3 Что такое ToolBar и для чего он используется?

ToolBar - это горизонтальная или вертикальная панель с кнопками и иконками.. Используется для быстрого доступа к наиболее часто используемым командам программы.

2.4 Что такое StatusBar и для чего он используется?

StatusBar - это информационная полоса в нижней части окна программы. Используется для отображения состояния приложения, прогресса выполнения операций и дополнительной информации о выбранных элементах.

**3 Вывод**

3.1 Я изучил свойства и процесс настройки внешнего вида меню и панели инструментов в приложении WPF.

**4 Ход работы**

4.2 Отображение строки состояния

<DockPanel>

<StatusBar x:Name="StatusBar" DockPanel.Dock="Bottom">

<StatusBarItem x:Name="TextLengthStatusBarItem" Content="Количество символов: 0"/>

<StatusBarItem x:Name="SizeStatusBarItem" HorizontalAlignment="Right">

<StackPanel Orientation="Horizontal">

<Button x:Name="ReduceSizeButton" Content="-" Click="ReduceSizeButton\_Click"/>

<TextBlock x:Name="SizeTextBlock" Text="{Binding StringFormat={}{0:F0}%, ElementName=SizeSlider, Path=Value}" Width="25"/>

<Slider x:Name="SizeSlider"

Minimum="1" Maximum="100" Width="100"

Value="50" Interval="10"/>

<Button x:Name="IncreaseSizeButton" Content="+" Click="IncreaseSizeButton\_Click"/>

</StackPanel>

</StatusBarItem>

<StatusBarItem

Content="{Binding Source={x:Static sys:DateTime.Today}}"

HorizontalAlignment="Right"/>

</StatusBar>

<Menu DockPanel.Dock="Top">

<MenuItem Header="Файл">

<MenuItem Header="Создать" InputGestureText="Ctrl+N">

<MenuItem.Icon>

<Image Source="/Images/plant.png"/>

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="Открыть" InputGestureText="Ctrl+O">

<MenuItem.Icon>

<Image Source="/Images/zombie.png"/>

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="Сохранить" InputGestureText="Ctrl+S" Command="Save">

<MenuItem.Icon>

<Image Source="/Images/zombie.png"/>

</MenuItem.Icon>

</MenuItem>

<Separator/>

<MenuItem Header="Печать" InputGestureText="Ctrl+P">

<MenuItem.Icon>

<Image Source="/Images/zombie.png"/>

</MenuItem.Icon>

</MenuItem>

<Separator/>

<MenuItem Header="Выход">

<MenuItem.Icon>

<Image Source="/Images/zombie.png"/>

</MenuItem.Icon>

</MenuItem>

</MenuItem>

<MenuItem Header="Формат">

<MenuItem Header="Перенос по словам" IsCheckable="True" IsChecked="True"/>

<MenuItem Header="Шрифт..."/>

</MenuItem>

<MenuItem Header="Вид">

<MenuItem Header="Масштаб">

<MenuItem Header="Увеличить" Click="IncreaseSizeButton\_Click"/>

<MenuItem Header="Уменьшить" Click="ReduceSizeButton\_Click"/>

<MenuItem Header="Восстановить масштаб по умолчанию"/>

</MenuItem>

<MenuItem

x:Name="StatusBarVisibilityMenuItem"

Header="Строка состояния"

IsCheckable="True"

IsChecked="True"

Checked="StatusBarVisibilityMenuItem\_Checked"

Unchecked="StatusBarVisibilityMenuItem\_Checked"/>

</MenuItem>

<MenuItem Header="Справка">

<MenuItem Header="Просмотреть справку"/>

<MenuItem Header="Отправить отзыв"/>

<Separator/>

<MenuItem Header="О программе"/>

</MenuItem>

</Menu>

<TextBox

x:Name="TextBox"

TextWrapping="Wrap"

AcceptsReturn="True"

VerticalScrollBarVisibility="Auto"

TextChanged="TextBox\_TextChanged"

FontSize="{Binding ElementName=SizeSlider, Path=Value}"/>

<DockPanel.ContextMenu>

<ContextMenu>

<MenuItem x:Name="Clear" Header="Очистить" Click=""/>

<Separator/>

<MenuItem Header="Открыть"/>

<MenuItem Header="Сохранить"/>

<Separator/>

<MenuItem Header="Выход"/>

</ContextMenu>

</DockPanel.ContextMenu>

</DockPanel>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

CommandBindings.Add(new CommandBinding(

ApplicationCommands.Save,

SaveExecuted,

SaveCanExecuted));

}

private void SaveCanExecuted(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = TextBox.Text.Length > 0;

}

private async void SaveExecuted(object sender, ExecutedRoutedEventArgs e)

{

SaveFileDialog SaveFileDialog = new();

SaveFileDialog.Filter = "Текстовый формат(\*.txt)|\*.txt";

if (!SaveFileDialog.ShowDialog().Value)

return;

using (StreamWriter writer = new(SaveFileDialog.FileName))

{

await writer.WriteAsync(TextBox.Text);

}

}

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

TextLengthStatusBarItem.Content = $"Количество символов: {TextBox.Text.Length}";

}

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

{

if (TextBox.FontSize > 10)

SizeSlider.Value -= 10;

}

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

{

SizeSlider.Value += 10;

}

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

{

StatusBar.Visibility = StatusBarVisibilityMenuItem.IsChecked ? Visibility.Visible : Visibility.Collapsed;

}

}

**Лабораторная работа №35**

**Создание проекта с использованием компонентов стандартных диалогов**

1. **Цель работы**
   1. Изучить процесс создания и применения стандартных диалоговых окон в приложениях WPF.

