**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

# Отчеты по лабораторным и практическим работам

**по МДК.01.01**

Студент: Чугин Е.В

Группа: ИСПП-21

Преподаватель: Ю.С. Маломан

Архангельск 2024

# Лабораторная работа №1

**Оценка сложности алгоритмов сортировки**

1. **Цель работы**
   1. Научиться реализовывать и оценивать сложность алгоритмов сортировки массивов на C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «массив»?

Ответ: Массив – это структура однотипных элементов, занимающих непрерывную область памяти.

* 1. Как описывается одномерный массив?

Ответ: Одномерный массив описывается указанием типа данных его элементов и их количество.

* 1. Как обратиться к некоторому элементу одномерного массива?

Ответ: Для обращения к некоторому элементу одномерного массива используется индексация, где каждый элемент имеет свой уникальный индекс начиная с 0.

* 1. Как можно задать одномерный массив?

Ответ: Одномерный массив можно задать явно, перечислив его элементы в фигурных скобках и разделяя их запятыми, например int[] array = {1, 2, 3, 4, 5}

* 1. Что такое «сортировка»?

Ответ: Сортировка – это процесс упорядочивания элементов в массиве по определенному критерию.

* 1. Что такое «алгоритм сортировки»?

Ответ: Алгоритм сортировки – это конкретный способ реализации сортировки элементов в массиве.

* 1. Какие виды сортировки массивов существуют?

Ответ: - Сортировка пузырьком (Bubble Sort), Сортировка выбором (Selection Sort), Сортировка вставками (Insertion Sort), Быстрая сортировка (Quick Sort), Сортировка слиянием (Merge Sort), Пирамидальная сортировка (Heap Sort), Сортировка подсчетом (Counting Sort), Сортировка Шелла (Shell Sort)

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился реализовывать и оценивать сложность алгоритмов сортировки массивов на C#.

# Лабораторная работа №2

**Оценка сложности алгоритмов поиска**

1. **Цель работы**
   1. Научиться реализовывать и оценивать сложность алгоритмов поиска элементов массивов на C#.
2. **Ответы на контрольные вопросы**
   1. Какие виды поиска элементов массивов существуют?

Ответ: Алгоритм сортировки – это конкретный способ реализации сортировки элементов в массиве.

* 1. В чем особенность алгоритма линейного поиска и какова его временная сложность?

Ответ: - Линейный поиск

Двоичный (бинарный) поиск, Поиск прыжками (jump search), Интерполяционный поиск, Фибоначчиев поиск.

* 1. В чем особенность алгоритма двоичного поиска и какова его временная сложность?

Ответ: Особенность алгоритма линейного поиска заключается в том, что элементы массива перебираются поочередно, начиная с первого, пока не будет найден искомый элемент или пока не будет достигнут конец массива. Временная сложность линейного поиска составляет O(n), где n - количество элементов в массиве.

* 1. В чем особенность алгоритма поиска прыжками и какова его временная сложность?

Ответ: Алгоритм двоичного поиска предполагает, что массив отсортирован, и работает путем деления массива пополам и сравнения искомого элемента с элементом в середине массива. Если искомый элемент меньше среднего, то поиск продолжается в левой половине массива, иначе - в правой. Временная сложность двоичного поиска составляет O(log n), где n - количество элементов в отсортированном массиве.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился реализовывать и оценивать сложность алгоритмов поиска элементов массивов на C#.

# Лабораторная работа №3

**Оценка сложности рекурсивных алгоритмов**

1. **Цель работы**
   1. Научиться разрабатывать и оценивать сложность рекурсивных функций в программах на C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «рекурсия»?

Ответ: Рекурсия – это процесс, при котором функция вызывает саму себя во время выполнения.

* 1. Какие проблемы могут возникать при реализации рекурсивных алгоритмов на электронных вычислительных машинах?

Ответ: При реализации рекурсивных алгоритмов на электронных вычислительных машинах могут возникать проблемы с использованием большого объема памяти из-за сохранения промежуточных результатов и вызовов функций. Также возможны проблемы с производительностью из-за накладных расходов на вызовы функций.

* 1. Какое определение функции может быть названо рекурсивным? Привести примеры.

Ответ: Определение функции называется рекурсивным, если в определении функции используется сама эта функция. Примеры:

- Факториал числа n: factorial(n) = n \* factorial(n-1), где factorial(n) — это рекурсивное определение.

- Вычисление чисел Фибоначчи: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2), где fibonacci(n) - также рекурсивное определение.

* 1. Что такое «глубина рекурсии»?

Ответ: Глубина рекурсии - это количество уровней вложенных вызовов функции, которые происходят во время выполнения рекурсивного алгоритма.

* 1. Что такое «рекурсивный спуск»?

Ответ: Рекурсивный спуск - это метод решения задачи, при котором функция разбивает задачу на более мелкие подзадачи и вызывает себя для их решения.

* 1. Что такое «рекурсивный подъём»?

Ответ: Рекурсивный подъем - это метод, противоположный рекурсивному спуску, при котором функция комбинирует результаты подзадач для получения ответа на более общую задачу.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился разрабатывать и оценивать сложность рекурсивных функций в программах на C#.

# Лабораторная работа №4

**Оценка сложности эвристических алгоритмов**

1. **Цель работы**
   1. Научиться реализовывать и оценивать сложность эвристических алгоритмов на C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «многомерный массив»?

Ответ: Многомерный массив - это массив, элементами которого также являются массивы. Например, двумерный массив - это массив, элементами которого являются другие массивы.

* 1. Как описывается двумерный массив?

Ответ: Двумерный массив описывается как массив массивов. int[,] array = new int[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

* 1. Как обратиться к некоторому элементу двумерного массива?

Ответ: Для доступа к элементу двумерного массива необходимо указать индексы строки и столбца, например array[1][2] обращается к элементу во второй строке и третьем столбце.

* 1. Как узнать количество строк двумерного массива?

Ответ: Количество строк двумерного массива можно узнать с помощью array.length, где array - сам массив.

* 1. Как узнать количество столбцов двумерного массива?

Ответ: Количество столбцов двумерного массива можно узнать, обратившись к длине любой строки в массиве. Например, array[0].length вернет количество столбцов в первой строке.

* 1. Как вывести двумерный массив на консоль в виде таблицы?

Ответ: int[,] array = new int[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

Console.Write(array[i, j] + "\t");

}

Console.WriteLine();

}

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился реализовывать и оценивать сложность эвристических алгоритмов на C#.

# Лабораторная работа №5

**Работа с классами**

1. **Цель работы**
   1. Изучить процесс разработки и применения классов на языке C#,
   2. Изучить реализацию механизма инкапсуляции на языке C#.
2. **Ответы на контрольные вопросы**
   1. Какова общая форма объявления класса в C#?

Ответ:

class ClassName

{

// Код

}

* 1. На какие виды делятся данные класса в C#?

Ответ: Поля (fields) - переменные, которые хранят данные внутри класса. Свойства (properties) - специальные методы доступа к полям, позволяющие контролировать доступ к данным. Индексаторы (indexers) - позволяют обращаться к объекту класса как к массиву. События (events) - позволяют реализовать механизм обработки событий.

* 1. На какие виды делятся функции класса в C#?

Ответ: Методы экземпляра - методы, которые принадлежат конкретному объекту класса. Статические методы - методы, которые принадлежат самому классу, а не его экземплярам.

* 1. Для чего применяются конструкторы классов?

Ответ: Конструкторы классов используются для инициализации новых экземпляров классов. Они выполняются при создании объекта и могут устанавливать начальные значения для полей объекта.

* 1. Что такое «цепочка конструкторов»?

Ответ: "Цепочка конструкторов" - это процесс вызова одного конструктора из другого конструктора в том же классе или в родительском классе.

* 1. Для чего применяются свойства классов?

Ответ Свойства классов позволяют контролировать доступ к полям класса, обеспечивая инкапсуляцию данных и обеспечивая безопасность и надежность кода.

* 1. Когда используются автоматически реализуемые свойства классов?

Ответ: Автоматически реализуемые свойства классов используются, когда нет необходимости в дополнительной логике доступа к полям. Они позволяют определить свойство всего в одной строке кода.

* 1. В чем отличие вызова статические членов классов от членов экземпляра класса?

Ответ: Вызов статических членов классов осуществляется через имя класса, а вызов членов экземпляра класса требует создания объекта класса и использования этого объекта для доступа к членам.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс разработки и применения классов на языке C#,
   2. В ходе проделанной лабораторной работы я изучил реализацию механизма инкапсуляции на языке C#.