**2 Контрольные вопросы**

2.1 Что такое диалоговое окно?

Диалоговое окно - это специальное окно с определенной целью и пользовательским опытом, которое временно прерывает работу основного приложения для получения информации от пользователя или отображения важных сообщений.

2.2 Для чего используется OpenFileDialog?

OpenFileDialog используется для выбора файла для открытия из файловой системы Windows.

2.3 Для чего используется SaveFileDialog?

SaveFileDialog применяется для выбора имени и расположения файла при сохранении данных.

2.4 Что такое MessageBox и какие настройки можно ему задать?

MessageBox - это диалоговое окно для отображения информации или запроса действий пользователя. Можно настроить текст сообщения, заголовок окна, иконку и набор кнопок.

2.5 Для чего используется PrintDialog?

PrintDialog используется для выбора принтера и настройки параметров печати перед отправкой документа на печать.

**3 Вывод**

3.1 Я изучил 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?

Через свойство DialogResult окна или для модального режима через ShowDialog()

2.2 Какие значения может принимать переменная DialogResult?

True, false, null.

2.3 Как открыть окно в диалоговом режиме?

var dialog = new MyDialog();

dialog.ShowDialog();

2.4 Как открыть окно в недиалоговом режиме?

var window = new MyWindow();

window.Show();

2.5 В чем отличие между диалоговым и недиалоговым режимами работы?

В диалоговом режиме другие окна блокируются до закрытия диалога, а в недиалоговом режиме пользователь может взаимодействовать с другими окнами одновременно.

**3 Вывод**

3.1 Я изучил 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?

Метод Where используется для фильтрации данных в коллекциях.

2.2 Какие логические операторы могут применяться при составлении условий?

Логические И, ИЛИ, НЕ.

2.3 Как проверить, что строка начинается с определенного текста?

Для проверки начала строки надо использовать метод StartsWith()

2.4 Как проверить, что строка содержит определенный текст?

Для проверки содержания подстроки надо использовать метод Contains()

2.5 Как составить LINQ-запрос для фильтрации по нескольким критериям, указываемым пользователем?

Для фильтрации по нескольким критериям в LINQ используются логические операторы (&& для И, || для ИЛИ) между условиями внутри метода Where. Каждое условие описывает отдельный критерий фильтрации, а операторы объединяют их в единую логику выборки данных

**3 Вывод**

3.1 Я научился 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?

Вывод данных постранично необходим для оптимизации интернет трафика и для удобного доступа к информации.

2.2 Что такое «пагинация»?

Разбиения данных на страницы для последовательного отображения.

2.3 Для чего используется метод Take?

Метод Take используется для получения указанного количества элементов с начала последовательности данных.

2.4 Для чего используется метод Skip?

Метод Skip используется для пропуска определенного количества элементов в начале последовательности данных.

2.5 Для чего используется метод TakeWhile?

Метод TakeWhile берет элементы из последовательности до тех пор, пока выполняется заданное условие.

2.6 Для чего используется метод SkipWhile?

Метод SkipWhile пропускает элементы из последовательности до тех пор, пока выполняется заданное условие.

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

Для использования методов Take и Skip достаточно иметь любой объект типа IEnumerable<T>.

**3 Вывод**

3.1 Я научился применять LINQ-запросы для постраничного ввода 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?

Sum, Aggregate, Average, Max, Min, Count.

2.2 Что возвращает метод Distinct?

Уникальные элементы, удаляя дубликаты.

2.3 Для чего используется метод GroupBy?

GroupBy группирует элементы последовательности по определённому свойству, создавая коллекции элементов с одинаковыми значениями ключа.

2.4 Для чего используется метод Join?

Join объединяет две последовательности на основе общего свойства, создавая новые объекты, содержащие данные из обеих последовательностей.

2.5 Для чего используется метод GroupJoin?

GroupJoin объединяет две последовательности на основе общего свойства, но в отличие от Join, возвращает группу соответствующих элементов из второй последовательности для каждого элемента первой последовательности.

2.6 В чем отличие результатов, полученных при вызове LINQ-методов Concat, Union, Except, Intersect?

- Concat - объединяет все элементы

- Union - объединяет, удаляя дубликаты

- Except - элементы только первой последовательности

- Intersect - элементы, общие для обеих последовательностей

**3 Вывод**

3.1 Я научился 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?

Ellipse, Shape, Rectangle, Polygon, Polyline, Line

2.2 Для чего используется Path?

Path используется для создания сложных форм путем комбинирования различных геометрических элементов и кривых.

2.3 Какие виды трансформаций объектов доступны в приложениях WPF?

TranslateTransform, RotateTransform, ScaleTransform, SkewTransform, MatrixTransform.

2.4 Как указать заливку и контур графических объектов в WPF?

Заливка через Fill и Brush, а контур через Stroke.

**3 Вывод**

3.1 Был изучен 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 приложениях WPF?

Через DoubleAnimation для свойств Width и Height элемента.

2.2 Как задать анимацию цвета в приложениях WPF?

ColorAnimation для свойств Fill или Background.

2.3 Чем отличается покадровая анимация от плавной анимации в WPF?

Плавная анимация использует DoubleAnimation для непрерывного изменения значений, покадровая анимация использует ObjectAnimationUsingKeyFrames для изменения значений в определенные моменты времени.

2.4 Какие свойства позволяют управлять анимацией в WPF и для чего предназначено каждое свойство?