# Лабораторная работа №6

**Перегрузка методов**

1. **Цель работы**
   1. Изучить процесс выполнения перегрузки методов на языке C#;
   2. Изучить различные способы передачи параметров в метод.
2. **Ответы на контрольные вопросы**
   1. Что такое «перегрузка методов»?

Ответ: Перегрузка методов - это возможность определить несколько методов с одинаковым именем в классе, но с разными параметрами.

* 1. К какому виду полиморфизма относится перегрузка методов?

Ответ: Перегрузка методов относится к статическому (compile-time) полиморфизму.

* 1. Какие типы функций класса можно перегружать?

Ответ: В классе можно перегружать как обычные методы, так и конструкторы.

* 1. Какие требования предъявляются к сигнатуре перегружаемых функций класса?

Ответ: Сигнатура перегружаемых функций должна отличаться по количеству или типу параметров, либо по их порядку.

* 1. Какие типы функций класса нельзя перегружать?

Ответ: Нельзя перегружать функции класса, если их сигнатуры отличаются только по возвращаемому типу.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс выполнения перегрузки методов на языке C#;
   2. В ходе проделанной лабораторной работы я изучить различные способы передачи параметров в метод.

# Лабораторная работа №7

**Определение операций в классе**

1. **Цель работы**
   1. Изучить процесс разработки дочерних классов на языке C#;
   2. Изучить реализацию механизма наследования на языке C#.
2. **Ответы на контрольные вопросы**
   1. Какое ключевое слово определяет операторный метод?

Ответ: Ключевое слово, которое определяет операторный метод в C#, это "operator".

* 1. Какие унарные операторы можно перегружать в C#?

Ответ: Унарные операторы, которые можно перегружать в C#, включают в себя: "+", "-", "!", " ", "++", "–", "true" и "false".

* 1. Какие бинарные операторы можно перегружать в C#

Ответ: Бинарные операторы, которые можно перегружать в C#, включают в себя: "+", "-", "\*", "/"

* 1. Какие операторы сравнения можно перегружать в C#?

Ответ: Операторы сравнения, которые можно перегружать в C#, включают в себя: "==", "!=", "<", ">", "<=", ">=".

* 1. Какие операторы требуется перегружать в C# попарно?

Ответ: В C# операторы требуется перегружать попарно для бинарных операторов: "==" / "!=" и "<" / ">".

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс разработки дочерних классов на языке C#;
   2. В ходе проделанной лабораторной работы я изучил реализацию механизма наследования на языке C#.

# Лабораторная работа №8

**Создание наследованных классов**

1. **Цель работы**
   1. Изучить процесс разработки дочерних классов на языке C#;
   2. Изучить реализацию механизма наследования на языке C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «наследование»?

Ответ: Наследование - это механизм, позволяющий создавать новый класс на основе уже существующего класса (родительского), чтобы наследовать его свойства и методы.

* 1. Сколько родительских классов может быть у дочернего класса в C#?

Ответ: В C# дочерний класс может иметь только один родительский класс (одинарное наследование).

* 1. Какое ключевое слово позволяет обратиться к реализации родительского класса из дочернего?

Ответ: Ключевое слово base позволяет обратиться к реализации родительского класса из дочернего.

* 1. Что такое «переопределение метода» и как оно выполняется?

Ответ: Переопределение метода – это процесс изменении реализации метода в дочернем классе, который был унаследован от родительского класса. В C# это выполняется с помощью ключевого слова override.

* 1. Что такое «абстрактный класс»?

Ответ: Абстрактный класс – это класс, который содержит один или несколько абстрактных методов, т.е. методов без реализации. Этот класс не может быть инстанциирован, но может быть использован для создания других классов

* 1. Для чего предназначены модификаторы virtual, override, abstract, new?

Ответ:

virtual - указывает, что метод может быть переопределен в дочерних классах. override - используется для переопределения виртуального метода из базового класса в производном классе. abstract - указывает, что метод не имеет реализации и должен быть переопределен в производных классах. new - позволяет создавать новую реализацию метода в производном классе, скрывая реализацию из базового класса

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс разработки дочерних классов на языке C#;
   2. В ходе проделанной лабораторной работы я изучил реализацию механизма наследования на языке C#.

# Лабораторная работа №9

**Работа с объектами через интерфейс**

1. **Цель работы**
   1. Изучить процесс разработки и реализации интерфейсов на языке C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «интерфейс»?

Ответ: Интерфейс – это абстрактный тип данных, который определяет набор методов, свойств и событий, которые должен реализовать любой класс, который реализует этот интерфейс.

* 1. Чем отличается интерфейс от абстрактного класса?

Ответ: Основное отличие между интерфейсом и абстрактным классом заключается в том, что интерфейс может содержать только сигнатуры методов, свойств и событий, но не их реализацию, в то время как абстрактный класс может содержать как абстрактные (без реализации) методы, так и реализованные методы.

* 1. Есть ли у класса ограничения по количеству реализуемых интерфейсов?

Ответ: В C# класс не имеет ограничений по количеству реализуемых интерфейсов. Класс может реализовывать любое количество интерфейсов.

* 1. Какова общая форма объявления интерфейсов и их элементов?

Ответ: interface ИмяИнтерфейса

{  
 // Сигнатуры методов, свойств и событий без реализации

}

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс разработки и реализации интерфейсов на языке C#.

# Лабораторная работа №10

**Использование стандартных интерфейсов**

1. **Цель работы**
   1. Изучить процесс реализации стандартных интерфейсов на языке C#.
2. **Ответы на контрольные вопросы**
   1. Для чего используется интерфейс IComparable?

Ответ: Интерфейс IComparable используется для сравнения объектов на основе их значений. Классы, которые реализуют интерфейс IComparable, могут определять способ сравнения своих экземпляров, что позволяет сортировать коллекции объектов и выполнять другие операции, требующие сравнения.

* 1. Для чего используется интерфейс IEquatable?

Ответ: Интерфейс IEquatable используется для определения метода Equals, который позволяет сравнивать объекты на равенство. Реализация интерфейса IEquatable позволяет более эффективно сравнивать объекты по содержимому, чем стандартный метод Equals.

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

Ответ: Отличие между обобщенным и необобщенным интерфейсами заключается в том, что обобщенные интерфейсы позволяют указать тип данных во время объявления интерфейса, что делает интерфейс более универсальным и гибким. Необобщенные интерфейсы не предоставляют возможности указать тип данных и работают только с типом object.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс реализации стандартных интерфейсов на языке C#.

# Лабораторная работа №11

**Коллекции. Параметризованные классы**

1. **Цель работы**
   1. Изучить процесс создания и применения параметризованных классов коллекций (обобщенных списков и словарей) на языке C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «коллекция»?

Ответ: «Коллекция» — это структура данных, которая позволяет хранить и управлять группой объектов. Коллекции обеспечивают удобный способ организации и доступа к данным.

* 1. Какие классы описаны в пространстве имен System.Collections.Generic?

Ответ: Пространство имен System.Collections.Generic содержит ряд классов для работы с коллекциями в .NET Framework, включая List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T> и другие.

* 1. Что такое List<T>?

Ответ: List<T> - это класс в пространстве имен System.Collections.Generic, представляющий динамический массив, который может хранить элементы указанного типа T. List<T> обеспечивает удобные методы для добавления, удаления, поиска элементов и других операций над коллекцией.

* 1. Как можно обратиться к элементу списка?

Ответ: Для доступа к элементу списка можно использовать индексатор, указав индекс элемента в квадратных скобках. Например, myList[0] обратится к первому элементу списка myList.

* 1. Что такое Dictionary<TKey, TValue>?

Ответ: Dictionary<TKey, TValue> - это класс в пространстве имен System.Collections.Generic, представляющий коллекцию пар ключ-значение. Каждый элемент словаря представлен ключом и соответствующим значением.

* 1. Как можно обратиться к элементу словаря?

Ответ: Для доступа к элементу словаря можно использовать ключ в квадратных скобках. Например, myDictionary["key"] обратится к значению, связанному с ключом "key" в словаре myDictionary.

1. **Вывод**
   1. В ходе проделанной лабораторной работы 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. Изучить процесс создания и применения структур на языке C#.
2. **Ответы на контрольные вопросы**
   1. В какой области памяти хранятся типы значения?

Ответ: Типы значения хранятся в стеке памяти

* 1. Что такое «структура»?

Ответ: "структура" (structure) - это пользовательский тип данных, который позволяет объединить несколько переменных различных типов в одну единицу.

* 1. Чем отличается структура от класса?

Ответ: - Структуры являются типами значений, а классы - ссылочными типами. Структуры не поддерживают наследование, в то время как классы могут быть унаследованы другими классами. Структуры не могут иметь явно определенного конструктора без параметров, в отличие от классов.