From: начальное значение, с которого будет начинается анимация;

To: конечное значение;

Duration: время анимации в виде объекта TimeSpan;

By: значение, которое указывает, насколько должно увеличиться анимируемое свойство. Свойство By используется вместо свойства To;

RepeatBehavior: позволяет установить, как анимация будет повторяться;

AccelerationRatio: задает ускорение анимации;

DecelerationRatio: устанавливает замедление анимации;

SpeedRatio: устанавливает скорость анимации. По умолчанию значение 1.0;

AutoReverse: при значении true анимация выполняется в противоположную сторону;

FillBehavior: определеяет поведение после окночания анимации. Если оно имеет значение Stop, то после окончания анимации объект возвращает прежние значения: buttonAnimation.FillBehavior = FillBehavior.Stop. Если же оно имеет значение HoldEnd, то анимация присваивает анимируемому свойству новое значение.

**3 Вывод**

3.1 Я изучил 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 совокупность различных типов медиаконтента (аудио, видео, изображения) и технологий их обработки.

2.2 Какие элементы позволяют отображать изображения в приложениях WPF?

Image и InkCanvas

2.3 Какие элементы позволяют воспроизводить аудио в приложениях WPF?

MediaElement и MediaPlayer

2.4 Какие элементы позволяют воспроизводить видео в приложениях WPF?

MediaElement.

**3 Вывод**

3.1 Был изучен процесс создания мультимедиа-приложений на WPF.

**4 Ход работы**

4.4 Разработка медиапроигрывателя

Создать приложение WPF, в котором пользователь выбирает аудио/видеофайл для воспроизведения. В приложении должны быть кнопки для пуска, паузы, остановки и метки для отображения общего и текущего времени в следующем виде: чч:мм:сс / чч:мм:сс

<DockPanel>

<Menu DockPanel.Dock="Top">

<MenuItem x:Name="LoadFileMenuItem" Header="Загрузить файл" Click="LoadFileMenuItem\_Click"/>

</Menu>

<StackPanel Orientation="Horizontal" DockPanel.Dock="Bottom">

<Button x:Name="PauseButton" Content="Пауза" Click="PauseButton\_Click"/>

<Button x:Name="PlayButton" Content="Пуск" Click="PlayButton\_Click"/>

<Button x:Name="StopButton" Content="Остановка" Click="StopButton\_Click"/>

<Label x:Name="CurrentTimelineLabel" Content="00:00:00 /00:00:00"/>

</StackPanel>

<MediaElement x:Name="MediaElement" LoadedBehavior="Manual" UnloadedBehavior="Manual"/>

</DockPanel>

public partial class MainWindow : Window

{

private DispatcherTimer \_timer = new();

public MainWindow()

{

InitializeComponent();

\_timer.Interval = TimeSpan.FromSeconds(1);

\_timer.Tick += Timer\_Tick;

}

private void Timer\_Tick(object? sender, EventArgs e)

{

CurrentTimelineLabel.Content = String.Format("{0} / {1}", MediaElement.Position.ToString(@"hh\:mm\:ss"), MediaElement.NaturalDuration.TimeSpan.ToString(@"hh\:mm\:ss"));

}

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

{

OpenFileDialog dialog = new();

if (dialog.ShowDialog() == true)

{

MediaElement.Source = new Uri(dialog.FileName);

}

}

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

{

MediaElement.Pause();

\_timer.Stop();

}

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

{

MediaElement.Play();

\_timer.Start();

}

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

{

MediaElement.Stop();

\_timer.Stop();

}

}

**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?

Таймер можно подключить через класс DispatcherTimer.

2.2 Как создать обработчик события для таймера в приложении WPF?

Обработчик создается через событие Tick, которое вызывается при каждом тике таймера.

2.3 Как изменить интервал таймера в приложении WPF?

Через свойство Interval класса DispatcherTimer.

**3 Вывод**

3.1 Я изучил 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 комплекс программных и языковых средств, который отвечает за хранение и управление информацией в базах данных.

2.2 Что такое «база данных»?

База данных - это организованная коллекция данных, хранящаяся на носителе информации.

2.3 Какие СУБД являются реляционными (привести примеры ПО)?

Реляционные СУБД: PostgreSQL, Microsoft SQL Server, SQLite, MySQL

**3 Вывод**

3.1 Был изучен 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 - это класс для создания соединения с базой данных MS SQL Server.

2.2 Как считать из БД одно значение?

Одно значение можно считать через команду ExecuteScalar().

2.3 Каково назначение элемента SqlDataReader?

SqlDataReader - это класс для чтения данных из базы данных построчно.

2.4 Какие пространства имен требуется подключить для реализации подключения к СУБД MS SQL Server?

System.Data.SqlClient.

**3 Вывод**

3.1 Я научился 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?

В SqlCommand нужно написать SQL-команду, которую надо исполнить, и надо передать подключение к БД, а после вызвать один из Execute-методов.

2.2 Как передать параметры в SqlCommand?

Можно использовать SqlParameter или писать передавать их в SQL-команде.

2.3 Для чего используются параметры в в SqlCommand?

Параметры защищают от SQL-инъекций.

2.4 Как изменить данные в БД, используя SqlDataAdapter?

Через методы Fill для передачи таблицы, и методы Update.

2.5 Для чего применяется компонент SqlCommandBuilder?

SqlCommandBuilder применяется для автоматической генерации SQL-инструкций INSERT, UPDATE и DELETE на основе запроса SELECT для одной таблицы базы данных.