* 1. Что такое «перечисление»?

Ответ: "Перечисление" (enum) в C# - это специальный тип данных, который позволяет задать именованный набор константных значений.

* 1. Для чего используются перечисления?

Ответ: Перечисления используются для удобного определения и использования набора константных значений, которые могут быть связаны с определенными именами. Например, дни недели, месяцы года и т. д.

* 1. Какова общая форма объявления перечисления в C#?

Ответ:

enum ИмяПеречисления

{

Значение1,

Значение2,  
 …

}

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс создания и применения структур на языке C#.

# Лабораторная работа №13

**Обработка и форматирование строк**

1. **Цель работы**
   1. Изучить процесс обработки строк на языке C#;
   2. Научиться применять стандартные методы классов String, StringBuilder и Char для обработки строковых и символьных данных в программах на языке C#.
2. **Ответы на Ответы на контрольные вопросы**
   1. К какому типу переменных относятся переменные типа string?

Ответ: Переменные типа string относятся к ссылочным типам данных.

* 1. Какие операции допустимы над строковыми данными?

Ответ: 1)Конкатенация (соединение) строк. 2)Использование методов для работы со строками, таких как Substring(), IndexOf(), Replace() и других. 3) Сравнение строк на равенство или неравенство. 4)Форматирование строк.

* 1. В чем отличие переменной типа string от массива символов?

Ответ: 1) Переменная типа string является неизменяемой (immutable), т.е. после создания строки ее значение нельзя изменить. 2) Массив символов позволяет изменять содержимое символ за символом.

* 1. Что такое «интерполяция строк»?

Ответ: Интерполяция строк (string interpolation) — это способ вставки значений переменных в строку, используя специальный синтаксис. В C# интерполяция строк выполняется с помощью символа $ перед строкой и вставки выражений в фигурных скобках {}.

* 1. Изменяют ли методы класса String исходную строку?

Ответ: Методы класса String в C# не изменяют исходную строку, а возвращают новую строку с результатом операции. Так как строки являются неизменяемыми, любые операции над строками создают новые строки.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс обработки строк на языке C#;
   2. В ходе проделанной лабораторной работы 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. Научиться составлять шаблоны регулярных выражений в программах на C#;
   2. Научиться применять регулярные выражения для поиска и замены подстрок в программах на языке C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «регулярное выражение»?

Ответ: Регулярное выражение – это шаблон, который используется для поиска и сопоставления текстовых данных с определенными правилами.

* 1. Для чего используются регулярные выражения?

Ответ: Регулярные выражения используются для поиска, извлечения, замены и проверки текстовой информации с определённым шаблоном.

* 1. Для чего используется класс Regex?

Ответ: Класс Regex используется для работы с регулярными выражениями, предоставляя методы для компиляции, сопоставления, поиска и замены текста по заданным шаблонам.

* 1. Каков алгоритм поиска подстроки при помощи регулярного выражения?

Ответ: Алгоритм включает в себя компиляцию шаблона, сопоставление его с текстом и нахождение всех соответствий.

* 1. Каков алгоритм замены подстроки при помощи регулярного выражения?

Ответ: Алгоритм включает в себя поиск всех соответствий шаблону в тексте и их замену на указанную строку.

* 1. Для чего в регулярных выражениях применяются escape-символы?

Ответ: Escape-символы используются для экранирования специальных символов, чтобы они были интерпретированы буквально, а не как часть шаблона.

* 1. Для чего в регулярных выражениях применяются классы символов?

Ответ: Классы символов используются для задания групп символов, которые могут быть сопоставлены с одним из этой группы.

* 1. Для чего в регулярных выражениях применяются квантификаторы?

Ответ: Квантификаторы используются для указания количества повторений символов или групп символов в тексте. Например \*, +, ? указывают на ноль или более, один или более, ноль или один повтор соответственно.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился составлять шаблоны регулярных выражений в программах на C#;
   2. В ходе проделанной лабораторной работы я научился применять регулярные выражения для поиска и замены подстрок в программах на языке C#.

# Лабораторная работа №15

**Разработка делегатов**

1. **Цель работы**
   1. Научиться разрабатывать и применять делегаты на C#
2. **Ответы на контрольные вопросы**
   1. Что такое «делегат» в C#?

Ответ: Делегат представляет собой тип данных, которые представляет ссылку на метод с определённой сигнатурой и возвращаемым значением. Делегаты позволяют передавать методы как параметры, хранить и вызывать их.

* 1. Как объявить делегат на C#?

Ответ: Для объявления делегата используется ключевое слово delegate. Например: delegate void MyDelegate(int x);

* 1. Какие способы вызова метода через делегат существуют?

Ответ: Существуют несколько способов вызова метода через делегат:

1) Использование метола Invoke(): myDelegate.Invoke(5)

2) Использование синтаксиса вызова делегата как метода: myDelegate(5);

* 1. Какие встроенные делегаты имеются в C# и для чего они используются?

Ответ:

1) Action: делегат без возвращаемого значения, которые может принимать от 0 до 16 аргументов.

2) Func: обобщённый делегат, который представляет метод с возвращаемым значением.

3) Predicate: делегат, который принимает один аргумент и возвращает булево значение.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился разрабатывать и применять делегаты на C#

# Лабораторная работа №16

**Разработка событий**

1. **Цель работы**
   1. Научиться создавать, вызывать и обрабатывать события на C#.
2. **Ответы на контрольные вопросы**
   1. Что такое «событие» в C#?

Ответ: В C# "событие" представляет собой механизм, который позволяет объектам уведомлять другие объекты о том, что произошло какое-то действие или изменение состояния.

* 1. Как объявить событие на C#?

Ответ: Для объявления события в C# используется ключевое слово event. Например: public event EventHandler MyEvent;

* 1. Как создать обработчик события?

Ответ: Чтобы создать обработчик события, нужно присоединить метод к событию. Например: MyObject.MyEvent += MyEventHandler;

* 1. Какой класс является родительским для всех классов данных события?

Ответ: Родительским классом для всех классов данных события является EventArgs. Обычно производные классы от EventArgs используются для передачи дополнительной информации о событии.

* 1. Какие классы делегатов являются стандартными для создания событий в C#?

Ответ: Стандартные классы делегатов для создания событий в C#:

1) EventHandler: делегат, который используется для обработки событий без дополнительной информации.

2) EventHandler<T>: обобщенный делегат, который позволяет передавать дополнительную информацию о событии через объект типа T.

# Лабораторная работа №17

**Операции со списками**

1. **Цель работы**
   1. Изучить процесс обработки списков стандартными методами на языке C#.У
2. **Ответы на контрольные вопросы**
   1. Что такое «LINQ»?

Ответ: LINQ (Language Integrated Query) — это набор инструментов в языке программирования C#, который позволяет выполнять запросы к данным в стиле SQL непосредственно в коде.

* 1. Что такое «анонимный тип»?

Ответ: Анонимный тип — это тип данных, который создается компилятором на основе выражения и не имеет явного объявления в коде. Он удобен для хранения временных результатов запросов LINQ.

* 1. Для чего используется var?

Ответ: Ключевое слово var используется для объявления переменной без указания ее типа явно. Компилятор самостоятельно определяет тип переменной на основе присваиваемого значения.

* 1. Для чего используется метод Select?

Ответ: Метод Select используется для проекции (отображения) данных из коллекции, например, для выбора определенных полей объектов или преобразования данных.

* 1. Для чего используются методы OrderBy, OrderByDescending?

Ответ: Методы OrderBy и OrderByDescending используются для сортировки элементов коллекции по возрастанию и убыванию соответственно.

* 1. Для чего используются методы ThenBy, ThenByDescending?

Ответ: Методы ThenBy и ThenByDescending используются для добавления дополнительных условий сортировки после основной сортировки с помощью методов OrderBy и OrderByDescending.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс обработки списков стандартными методами на языке C#.

# Лабораторная работа №18

**Проектирование диаграммы классов**

1. **Цель работы**
   1. Изучить процесс описания типов данных с применением диаграммы классов.
2. **Ответы на контрольные вопросы**
   1. Для чего используется диаграмма классов?

Ответ: Диаграмма классов используется для визуализации структуры классов в системе, их атрибутов, методов и взаимосвязей между классами.

* 1. Из каких элементов состоит описание типа в диаграмме классов?

Ответ: Описание типа в диаграмме классов состоит из: названия класса, атрибутов класса, методов класса.

* 1. На какие группы делятся взаимосвязи в диаграмме классов?

Ответ: Взаимосвязи в диаграмме классов делятся на три группы: наследование, ассоциация, зависимость.

* 1. В чем отличие между взаимосвязью «наследование» и «реализация»?

Ответ: Взаимосвязь "наследование" означает отношение между базовым классом и производным классом, где производный класс наследует свойства и методы базового класса. Взаимосвязь "реализация" означает, что класс реализует интерфейс или абстрактный класс.

* 1. В чем отличие между взаимосвязью «агрегация» и «композиция»?

Ответ: Взаимосвязь "агрегация" означает, что один объект содержит другой объект, но они могут существовать независимо. Взаимосвязь "композиция" означает более сильную связь, где один объект является частью другого и не может существовать без него.

* 1. Что показывает взаимосвязь «ассоциация»?

Ответ: Взаимосвязь "ассоциация" показывает связь между двумя классами, которая может быть однонаправленной или двунаправленной.

* 1. Какие виды мощности отношений могут быть указаны в диаграмме классов?

Ответ: В диаграмме классов мощность отношений может быть выражена с помощью кардинальности, которая указывает количество объектов одного класса, связанных с объектами другого класса.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс описания типов данных с применением диаграммы классов.

# Лабораторная работа №19

**Использование шаблонов проектирования**

1. **Цель работы**
   1. Научиться применять паттерны проектирования в разработке программ.
2. **Ответы на контрольные вопросы**
   1. Для чего используются порождающие паттерны?

Ответ: Порождающие паттерны используются для управления процессом создания объектов, обеспечивая более гибкое и эффективное создание объектов в приложении.

* 1. Какие паттерны относятся к порождающим?

Ответ: Некоторые из порождающих паттернов включают:

1) Фабричный метод (Factory Method)

2) Абстрактная фабрика (Abstract Factory)

3) Одиночка (Singleton)

4) Строитель (Builder)

5) Прототип (Prototype)

* 1. Для чего используются структурные паттерны?

Ответ: Структурные паттерны используются для организации различных классов и объектов в более сложные структуры, обеспечивая их взаимодействие и упрощая расширение функциональности

* 1. Какие паттерны относятся к структурным?

Ответ: Некоторые из структурных паттернов включают:

1) Адаптер (Adapter)

2) Мост (Bridge)

3) Компоновщик (Composite)

4) Декоратор (Decorator)

5) Фасад (Facade)

6) Прокси (Proxy)

* 1. Для чего используются поведенческие паттерны?

Ответ: Поведенческие паттерны используются для управления взаимодействием между объектами и классами, обеспечивая более гибкое и эффективное управление поведением приложения.

* 1. Какие паттерны относятся к поведенческим?

Ответ: Некоторые из поведенческих паттернов включают:

1) Цепочка обязанностей (Chain of Responsibility)

2) Команда (Command)

3) Итератор (Iterator)

4) Снимок (Memento)

5) Наблюдатель (Observer)

6) Состояние (State)

7) Стратегия (Strategy)

8) Посетитель (Visitor)

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился применять паттерны проектирования в разработке программ.

# Лабораторная работа №20

**Проектирование API**

1. **Цель работы**
   1. Научиться проектировать API.
2. **Ответы на контрольные вопросы**
   1. Что такое REST?

Ответ: REST (Representational State Transfer) — это архитектурный стиль для создания веб-сервисов, который определяет правила для взаимодействия между клиентом и сервером.

* 1. Для чего используется метод GET?

Ответ: Метод GET используется для получения данных с сервера. Он не изменяет состояние сервера и используется для запросов на чтение информации.

* 1. Для чего используется метод POST?

Ответ: Метод POST используется для отправки данных на сервер для создания новых ресурсов. Он изменяет состояние сервера и обычно используется при отправке форм или загрузке файлов.

* 1. Для чего используется метод PUT?

Ответ: Метод PUT используется для обновления существующего ресурса на сервере. Он заменяет существующие данные новыми данными.

* 1. Для чего используется метод DELETE?

Ответ: Метод DELETE используется для удаления ресурса на сервере. Он удаляет указанный ресурс из базы данных или хранилища данных.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился проектировать API.

# Лабораторная работа №21

**Оптимизация кода**

1. **Цель работы**
   1. Изучить методы оптимизации программного кода.
2. **Ответы на контрольные вопросы**
   1. Что такое «оптимизация программного кода»?

Ответ: Оптимизация программного кода — это процесс улучшения работы программы путем изменения кода с целью увеличения скорости выполнения, уменьшения потребления ресурсов (памяти, процессорного времени) или улучшения общей производительности.

* 1. Какова цель оптимизации программного кода?

Ответ: Цель оптимизации программного кода может быть различной в зависимости от конкретной задачи. Основные цели включают увеличение скорости работы программы, снижение использования ресурсов (например, памяти или процессорного времени), улучшение отзывчивости и производительности приложения.

* 1. Какие методы оптимизации программного кода применяются?

Ответ: Существует множество методов оптимизации программного кода, включая:

- Улучшение алгоритмов: выбор более эффективных алгоритмов для решения задачи.

- Использование более эффективных структур данных: выбор наиболее подходящих структур данных для хранения и обработки информации.

- Оптимизация циклов: уменьшение количества итераций в циклах, избегание лишних операций.

- Устранение избыточных операций: удаление ненужных вычислений или операций.

- Параллелизм: использование параллельных вычислений для повышения производительности.

- Кэширование данных: использование кэшей для ускорения доступа к данным.

- Профилирование и анализ кода: идентификация узких мест и проблем в коде для последующей оптимизации.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил методы оптимизации программного кода.

# Лабораторная работа №22

**Асинхронное программирование**

1. **Цель работы**
   1. Научиться реализовывать и запускать асинхронные операции на C#.
   2. Научиться выполнять вычисления, используя асинхронные операции.
   3. Научиться выполнять ввод и вывод данных, используя асинхронные операции.
2. **Ответы на контрольные вопросы**
   1. Какие ключевые слова используются в C# для работы с асинхронными вызовами?

Ответ: Для работы с асинхронными вызовами используются ключевые слова async и await

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

Ответ: Асинхронные методы могут иметь следующие типы возврата:

- Task: возвращает задачу без результата

- Task<T>: возвращает задачу с результатом типа T

- void: используется редко и обычно для обработки событий.

* 1. Как вызвать метод в асинхронном режиме?

Ответ: Для вызова метода в асинхронном режиме необходимо пометить метод, который его вызывает, как async, а затем использовать ключевое слово await перед вызовом асинхронного метода.

* 1. Как указать, что в методе могут быть асинхронные вызовы?

Ответ: Для указания, что метод содержит асинхронные вызовы, необходимо добавить ключевое слово async к объявлению метода.

* 1. Как обработать исключения, возникшие в асинхронных вызовах?

Ответ: Для обработки исключений, возникших в асинхронных вызовах, можно использовать блоки try-catch вокруг вызова await.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился реализовывать и запускать асинхронные операции на C#.
   2. В ходе проделанной лабораторной работы я научился выполнять вычисления, используя асинхронные операции.
   3. В ходе проделанной лабораторной работы я научился выполнять ввод и вывод данных, используя асинхронные операции.

# Лабораторная работа №23

**Документирование кода**

1. **Цель работы**
   1. Изучить процесс документирования программного кода.
2. **Ответы на контрольные вопросы**
   1. Что такое «XML-документация»?

Ответ: XML-документация – это специальные комментарии в исходном коде программы, написанные в формате XML, которые содержат описания элементов кода (например, классов, методов, свойств) и их параметров.

* 1. Как сгенерировать XML-комментарий?

Ответ: Для генерации XML-комментария в большинстве современных интегрированных сред разработки (IDE) достаточно набрать три символа "///" перед объявлением элемента кода (например, класса, метода) и IDE автоматически создаст шаблон XML-комментария, который можно заполнить описанием.

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

Ответ: Для того чтобы XML-документация была видна при подключении библиотеки в стороннем решении, необходимо сгенерировать файл с документацией в формате XML и предоставить его вместе с библиотекой. Этот файл обычно имеет расширение .xml и содержит описания элементов кода из XML-комментариев.

* 1. Что пишется в разделе summary?

Ответ: В разделе summary пишется краткое описание элемента кода (например, метода или класса). Это описание должно быть достаточно информативным, чтобы пользователь мог понять назначение элемента без необходимости изучения деталей его реализации.

* 1. Что пишется в разделе param?

Ответ: В разделе param указывается описание параметров метода. Каждый параметр должен быть описан отдельно, чтобы пользователь понимал, какие значения можно передать в метод и какие они должны быть.

* 1. Что пишется в разделе returns?

Ответ: В разделе returns указывается описание возвращаемого значения метода. Здесь можно описать тип возвращаемого значения и его значение в контексте выполнения метода.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс документирования программного кода.