2.6 Как связать SqlCommandBuilder и SqlDataAdapter?

Нужно передать экземпляр SqlDataAdapter в конструктор класса SqlCommandBuilder.

**3 Вывод**

3.1 Я научился создавать приложения для чтения данных из БД..

**Лабораторная работа №47**

**Создание запросов к БД**

**1 Цель работы**

* 1. Научиться выполнять запросы к БД из клиентского приложения.
  2. Научиться передавать параметры в запросы.

**2 Контрольные вопросы**

2.1 Для чего используются классы репозитория при работе с БД?

Репозитории абстрагируют доступ к данным и инкапсулируют логику работы с БД.

2.2 Для чего используются классы сервиса при работе с БД?

Сервисы содержат бизнес-логику приложения и координируют работу с репозиториями.

2.3 Для чего используются ORM при работе с БД?

ORM - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

2.4 Какие свойства можно указать у параметра?

Обязательно ли передавать параметр, размер параметра, его тип.

**3 Вывод**

3.1 Я смог научиться выполнять запросы к БД из клиентского приложения.

3.2 Я смог научиться передавать параметры в запросы.

**Лабораторная работа №48**

**Создание сервисов**

**1 Цель работы**

* 1. Научиться выполнять запросы к БД из клиентского приложения.
  2. Научиться передавать параметры в запросы.

**2 Контрольные вопросы**

2.1 Для чего используются классы репозитория при работе с БД?

Репозитории абстрагируют доступ к данным и инкапсулируют логику работы с БД.

2.2 Для чего используются классы сервиса при работе с БД?

Сервисы содержат бизнес-логику приложения и координируют работу с репозиториями.

2.3 Для чего используются ORM при работе с БД?

ORM - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

2.4 Какие свойства можно указать у параметра?

Обязательно ли передавать параметр, размер параметра, его тип.

**3 Вывод**

3.1 Я научился выполнять запросы к БД из клиентского приложения.

3.2 Я научился передавать параметры в запросы.

**Практическая работа №1**

**Разработка приложения с использованием элементов отображения списков**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 В чем отличие между классами Directory и DirectoryInfo?

Directory — статический класс, который предоставляет статические методы для работы с каталогами.

DirectoryInfo — класс, который предоставляет методы для работы с каталогами.

2.2 В чем отличие между классами File и FileInfo?

File — статический класс, который предоставляет методы для выполнения операций с файлами.

FileInfo — класс, который сохраняет состояние файла.

2.3 Как получить список файлов и папок определенного каталога?

GetFiles — возвращает список файлов в каталоге;

2.4 Какие свойства класса FileInfo позволяют получить информацию о файле?

Directory, Extension, Name, Exists.

**3 Вывод**

3.1 Я научился применять классы для работы с файлами в приложениях на C#.

**4 Ход работы**

4.1 Чтение текста из файла (File.ReadAllText)

Разработать приложение, запрашивающее у пользователя имя файла.

Если файл с указанным именем не существует, сообщить об этом пользователю.

Если файл с указанным именем существует, вывести его содержимое на экран.

Console.Write("Введите имя файла: ");

string path = Console.ReadLine();

if (!File.Exists(path))

Console.WriteLine("Файла с данным именем нет.");

else

Console.WriteLine(File.ReadAllText(path));

4.2 Запись текста в файл (File.AppendAllText)

Разработать приложение, запрашивающее у пользователя имя файла.

Если файл не существует, сообщить, что файл с указанным названием будет создан.

Если файл существует, сообщить, что файл открыт на дозапись.

После вывода любого из сообщений реализовать дозапись в конец файла введенных

с консоли строк, пока пользователь не ввел строку end. end в файл не записывать.

using System.Text;

Console.Write("Введите имя файла: ");

string path = Console.ReadLine();

if (File.Exists(path))

Console.WriteLine("Файл с данным именем существет и он будет открыт на дозапись.");

else

Console.WriteLine("Файла с данным именем не существует и поэтому он будет создан.");

Console.WriteLine("Вводите содержание файла:");

StringBuilder builder = new();

string line;

while(true)

{

line = Console.ReadLine();

if (line == "end")

break;

builder.AppendLine(line);

}

File.AppendAllText(path, builder.ToString());

4.3 Построчное чтение файла (File.ReadAllLines) Разработать приложение, в котором пользователь указывает текст для поиска и имя текстового файла. Если файл существует, выполнить построчное чтение выбранного файла и вывести те строки файла, в которых содержится указанный пользователем текст, с указанием их номеров в файле.

Console.Write("Введите имя файла: ");

var fileName = Console.ReadLine();

if (!File.Exists(fileName))

{

Console.WriteLine("Файла с данным именем нет.");

return;

}

Console.Write("Введите искомый текст: ");

var searchText = Console.ReadLine();

var text = File.ReadAllLines(fileName);

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

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

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

4.4 Поиск и вывод списка файлов на экран (GetFiles) Разработать приложение, запрашивающее у пользователя имя папки и часть имени файла. Вывести на экран список полных имен и размеров файлов из указанной папки и ее подпапок, в которых имя файла содержит указанный пользователем текст.