# Лабораторная работа №24

**Рефакторинг кода**

1. **Цель работы**
   1. Изучить техники рефакторинга программного кода.
2. **Ответы на контрольные вопросы**
   1. Что такое «рефакторинг»?

Ответ: Рефакторинг - это процесс изменения внутренней структуры программы с целью улучшения её читаемости, понимания, сопровождаемости и/или производительности без изменения её внешнего поведения.

* 1. Какие группы техник рефакторинга существуют?

Ответ: Существует несколько групп техник рефакторинга, такие как:

- Реорганизация кода (например, выделение метода, выделение класса)

- Упрощение выражений (например, устранение дублирования кода, улучшение структуры условных операторов)

- Управление зависимостями (например, уменьшение связанности между классами)

* 1. Как выполнить рефакторинг в Visual Studio?

Ответ: Visual Studio выполнить рефакторинг можно следующим образом:

1) Выделите блок кода, который вы хотите отрефакторить

2) Нажмите правой кнопкой мыши на выделенном коде

3) В контекстном меню выберите опцию "Refactor" (Рефакторинг)

4)В появившемся подменю выберите нужную технику рефакторинга (например, Extract Method - Выделить метод)

5) Cледуйте инструкциям Visual Studio для завершения процесса рефакторинга

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил техники рефакторинга программного кода.

# Лабораторная работа №25

**Работа с системой контроля версий**

1. **Цель работы**
   1. Научиться применять систему контроля версий git в процессе разработки программного обеспечения.
2. **Ответы на контрольные вопросы**
   1. Что такое «репозиторий»?

Ответ: Репозиторий – это хранилище данных, где хранится история изменений кода и другие файлы, относящиеся к проекту

* 1. Что указывается в файле readme.md?

Ответ: Файл readme.md обычно содержит информацию о проекте, инструкции по его установке и запуску, описание функциональности и другую полезную информацию для разработчиков и пользователей.

* 1. Что указывается в файле .gitignore?

Ответ: Файл .gitigonre используется для указания файлов и папок, которые Git должен игнорировать при отслеживании изменений в репозитории. Это позволяет исключить временные файлы, файлы конфигурации и т.д.

* 1. Какое программное обеспечение может применяться для управления git-репозиторием?

Ответ: GitKraken, SourceTree, GitHub Desktop, GitLab.

* 1. Где может располагаться репозиторий?

Ответ: GitHub, GitLab, Bitbucket.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился применять систему контроля версий git в процессе разработки программного обеспечения.

# Лабораторная работа №26

**Разработка интерфейса пользователя: компоновка элементов**

1. **Цель работы**
   1. Изучить элементы-контейнеры, применяющиеся в приложениях WPF для компоновки.
2. **Ответы на контрольные вопросы**
   1. Что такое «компоновка» в WPF?

Ответ: Компоновка (layout) отвечает за расположение и размеры элементов управления на форме. Она определяет, как элементы будут размещаться и выравниваться относительно друг друга

* 1. Какой класс является родительским для всех элементов-контейнеров в WPF?

Ответ: Класс Panel является родительским для всех элементов-контейнеров.

* 1. Как выровнять элементы внутри контейнера по высоте и по ширине?

Ответ: Для выравнивания элементов по высоте и ширине можно использовать различные свойствами, такими как HorizontalAlignment, VerticalAlignment, Width, Height, Margin.

* 1. В чем особенность компоновки с использованием следующих элементов-контейнеров: Grid и Canvas?

Ответ: Особенности компоновки с использованием элементов-контейнеров в WPF:

- Grid: позволяет размещать элементы в виде сетки с ячейками, где каждый элемент может занимать несколько строк и столбцов.

- Canvas: позволяет абсолютно позиционировать элементы на плоскости.

* 1. Чем отличается компоновка с использованием StackPanel, DockPanel, WrapPanel?

Ответ: Различия между компоновками с использованием StackPanel, DockPanel, WrapPanel:

- StackPanel: располагает элементы последовательно в одну линию по горизонтали или вертикали.

- DockPanel: позволяет привязывать элементы к краям контейнера или заполнять ими доступное пространство.

- WrapPanel: автоматически переносит элементы на новую строку или столбец, когда они не помещаются в доступном пространстве.

* 1. В каких единицах измерения могут задаваться размеры элементов в приложениях WPF?

Ответ: Размеры элементов в приложениях WPF могут задаваться в различных единицах измерения, таких как пиксели (px), точки (pt), проценты (%), дюймы (in), сантиметры (cm), а также в относительных единицах, таких как Auto и \* для автоматического и пропорционального размера соответственно.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил элементы-контейнеры, применяющиеся в приложениях WPF для компоновки.

# Лабораторная работа №27

**Организация интерфейса пользователя**

1. **Цель работы**
   1. Изучить процесс настройки интерфейса и организации переходов в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Для чего используется элемент управления Frame?

Ответ: “Frame” используется для отображения другого содержимого, такого как страницы, внутри определенного раздела пользовательского интерфейса

* 1. Для чего используется элемент управления Page?

Ответ: Page используется для представления отдельного экрана или страницы в приложении.

* 1. Как перейти к определенной странице, используя фрейм?

Ответ: Чтобы перейти к определенной странице, используя Frame, можно использовать метод Navigate.

Frame.Navigate(typeof(Страница));

* 1. Как проверить, что во фрейме можно вернуться к предыдущей странице?

Ответ: Чтобы проверить, можно ли вернуться к предыдущей странице во фрейме, нужно проверить свойство CanGoBack

* 1. Как перейти к предыдущей странице, используя фрейм?

Ответ: Используя код: myFrame.GoBack();

* 1. Какие элементы позволяют сгруппировать содержимое?

Ответ: Элементы, которые позволяют сгруппировать содержимое, включают: 1) **Grid**: Используется для создания сеточной компоновки. 2)**StackPanel**: Выстраивает дочерние элементы в вертикальный или горизонтальный стек. 3) **Canvas**: Позволяет абсолютно позиционировать элементы. 4) **WrapPanel**: Располагает дочерние элементы по горизонтали или вертикали и переносит их на новую строку или столбец, если пространство заканчивается. 5) **GroupBox**: Оборачивает содержимое в рамку с заголовком. 6) **Border**: Позволяет обернуть один элемент управления с возможностью настройки границ и фона.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс настройки интерфейса и организации переходов в приложениях WPF.

# Лабораторная работа №28

**Разработка интерфейса пользователя: настройка стилей**

1. **Цель работы**
   1. Изучить процесс настройки интерфейса с использованием стилей в приложениях WPF
2. **Ответы на контрольные вопросы**
   1. Для чего используются стили в приложениях WPF?

Ответ: Стили в WPF используются для централизованного управления внешним видом и поведением элементов управления.

* 1. Какова общая форма локального определения стиля элемента управления?

Ответ: Локальное определение стиля элемента управления в WPF выглядит следующим образом:

<Window.Resources>

<Style TargetType="Button">

Здесь задаются свойства стиля

</Style>

</Window.Resources>

* 1. Какова общая форма определения стиля приложения?

Ответ: Определение стиля для всего приложения обычно происходит в ресурсах приложения (App.xaml)

<Application.Resources>

<Style TargetType="Button">

Здесь задаются свойства стиля

</Style>

</Application.Resources>

* 1. Как указать явное использование стилей?

Ответ: Для явного указания использования стиля, необходимо задать свойство Style элемента управления:

<Button Style="{StaticResource MyButtonStyle}">Click Me</Button>

* 1. Как указать наследование стиля?

Ответ: Для наследования стиля можно использовать свойство BasedOn

<Style x:Key="SpecialButtonStyle" TargetType="Button" BasedOn="{StaticResource MyButtonStyle}">

Здесь задаются специфичные свойства стиля

</Style>

* 1. Как добавить новую тему в приложение?

Ответ: Чтобы добавить новую тему, создайте новый ресурсный словарь и подключите его к приложению

* 1. Как выполнить переключение между темами?

Ответ: Для переключения между темами можно заменить ресурсный словарь.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс настройки интерфейса с использованием стилей в приложениях WPF

# Лабораторная работа №29

**Разработка интерфейса пользователя: применение триггеров**

1. **Цель работы**
   1. Изучить процесс применения триггеров в приложениях WPF
   2. Закрепить навык применения стилей в приложениях на WPF.
2. **Ответы на контрольные вопросы**
   1. Что позволяют делать триггеры в приложениях WPF?

Ответ: Триггеры в WPF позволяют изменять свойства элементов управления в ответ на определенные условия или события, такие как изменение значений свойств или возникновение событий.

* 1. Какие виды триггеров можно разработать в приложениях WPF?

Ответ: Существует три основных вида триггеров:

1) Property Triggers (Триггеры свойств)

2) Data Triggers (Триггеры данных)

3) Event Triggers (Триггеры событий)

* 1. Для чего используется и когда срабатывает триггер свойств?

Ответ: Триггер свойств срабатывает при изменении значения свойства элемента управления.

* 1. Для чего используется и когда срабатывает триггер данных?

Ответ: Триггер данных используется для реагирования на изменения данных в привязке.

* 1. Для чего используется и когда срабатывает триггер событий?

Ответ: Триггер событий срабатывает в ответ на события, такие как нажатие кнопки.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс применения триггеров в приложениях WPF
   2. В ходе проделанной лабораторной работы я закрепил навык применения стилей в приложениях на WPF.

# Лабораторная работа №30

**Изучение особенностей элементов выбора в приложениях WPF**

1. **Цель работы**
   1. Изучить свойства и процесс обработки событий элементов выбора в приложениях WPF
2. **Ответы на контрольные вопросы**
   1. Что такое CheckBox и для чего он используется?

Ответ: CheckBox - это элемент управления, который позволяет пользователю выбрать или снять выбор одного или нескольких вариантов из набора.

* 1. Что такое RadioButton и для чего он используется?

Ответ: RadioButton - это элемент управления, который позволяет пользователю выбрать один вариант из группы взаимно исключающих вариантов.

* 1. Как проверить, что флажок или переключатель выбран?

Ответ: Можно проверить свойство IsChecked

* 1. Какое событие срабатывает при выборе флажка или переключателя?

Ответ: Событие Checked срабатывает при выборе флажка или переключателя

* 1. Какое событие срабатывает при снятии выбора флажка или переключателя?

Ответ: Событие Unchecked срабатывает при снятии выбора флажка или переключателя

* 1. Какие значения могут принимать флажки и переключатели?

Ответ: Флажки могут принимать значения true, false и null (если свойство IsThreeState установлено в true). Переключатели могут принимать значения true и false

1. **Вывод**
   1. В ходе проделанной лабораторной работы 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. Изучить различные типы полей ввода, применяющихся в приложениях WPF;
   2. Изучить свойства полей ввода и процесс обработки событий полей ввода.
2. **Ответы на контрольные вопросы**
   1. Как задать имя элементам управления в WPF?

Ответ: При помощи атрибута “x:Name”.

* 1. Как создать обработчик события в WPF?

Ответ: Можно задать обработчик в XAML и определить его в коде

* 1. Для чего используется Slider в WPF?

Ответ: Slider используется для выбора числового значения из определенного диапазона.

* 1. Для чего используется TextBox в WPF?

Ответ: TextBox используется для ввода и отображения текста.

* 1. Для чего используется TextBlock в WPF?

Ответ: TextBlock используется для отображения текста с возможностью форматирования.

* 1. Для чего используется Calendar в WPF?

Ответ: Calendar используется для выбора даты из календаря.

* 1. Для чего используется DatePicker в WPF?

Ответ: DatePicker используется для выбора даты с помощью выпадающего календаря.

* 1. Для чего используется PasswordBox в WPF?

Ответ: PasswordBox используется для ввода пароля, скрывая введенные символы.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил различные типы полей ввода, применяющихся в приложениях WPF;
   2. В ходе проделанной лабораторной работы я изучил свойства полей ввода и процесс обработки событий полей ввода.

# Лабораторная работа №32

**Разработка приложения с использованием элементов отображения списков**

1. **Цель работы**
   1. Изучить свойства и процесс обработки событий элементов отображения списков в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Что такое ComboBox и для чего он используется?

Ответ: ComboBox - это элемент управления, который позволяет пользователю выбрать один элемент из выпадающего списка.

* 1. Что такое ListBox и для чего он используется?

Ответ: ListBox - это элемент управления, который позволяет пользователю выбрать один или несколько элементов из списка.

* 1. Какое событие срабатывает при выборе элемента в селекторе?

Ответ: Событие SelectionChanged срабатывает при выборе элемента.

* 1. В каком свойстве хранятся элементы селекторов?

Ответ: Элементы хранятся в свойстве “Items”.

* 1. Какого типа элементы могут быть в селекторе?

Ответ: Элементы могут быть любого типа, но обычно это объекты, наследующиеся от UIElement, например, ComboBoxItem или ListBoxItem.

* 1. Какое свойство позволяет привязать селектор к набору данных?

Ответ: Свойство ItemsSource позволяет привязать селектор к набору данных.

* 1. Для чего используется свойство DisplayMemberPath в селекторе?

Ответ: DisplayMemberPath указывает, какое свойство элементов коллекции должно использоваться для отображения.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил свойства и процесс обработки событий элементов отображения списков в приложениях WPF.

# Лабораторная работа №33

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

1. **Цель работы**
   1. Изучить свойства и процесс настройки внешнего вида элемента DataGrid в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Что такое DataGrid и для чего он используется?

Ответ: DataGrid - это элемент управления, который используется для отображения данных в виде таблицы с возможностью редактирования, сортировки и фильтрации.

* 1. Какие типы столбцов поддерживаются в DataGrid?

Ответ: DataGrid поддерживает различные типы столбцов, включая:

1. DataGridTextColumn
2. DataGridCheckBoxColumn
3. DataGridComboBoxColumn
4. DataGridHyperlinkColumn
5. DataGridTemplateColumn
   1. Как добавить кнопку в строки DataGrid?

Ответ: Используйте DataGridTemplateColumn для добавления кнопки.

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

* 1. Как указать источник данных для DataGrid?

Ответ: Укажите источник данных с помощью свойства ItemsSource.

<DataGrid ItemsSource="{Binding MyData}" />

* 1. Как указать источник данных для выпадающего списка DataGrid?

Ответ: Используйте DataGridComboBoxColumn и укажите ItemsSource для выпадающего списка.

<DataGridComboBoxColumn Header="Options" SelectedItemBinding="{Binding SelectedOption}" ItemsSource="{Binding Options}" />

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил свойства и процесс настройки внешнего вида элемента DataGrid в приложениях WPF.

# Лабораторная работа №34

**Разработка приложения с меню и панелью инструментов**

1. **Цель работы**
   1. Изучить свойства и процесс настройки внешнего вида меню и панели инструментов в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Что такое Menu и для чего он используется?

Ответ: Menu - это элемент управления, который используется для создания многоуровневых меню в приложении.

* 1. Что такое ContextMenu и для чего он используется?

Ответ: ContextMenu - это контекстное меню, которое появляется при щелчке правой кнопкой мыши и предоставляет действия, связанные с текущим контекстом.

* 1. Что такое ToolBar и для чего он используется?

Ответ: ToolBar - это элемент управления, который содержит коллекцию кнопок и других элементов управления, часто используемых для быстрого доступа к функциям приложения.

* 1. Что такое StatusBar и для чего он используется?

Ответ: StatusBar - это элемент управления, который отображает статусную информацию и другие индикаторы состояния в нижней части окна приложения.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил свойства и процесс настройки внешнего вида меню и панели инструментов в приложениях WPF.

# Лабораторная работа №35

**Создание проекта с использованием компонентов стандартных диалогов**

1. **Цель работы**
   1. Изучить процесс создания и применения стандартных диалоговых окон в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Что такое диалоговое окно?

Ответ: Диалоговое окно - это интерактивный элемент пользовательского интерфейса, который появляется поверх основного окна приложения и предназначен для обмена информацией с пользователем или получения от него определенных действий.

* 1. Для чего используется OpenFileDialog?

Ответ: OpenFileDialog - это стандартное диалоговое окно в WPF, которое используется для выбора одного или нескольких файлов для открытия в приложении. Пользователь может навигироваться по файловой системе и выбирать файлы для загрузки.

* 1. Для чего используется SaveFileDialog?

Ответ: SaveFileDialog - это стандартное диалоговое окно в WPF, которое используется для выбора имени и расположения файла, в который пользователь хочет сохранить данные из приложения. Оно предоставляет возможность выбора имени файла и его расположения.

* 1. Что такое MessageBox и какие настройки можно ему задать?

Ответ: MessageBox - это стандартное диалоговое окно в WPF, которое используется для отображения сообщений пользователю. Он может быть использован для вывода информации, предупреждений или запроса подтверждения от пользователя. Настройки MessageBox могут включать заголовок окна, текст сообщения, кнопки выбора (например, ОК, Отмена, Да, Нет) и значок сообщения (например, вопрос, предупреждение, информация).

* 1. Для чего используется PrintDialog?