Console.Write("Введите путь до папки: ");

var path = Console.ReadLine();

if (!Directory.Exists(path))

{

Console.WriteLine("Данной папки не существует.");

return;

}

Console.Write("Введите текст: ");

var searchFileName = Console.ReadLine() ?? string.Empty;

var fileNames = Directory.GetFiles(path, "\*", SearchOption.AllDirectories);

foreach (var fileName in fileNames)

if (fileName.Contains(searchFileName, StringComparison.OrdinalIgnoreCase))

{

FileInfo fileInfo = new(fileName);

Console.WriteLine($"Полное имя файла: {fileInfo.FullName} | Размер файла: {fileInfo.Length} Б");

}

4.5 Сортировкафайлов по папкам (CreateDirectory/Create и Move/MoveTo) Разработать приложение, запрашивающее у пользователя имя папки (например, использовать скопированную папку с заданиями по МДК.01.01). Переместить файлы из исходной папки в подпапку год\месяц\день в зависимости от даты изменения файла. Например: для файла ЛР №1.pdf с датой изменения 20.01.2023 новое расположение будет: папка\2023\01\20\ЛР №1.pdf. Перед перемещением проверить, что папка существует. Если не существует – программно создать ее.

Console.Write("Введите путь до папки: ");

var path = Console.ReadLine();

if (!Directory.Exists(path))

{

Console.WriteLine("Данной папки не существует.");

return;

}

var files = Directory.GetFiles(path);

foreach (var file in files)

{

FileInfo fileInfo = new(file);

Directory.CreateDirectory($@"{path}\{fileInfo.CreationTime.ToString().Replace(":", "\_")}");

fileInfo.MoveTo($@"{path}\{fileInfo.LastAccessTime.ToString().Replace(":", "\_")}\{fileInfo.Name}", true);

}

**Практическая работа №2**

**Работа с табличными файлами**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Какое пространство имен требуется подключить для работы с Excel?

Microsoft.Office.Interop.Excel

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

Чтобы создать объект типа «приложение Excel» в программе на C#, нужно использовать класс Excel.Application из библиотеки Microsoft.Office.Interop.Excel.

2.3 Что такое Workbooks?

* Workbooks — это коллекция объектов Workbook, которая представляет открытые рабочие книги в приложении Excel.

2.4 Что такое Worksheets?

Worksheets — это коллекция объектов, представляющих рабочие листы в файле Excel.

2.5 Что такое Range?

Range — это оператор, который возвращает поддиапазон последовательности.

2.6 Что такое Cells?

Cells— это коллекция ячеек из объекта DataGridRowCollection в компоненте DataGridView.

2.7 Как получить доступ к значению ячейки и диапазона?

В C# получить доступ к значению ячейки и диапазону в Excel можно с помощью объектов Cell и Range

**3 Вывод**

3.1 Я научился выполнять создание и редактирование табличных документов на C#.

**4 Ход работы**

4.1 Работа с файлами формата csv

CSV (от англ. Comma-Separated Values — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных. Строка таблицы соответствует строке текста, которая содержит один или несколько

столбцов, разделенных определенным символом (запятой, точкой с запятой, табуляцией, пробелом и т.д.).Файлы формата csv могут открываться в табличных редакторах (MS Excel, LibreOffice Calc и т.д.).

Разработать оконное приложение для регистрации пользователя, в котором пользователь указывает в полях ввода логин и пароль.

После нажатия кнопки «Зарегистрироваться», если введенный логин имеется в файле logins.csv, то запрашивать у пользователя повторный ввод логина.

После указания логина, не существующего в файле logins.csv, дописать введенные логин, пароль и дату регистрации (текущую) в конец файла и сообщить пользователю, что он зарегистрирован.

Данные в файле должны храниться в виде: логин;пароль;дата регистрации

Для чтения логина нужно построчно считывать данные и разделять их с помощью

Split с разделителем «;». Логин – в нулевой ячейке полученного массива.

public partial class Registration : Form

{

private const string LoginFileName = "logins.csv";

public Registration()

{

InitializeComponent();

}

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

{

RegisterUser();

}

private void RegisterUser()

{

var errors = CheckTextBoxs();

if (errors != string.Empty)

{

MessageBox.Show(errors, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (File.Exists(LoginFileName))

{

try

{

CheckLogin();

}

catch (Exception ex)

{

MessageBox.Show("Пользователь с таким логином уже существет.",

"Отказ регистрации",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

LoginTextBox.Text = string.Empty;

PasswordTextBox.Text = string.Empty;

return;

}

}

WriteUser();

MessageBox.Show("Поздравляем вы зарегестрировали нового пользователя.",

"Ура", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoginTextBox.Text = string.Empty;

PasswordTextBox.Text = string.Empty;

}

private string CheckTextBoxs()

{

StringBuilder builder = new();

if (string.IsNullOrWhiteSpace(LoginTextBox.Text))

builder.AppendLine("Поле 'Логин' не должно быть пустым или состоять из пробелов.");

if (string.IsNullOrWhiteSpace(PasswordTextBox.Text))

builder.AppendLine("Поле 'Пароль' не должно быть пустым или состоять из пробелов.");

return builder.ToString();

}

private void WriteUser()

{

using (StreamWriter writer = new(LoginFileName, true))

{

writer.WriteLine($"{LoginTextBox.Text};{PasswordTextBox.Text};{DateTime.Now}");

}

}

private void CheckLogin()

{

using (StreamReader reader = new(LoginFileName))

{

string line;

while ((line = reader.ReadLine()) != null)

if (line.Split(";")[0] == LoginTextBox.Text)

throw new Exception("Пользователь с данным именем существет.");

}

}

}

4.2 Создание документа Excel на основе шаблона Создать приложение, в котором должна создаваться книга Excel на основе существующей книги из двух листов с настроенным форматированием (текст заголовков, цвет ячеек, границы). Первый лист – с подписью «Файлы», второй лист – с подписью «Подпапки». На первом листе должны отображаться 3 столбца с заголовками: - номер файла - имя файла - размер файла На втором листе должны отображаться 2 столбца с заголовками: - номер подпапки - имя подпапки В книгу нужно записать информацию о файлах и подпапках указанного пользователем каталога (способ указания выбрать самостоятельно).

using Excel = Microsoft.Office.Interop.Excel;

Console.Write("Напишите путь до папки: ");

string path = Console.ReadLine();

if (!Directory.Exists(path))

{

Console.WriteLine("Данной папки не существует.");

return;

}

DirectoryInfo directory = new(path);

var files = directory.GetFiles();

const string Template = "template.xlsx";

const string FolderContents = "Folder contents.xlsx";

var excelApp = new Excel.Application();

var workbook = excelApp.Workbooks.Add(Template);

var workSheet = (Excel.Worksheet)workbook.Worksheets[1];

for (int i = 2; i <= files.Length; i++)

{

var file = files[i - 2];

workSheet.Cells[i, 1] = i - 1;

workSheet.Cells[i, 2] = file.Name;

workSheet.Cells[i, 3] = file.Length;

}

var dirictories = directory.GetDirectories();

workSheet = (Excel.Worksheet)workbook.Worksheets[2];

for (int i = 2; i <= dirictories.Length; i++)

{

directory = dirictories[i - 2];

workSheet.Cells[i, 1] = i - 1;

workSheet.Cells[i, 2] = directory.Name;

}

workbook.SaveAs(FolderContents);

excelApp.Quit();

4.3 Программное создание документа Excel Создать оконное приложение, в котором при нажатии на кнопку «Таблица умножения» должна создаваться книга Excel (строки и столбцы нумеруются от 2 до 9, в ячейках на пересечении – результат умножения). Лист должен быть с подписью «Умножение».

4.4 Программное форматирование документа Excel (текст и объединение ячеек) Выполнить программное форматирование таблицы умножения (в п.3.3): Таблица должна отображаться с ячейки D10. Над таблицей должен быть заголовок (Объединенные ячейки над всей таблицей) с текстом «Таблица умножения», написанным полужирным и курсивом. Размер текста ячеек – 15, заголовка – 20. Выравнивание заголовка – по центру.

public partial class Multiplication : Form

{

public Multiplication()

{

InitializeComponent();

}

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

{

GenerateMultiplicationTable();

}

private static void GenerateMultiplicationTable()

{

var application = new Excel.Application();

var workbook = application.Workbooks.Add();

workbook.Worksheets.Add();

var workSheet = (Excel.Worksheet)workbook.Worksheets[0];

workSheet.Name = "Умножение";

for (int i = 2; i < 10; i++)

workSheet.Cells[1, i] = i;

for (int i = 2; i < 10; i++)

workSheet.Cells[i, 1] = i;

for (int i = 2; i < 10; i++)

for (int j = 2; j < 10; j++)

workSheet.Cells[i, j] = i \* j;

var renge = workSheet.Range[workSheet.Cells[2, 2], workSheet.Cells[9, 9]].Cut();

workbook.SaveAs("Таблица умножения.xlsx");

application.Quit();

}

}

**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 имен требуется подключить для работы с Word?

Microsoft.Office.Interop.Word

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

Чтобы создать объект типа «приложение Word» в программе на C#, нужно использовать класс Word.Application из библиотеки Microsoft.Office.Interop.Word.

2.3 Что такое Documents?

* Documents — это набор объектов, который представляет все открытые документы приложения Word.

2.4 Что такое Range?

Range — это оператор, который возвращает поддиапазон последовательности.

2.5 Что такое Selection?

Selection — объект, который представляет текущий выделенный фрагмент текста в документе.

2.6 Что такое Paragraphes?

Paragraphs— абзац текста в документе Word.

2.7 Что такое Tables?

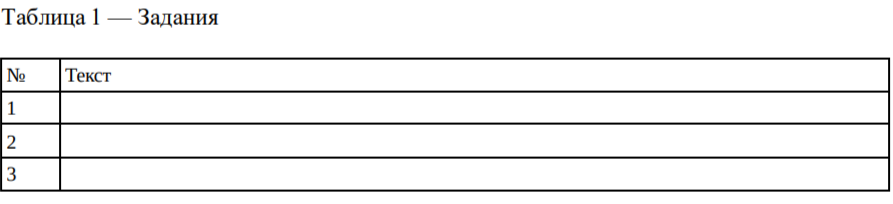
Tables — коллекция объектов, представляющих таблицы в выделенном фрагменте, диапазоне или документе.e

**3 Вывод**

3.1 Я научился выполнять создание и редактирование текстовых документов на C#.

**4 Ход работы**

4.1 Программное создание документа Word по шаблону 3.1.1 Создать оконное приложение с многострочным полем ввода и полем ввода количества заданий, в котором при нажатии на кнопку «Создать документ по шаблону» должен создаваться документ Word на основе файла Шаблон.docx: - с текстом, полученным из поля ввода, - с датой и временем печати в конце документа, - с количеством пустых нумерованных строк в таблице, равным указанному пользователем количеству заданий. Таблица состоит из n+1 строк и 2 столбцов (n = количество заданий, указанное пользователем). При реализации использовать поиск и замену строк с датой и временем и текстом из поля ввода. Документ Word должен отображаться при создании. Образец таблицы (при n=3):



3.2 Сохранение документа После создания документа реализовать его сохранение в формате docx или pdf (указывается пользователем). Для указания имени сохраняемого файла можно использовать SaveFileDialog с настроенным фильтром на форматы файлов.