Ответ: PrintDialog - это стандартное диалоговое окно в WPF, которое используется для настройки параметров печати и выбора принтера перед печатью документа или изображения. Пользователь может выбрать принтер, настроить параметры печати (например, количество копий, диапазон страниц и т. д.) и запустить процесс печати.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс создания и применения стандартных диалоговых окон в приложениях WPF.

# Лабораторная работа №36

**Разработка приложения с несколькими формами**

1. **Цель работы**
   1. Изучить процесс создания и применения пользовательских окон в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Как сделать доступными данные пользовательского диалогового окна вызывающим его окнам?

Ответ: Для передачи данных из пользовательского диалогового окна вызывающим его окнам можно использовать свойство DialogResult. Для этого необходимо установить это свойство в окне диалога, когда пользователь завершает работу с ним. Затем вызывающее окно может получить результат из этого свойства.

* 1. Какие значения может принимать переменная DialogResult?

Ответ: true: указывает, что пользователь завершил работу с диалоговым окном, подтвердив его действие.

false: указывает, что пользователь отменил работу с диалоговым окном или нажал кнопку "Отмена" или "Закрыть".

null: значение по умолчанию, которое указывает, что диалоговое окно еще не завершило работу.

* 1. Как открыть окно в диалоговом режиме?

Ответ: Для открытия окна в диалоговом режиме используется метод ShowDialog(). Этот метод блокирует вызывающее окно до тех пор, пока диалоговое окно не будет закрыто

* 1. Как открыть окно в не диалоговом режиме?

Ответ: Для открытия окна в недиалоговом режиме используется метод Show(). Этот метод открывает окно независимо от вызывающего окна и не блокирует его.

* 1. В чем отличие между диалоговым и не диалоговым режимами работы?

Ответ: Основное отличие между диалоговым и недиалоговым режимами работы заключается в том, как управляется поток выполнения программы. В диалоговом режиме вызывающее окно блокируется до закрытия диалогового окна, тогда как в недиалоговом режиме выполнение программы продолжается без ожидания закрытия окна.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс создания и применения пользовательских окон в приложениях WPF.

# Лабораторная работа №37

**Реализация фильтрации данных**

1. **Цель работы**
   1. Научиться применять LINQ-запросы для фильтрации данных по одному критерию и набору критериев.
2. **Ответы на контрольные вопросы**
   1. Для чего используется метод Where?

Ответ: Метод Where используется для фильтрации элементов в коллекции на основе заданного условия. Он возвращает новую последовательность, содержащую только те элементы, для которых условие истинно.

* 1. Какие логические операторы могут применяться при составлении условий?

Ответ: && (логическое "и"): все условия должны быть истинны.

|| (логическое "или"): хотя бы одно из условий должно быть истинным.

! (логическое "не"): инвертирует результат условия.

* 1. Как выполнить регистронезависимый поиск?

Ответ: Для выполнения регистронезависимого поиска можно использовать методы ToLower() или ToUpper() для преобразования строк к одному регистру перед сравнением.

* 1. Как проверить, что строка начинается с определенного текста?

Ответ: Для проверки, что строка начинается с определенного текста, можно использовать метод StartsWith().

* 1. Как проверить, что строка содержит определенный текст?

Ответ: Для проверки, что строка содержит определенный текст, можно использовать метод Contains().

* 1. Как составить LINQ-запрос для фильтрации по нескольким критериям, указываемым пользователем?

Ответ: Для фильтрации по нескольким критериям, указываемым пользователем, можно использовать комбинацию условий с логическими операторами.

1. **Вывод**
   1. В ходе проделанной лабораторной работы 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. Научиться применять LINQ-запросы для постраничного вывода данных.
2. **Ответы на контрольные вопросы**
   1. Почему может потребоваться выводить данные постранично?

Ответ: Улучшение производительности: Загрузка всех данных сразу может занять много времени и потребовать значительных ресурсов. Постраничный вывод позволяет загружать и отображать данные небольшими порциями, что снижает нагрузку на сервер и улучшает отклик приложения.

Удобство пользователя: Пользователю легче просматривать данные небольшими порциями. Постраничная навигация позволяет пользователю быстрее находить нужную информацию и улучшает общее восприятие интерфейса.

Оптимизация использования памяти: Постраничный вывод уменьшает потребление памяти на клиентской стороне, так как в памяти хранится только текущая страница данных.

* 1. Что такое «пагинация»?

Ответ: Пагинация (pagination) — это процесс разделения большого объема данных на несколько страниц, каждая из которых отображает подмножество данных. Пагинация часто используется в веб-приложениях для удобства отображения длинных списков или таблиц данных, позволяя пользователю переходить между страницами с помощью кнопок навигации.

* 1. Для чего используется метод Take?

Ответ: Метод “Take” используется для получения первых N элементов из коллекции. Этот метод полезен, когда необходимо извлечь ограниченное количество элементов из большого набора данных.

* 1. Для чего используется метод Skip?

Ответ: Метод Skip используется для пропуска первых N элементов в коллекции. Этот метод часто используется вместе с методом Take для реализации пагинации, чтобы пропустить уже отображенные элементы и перейти к следующей странице данных.

* 1. Для чего используется метод TakeWhile?

Ответ: Метод TakeWhile используется для получения элементов из коллекции до тех пор, пока выполняется заданное условие. Как только условие перестает выполняться, метод прекращает извлечение элементов.

* 1. Для чего используется метод SkipWhile?

Ответ: Метод SkipWhile используется для пропуска элементов в коллекции до тех пор, пока выполняется заданное условие. Как только условие перестает выполняться, метод начинает извлекать элементы. Этот метод полезен, когда нужно пропустить начальные элементы, соответствующие определенному критерию, и затем получить оставшиеся элементы.

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

Ответ: Для использования методов Take и Skip набор данных должен быть итерируемым, то есть поддерживать интерфейс, который позволяет проходить по его элементам по порядку. В C# это может быть любая коллекция, реализующая интерфейсы IEnumerable<T> или IQueryable<T>, такие как массивы, списки, последовательности, полученные из базы данных, или другие коллекции данных.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился применять LINQ-запросы для постраничного вывода данных

# Лабораторная работа №39

**Реализация группировки и соединения данных**

1. **Цель работы**
   1. Научиться применять LINQ-запросы для группировки и соединения данных.
2. **Ответы на контрольные вопросы**
   1. Какие агрегатные функции поддерживаются в LINQ?

Ответ:

Count: Подсчитывает количество элементов в коллекции.

Sum: Вычисляет сумму значений в коллекции.

Average: Вычисляет среднее значение элементов в коллекции.

Min: Находит минимальное значение в коллекции.

Max: Находит максимальное значение в коллекции.

Aggregate: Применяет настраиваемую функцию агрегации, которая может быть определена пользователем.

First: Возвращает первый элемент коллекции.

Last: Возвращает последний элемент коллекции.

Single: Возвращает единственный элемент коллекции или выбрасывает исключение, если коллекция содержит более одного элемента.

* 1. Что возвращает метод Distinct?

Ответ: Метод Distinct возвращает новую коллекцию, содержащую только уникальные элементы исходной коллекции. Этот метод устраняет дубликаты, основываясь на стандартном сравнении элементов или на указанном компараторе.

* 1. Для чего используется метод GroupBy?

Ответ: Метод GroupBy используется для группировки элементов коллекции по определенному ключу. Этот метод создает коллекцию групп, где каждая группа содержит элементы, имеющие одинаковое значение ключа. Например, можно сгруппировать коллекцию студентов по курсам, на которых они учатся.

* 1. Для чего используется метод Join?

Ответ: Метод Join используется для объединения двух коллекций на основе соответствующих ключей. Он позволяет выполнить внутреннее соединение (inner join), которое объединяет элементы из двух коллекций, имеющие совпадающие ключи.

* 1. Для чего используется метод GroupJoin?

Ответ: Метод GroupJoin используется для выполнения группового соединения (group join) двух коллекций на основе ключей. Этот метод создает коллекцию, где каждый элемент включает элемент из первой коллекции и группу соответствующих элементов из второй коллекции. Это полезно для создания иерархических структур, таких как категория и ее продукты.

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

Ответ:

Union, Except, Intersect?

Concat: Объединяет две коллекции, добавляя элементы второй коллекции в конец первой коллекции. Дубликаты сохраняются.

var result = collection1.Concat(collection2);

Union: Объединяет две коллекции, возвращая только уникальные элементы, встречающиеся в любой из коллекций.

var result = collection1.Union(collection2);

Except: Возвращает элементы из первой коллекции, которых нет во второй коллекции.

var result = collection1.Except(collection2);

Intersect: Возвращает элементы, которые присутствуют в обеих коллекциях.

var result = collection1.Intersect(collection2);

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился применять LINQ-запросы для группировки и соединения данных.