internal class DocumentCreator

{

public static void CreateDocument(string replaceText, int taskCount)

{

var wordApp = new Word.Application();

wordApp.Visible = true;

var document = wordApp.Documents.Add(Environment.CurrentDirectory + "\\Шаблон.docx");

document.Content.Find.Execute(FindText: "ТекстИзПоляВвода", ReplaceWith: replaceText,

Replace: Word.WdReplace.wdReplaceOne);

var table = document.Tables[1];

if (taskCount > 0)

{

table.Cell(2, 1).Range.Text = 1.ToString();

for (int i = 3; i <= taskCount + 1; i++)

{

table.Rows.Add();

table.Cell(i, 1).Range.Text = (i - 1).ToString();

}

}

else

{

table.Rows[2].Delete();

}

document.Content.Find.Execute(FindText: "дд.мм.гггг чч:мм", ReplaceWith: DateTime.Now,

Replace: Word.WdReplace.wdReplaceOne);

SaveFileDialog saveFileDialog = new();

saveFileDialog.Filter = "docx|\*.docx|pdf|\*.pdf";

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

{

if (saveFileDialog.FilterIndex == 1)

{

document.SaveAs2(saveFileDialog.FileName);

}

else

{

document.SaveAs2(saveFileDialog.FileName, Word.WdSaveFormat.wdFormatPDF);

}

}

wordApp.Quit(Word.WdSaveOptions.wdDoNotSaveChanges);

}

public partial class TemplateFillForm : Form

{

public TemplateFillForm()

{

InitializeComponent();

}

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

{

DocumentCreator.CreateDocument(DiscriptionTextBox.Text, int.Parse(TaskCountTextBox.Text));

}

}}

3.3 Программное создание документа Word

3.3.1 Создать оконное приложение с многострочным полем ввода, в котором при нажатии на кнопку «Создать новый документ» должен создаваться документ Word с текстом, полученным из поля ввода.

Документ Word должен отображаться при создании.

3.3.2 Выполнить программное форматирование документа:

- шрифт документа: Times New Roman

- текст выравнивается по ширине,

- размер текста: 14.

internal class DocumentCreator

{

public static void CreateDocument(string text)

{

var wordApp = new Word.Application();

wordApp.Visible = true;

var document = wordApp.Documents.Add();

var paragraph = document.Paragraphs[1];

paragraph.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphDistribute;

paragraph.Range.Font.Size = 14;

paragraph.Range.Font.Name = "Times New Roman";

paragraph.Range.Text = text;

SaveFileDialog saveFileDialog = new();

saveFileDialog.Filter = "docx|\*.docx|pdf|\*.pdf";

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

{

if (saveFileDialog.FilterIndex == 1)

{

document.SaveAs2(saveFileDialog.FileName);

}

else

{

document.SaveAs2(saveFileDialog.FileName, Word.WdSaveFormat.wdFormatPDF);

}

}

wordApp.Quit(Word.WdSaveOptions.wdDoNotSaveChanges);

}

}

public partial class DocumentCreatorForm : Form

{

public DocumentCreatorForm()

{

InitializeComponent();

}

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

{

DocumentCreator.CreateDocument(DescriptionTextBox.Text);

}

}

**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 — это механизм, который позволяет связывать данные между элементами пользовательского интерфейса.

2.2 Каков шаблон настройки привязки данных?

Шаблон настройки привязки данных в C# для элементов включает использование класса System.Windows.Data.Binding и указание свойств элемента-источника и элемента-приемника.

2.3 Какой интерфейс надо реализовать для создания конвертера значений?

* Для создания конвертера значений в C# можно реализовать интерфейс IValueConverter

2.4 Какой интерфейс надо реализовать для валидации данных при привязке?

Для валидации данных при привязке можно реализовать интерфейс IdataErrorInfo.

**3 Вывод**

3.1 Я научился выполнять привязку данных в приложении WPF.

**4 Ход работы**

4.1 Создать в приложении набор объектов Game. Реализовать отображение данных из файла Games.txt в табличном виде в ListView или DataGrid, используя привязку.

4.2 Реализовать отображение значений выбранного объекта вне таблицы в полях ввода и изменение значений (все кроме категории).

4.3 Реализовать возможность смены категории выбранного объекта вне таблицы, добавив отображение списка категорий игр в выпадающем списке и связав выбранный в списке элемент с категорией игры выбранного объекта. При изменении категории в выпадающем списке она должна меняться у выбранного объекта.

4.4 Добавить конвертеры для отображения данных в табличном виде: - для отображения старой цены в зачеркнутом виде, если у цены есть скидка, и новой цены (с примененной скидкой), - для отображения разным цветом строк с разными категориями игр (если категория не указана, то цвет белый).

4.5 Добавить валидацию данных в полях ввода выбранной игры с выводом всплывающих подсказок для следующих значений: - скидку (от 1 до 30), - название (не меньше двух букв), - цену (не может быть отрицательной).

public class Game

{

public int IdGame { get; set; }

public string Name { get; set; }

public string? Description { get; set; }

public string? Category { get; set; }

public double Price { get; set; }

public string PriceInfo { get

{

if (Sale > 0)

return $"~~{Price}~~₽ {Price \* (Sale / 100)}₽";

return $"{Price}₽";

}

}

public double Sale { get; set; } = 0;

}

public class Category

{

public string Name { get; set; }

public string Description { get; set; }

}

<StackPanel>

<DataGrid x:Name="GamesListView" ItemsSource="{Binding Games}" Selected="GamesListView\_Selected">

<DataGrid.ItemTemplate>

<DataTemplate DataType="local:Game"/>

</DataGrid.ItemTemplate>

<DataGrid.Columns>

<DataGridTextColumn Header="Id" Binding="{Binding IdGame}"/>

<DataGridTextColumn Header="Name" Binding="{Binding Name}"/>

<DataGridTextColumn Header="Description" Binding="{Binding Description}"/>

<DataGridTextColumn Header="Category" Binding="{Binding Category}"/>

<DataGridTextColumn Header="Price"/>

<DataGridTextColumn Header="Sale" Binding="{Binding Sale}"/>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal">

<TextBox Text="{Binding SelectedGame.Id}"/>

<TextBox Text="{Binding ElementName=GamesListView, Path=SelectedItem.Name}"/>

<TextBox Text="{Binding ElementName=GamesListView, Path=SelectedItem.Description}"/>

<ComboBox SelectedItem="{Binding ElementName=GamesListView, Path=SelectedItem.Category}" ItemsSource="{Binding ElementName=CategoriesListView, Path=Items}" DisplayMemberPath="Name"/>

<TextBox Text="{Binding ElementName=GamesListView, Path=SelectedItem.Price}"/>

<TextBox Text="{Binding ElementName=GamesListView, Path=SelectedItem.Sale}"/>

</StackPanel>

<DataGrid x:Name="CategoriesListView" ItemsSource="{Binding Categories}">

<DataGrid.ItemTemplate>

<DataTemplate DataType="local:Category"/>

</DataGrid.ItemTemplate>

</DataGrid>

</StackPanel>

public partial class MainWindow : Window

{

public List<Game> Games { get; set; }

public Game SelectedGame { get; set; }

public List<Category> Categories { get; set; }

public MainWindow()

{

InitializeComponent();

Games = new()

{

new Game{IdGame = 1, Name = "Tetris", Category = "головоломка", Price = 150},

new Game{IdGame = 2, Name = "Flappy Bird", Description = "игра про летучую птицу", Category = "платформер", Price = 10},

new Game{IdGame = 3, Name = "Pac-man", Description = "игра про колобка", Category = "аркада", Price = 300},

new Game{IdGame = 4, Name = "Arkanoid", Category = "аркада", Price = 400},

new Game{IdGame = 5, Name = "Mario", Description = "игра про Марио", Category = "платформер", Price = 1000},

new Game{IdGame = 6, Name = "Tetris2", Price = 150},

new Game{IdGame = 7, Name = "Flappy Bird2", Description = "игра про летучую птицу", Category = "платформер", Price = 10},

new Game{IdGame = 8, Name = "Pac-man2", Description = "игра про колобка", Price = 300},

new Game{IdGame = 9, Name = "Arkanoid2", Category = "аркада", Price = 400},

new Game{IdGame = 10, Name = "Mario2", Description = "игра про Марио", Price = 1000},

};

Categories = new()

{

new Category{Name = "головоломка", Description = "описание жанра головоломка"},

new Category{Name = "платформер", Description = "описание жанра платформер"},

new Category{Name = "RPG", Description = "описание жанра RPG"},

new Category{Name = "аркада", Description = "описание жанра аркада"}

};

GamesListView.DataContext = this;

CategoriesListView.DataContext = this;

}

private void GamesListView\_Selected(object sender, RoutedEventArgs e)

{

SelectedGame = (Game)sender;

}

}

**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#?

В приложении на C# настройки можно добавить с помощью файла Settings.settings или создать собственные файлы настроек.

2.2Как программно считать значение параметра из настроек?

Программно считать значение параметра из настроек можно с помощью пространства имен Properties.Settings.

2.3 Как программно изменить значение параметра в настройках?

* Значения настроек доступны через свойства объекта Properties.Settings.Default.

2.4 Как выполнить сохранение значений параметров в настройках?

Чтобы сохранить измененные настройки, нужно вызвать метод Save().

**3 Вывод**

3.1 Я научился 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 применяется для задания представления и визуального поведения элементов управления в приложениях

2.2 Где может быть описан ControlTemplate?

* В корневом элементе XAML-файла определения приложения. В этом случае шаблон можно будет использовать в любом месте приложения.

2.3 Для чего применяется UserControl?

* UserControl применяется для создания пользовательских элементов управления.

2.4 Каков алгоритм создания пользовательского элемента управления?

* 1. Создать новый проект Библиотека пользовательских классов WPF.
* 2. Щелкнуть правой кнопкой по решению и выбрать «Добавить» → «Новый элемент...».
* 3. Выбрать «User Control (WPF)» и назвать его, например, MyUserControl.
* 4. Нажать кнопку «Добавить» — в решение будут добавлены два новых файла: MyUserControl.xaml и MyUserControl.cs.
* 5. В файле MyUserControl.xaml создать элемент управления, добавив нужные компоненты (кнопку, текстовое поле) и настроив их свойства.
* 6. В файле MyUserControl.cs реализовать логику элемента, например, добавить обработчик событий.

2.5 Как программно создать обработчик события?

Создать метод, сигнатура которого соответствует нужному типу

Добавить этот метод в список обработчиков нужного события с помощью оператора +=.

2.6 Для чего используется свойство зависимости?

Свойства зависимости в WPF используются для того, чтобы вычислить значение свойства на основе других входных данных.

**3 Вывод**

3.1 Я смог научится создавать пользовательские элементы управления в приложении WPF.