# Лабораторная работа №40

**Разработка приложения для работы с графикой**

1. **Цель работы**
   1. Изучить процесс рисования и трансформации объектов в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Какие графические примитивы доступны в приложениях WPF?

Ответ:

Line: Линия между двумя точками.

Rectangle: Прямоугольник.

Ellipse: Эллипс или окружность.

Polygon: Многоугольник, который может иметь любую форму, заданную множеством вершин.

Polyline: Ломаная линия, состоящая из нескольких сегментов.

Path: Контейнер для сложных геометрических фигур, состоящих из линий, кривых и дуг.

Text: Текстовый элемент для отображения текста.

Geometry: Абстрактный класс, представляющий геометрические фигуры (LineGeometry, RectangleGeometry, EllipseGeometry, PathGeometry и т.д.).

* 1. Для чего используется Path?

Ответ: Path используется для отображения сложных геометрических фигур и произвольных контуров. С его помощью можно рисовать фигуры, которые не могут быть легко описаны стандартными примитивами (например, сложные кривые и контуры). Path может содержать множество сегментов, таких как линии, кривые Безье и дуги, что делает его очень гибким для создания векторной графики.

* 1. Какие виды трансформаций объектов доступны в приложениях WPF?

Ответ: TranslateTransform: Перемещение (трансляция) объекта на заданное расстояние по осям X и Y.

RotateTransform: Вращение объекта вокруг заданной точки на заданный угол.

ScaleTransform: Масштабирование объекта по осям X и Y.

SkewTransform: Косое преобразование объекта по осям X и Y.

MatrixTransform: Общая трансформация, определяемая матрицей преобразования, позволяющая комбинировать различные виды трансформаций.

* 1. Как указать заливку и контур графических объектов в WPF?

Ответ: Fill: Задает заливку внутренней области графического объекта. Можно использовать сплошной цвет, градиент или изображение.

Stroke: Задает цвет и параметры контура (границы) графического объекта. Также можно указать толщину контура через свойство StrokeThickness.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс рисования и трансформации объектов в приложениях WPF.

# Лабораторная работа №41

**Разработка приложения с анимацией**

1. **Цель работы**
   1. Изучить процесс анимации объектов в приложениях WPF.
2. **Ответы на контрольные вопросы**
   1. Как задать анимацию размеров в приложениях WPF?

Ответ: Для задания анимации размеров в WPF можно использовать класс DoubleAnimation, который позволяет анимировать числовые значения, такие как ширина и высота элементов.

* 1. Как задать анимацию цвета в приложениях WPF?

Ответ: Для задания анимации цвета в WPF можно использовать класс ColorAnimation, который позволяет анимировать значения цвета.

* 1. Чем отличается покадровая анимация от плавной анимации в WPF?

Ответ: Плавная анимация (Smooth Animation): Использует классы анимаций, такие как DoubleAnimation, ColorAnimation, и другие, для плавного изменения значений свойств элементов с течением времени. Это достигается за счет интерполяции значений между начальным и конечным состоянием. Плавная анимация выполняется автоматически WPF и создает гладкие переходы.

Покадровая анимация (Frame-Based Animation): Включает в себя определение ключевых кадров (KeyFrames), каждый из которых задает конкретное значение свойства в определенный момент времени. WPF поддерживает покадровую анимацию через классы, такие как DoubleAnimationUsingKeyFrames, ColorAnimationUsingKeyFrames, и другие. Покадровая анимация позволяет более точно контролировать анимацию, задавая конкретные значения в конкретные моменты времени.

* 1. Какие свойства позволяют управлять анимацией в WPF и для чего предназначено каждое свойство??

Ответ:

Duration: Задает продолжительность анимации. Определяет, сколько времени займет анимация от начала до конца.

Duration="0:0:2" // 2 секунды

From: Начальное значение анимации. Указывает, с какого значения должно начинаться анимационное изменение.

From="100"

To: Конечное значение анимации. Указывает, на какое значение должно измениться анимируемое свойство.

To="200"

By: Задает величину изменения свойства относительно начального значения. Используется вместо или в дополнение к To.

By="50"

RepeatBehavior: Определяет, сколько раз будет повторяться анимация. Можно задать количество повторений или задать бесконечное повторение.

RepeatBehavior="3x" // Повторение 3 раза

RepeatBehavior="Forever" // Бесконечное повторение

AutoReverse: Указывает, будет ли анимация автоматически воспроизводиться в обратном порядке после достижения конечного значения.

AutoReverse="True"

AccelerationRatio: Задает долю времени анимации, в течение которой скорость анимации увеличивается от 0 до полной скорости.

AccelerationRatio="0.5"

DecelerationRatio: Задает долю времени анимации, в течение которой скорость анимации уменьшается от полной скорости до 0.

DecelerationRatio="0.5"

BeginTime: Задает время задержки перед началом анимации.

BeginTime="0:0:1" // Задержка на 1 секунду перед началом

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс анимации объектов в приложениях WPF.

# Лабораторная работа №42

**Разработка мультимедиа-приложения**

1. **Цель работы**
   1. Изучить процесс создания мультимедиа-приложений на WPF.
2. **Ответы на контрольные вопросы**
   1. Что такое «мультимедиа»?

Ответ: «Мультимедиа» — это технология, которая объединяет различные виды медиа, такие как текст, аудио, изображения, анимацию и видео, для создания интерактивных и мультимедийных приложений.

* 1. Какие элементы позволяют отображать изображения в приложениях WPF?

Ответ: Image: Основной элемент для отображения растровых изображений. Он поддерживает различные форматы изображений, такие как BMP, JPEG, PNG, GIF, TIFF и др.

<Image Source="image.jpg" Width="200" Height="200"/>

ImageBrush: Используется для заполнения областей других элементов (например, фона кнопок, границ и т.д.) изображениями.

<Rectangle Width="200" Height="200">

<Rectangle.Fill>

<ImageBrush ImageSource="image.jpg"/>

</Rectangle.Fill>

</Rectangle>

* 1. Какие элементы позволяют воспроизводить аудио в приложениях WPF?

Ответ: MediaElement: Основной элемент для воспроизведения аудио и видео. Он поддерживает различные форматы аудио, такие как MP3, WMA и др.

<MediaElement Name="audioPlayer" Source="audio.mp3" LoadedBehavior="Manual" UnloadedBehavior="Stop"/>

SoundPlayer: Класс, используемый для воспроизведения звуковых файлов в формате WAV.

using System.Media;

SoundPlayer player = new SoundPlayer("sound.wav");

player.Play();

* 1. Какие элементы позволяют воспроизводить видео в приложениях WPF?

Ответ: MediaElement: Основной элемент для воспроизведения видео. Поддерживает различные видеоформаты, такие как WMV, MP4 и др.

<MediaElement Name="videoPlayer" Source="video.mp4" Width="400" H eight="300" LoadedBehavior="Manual" UnloadedBehavior="Stop"/>

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс создания мультимедиа-приложений на WPF.

# Лабораторная работа №43

**Разработка игрового приложения**

1. **Цель работы**
   1. Изучить процесс разработки игровых приложений WPF, использующих графику, обработчики событий и таймеры.
2. **Ответы на контрольные вопросы**
   1. Как подключить таймер к приложению на WPF?

Ответ: Для подключения таймера к приложению на WPF можно использовать класс DispatcherTimer, который интегрирован с главным потоком пользовательского интерфейса (UI) и безопасен для использования в WPF-приложениях.

* 1. Как создать обработчик события для таймера в приложении WPF?

Ответ: Создание обработчика события для таймера осуществляется подпиской на событие Tick таймера. В примере выше обработчик события Timer\_Tick вызывается каждый раз, когда срабатывает таймер.

* 1. Как изменить интервал таймера в приложении WPF?

Ответ: Интервал таймера можно изменить, установив новое значение свойства Interval у экземпляра DispatcherTimer. Это можно сделать в любое время, когда таймер работает.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс разработки игровых приложений WPF, использующих графику, обработчики событий и таймеры.

# Лабораторная работа №44

**Создание БД**

1. **Цель работы**
   1. Изучить процесс создания таблиц и связей между ними в реляционной СУБД.
2. **Ответы на контрольные вопросы**
   1. Что такое «система управления базами данных»?

Ответ: Система управления базами данных (СУБД) — это программное обеспечение, предназначенное для создания, управления и взаимодействия с базами данных. СУБД обеспечивает возможность определения, создания, редактирования, обновления и удаления данных в базе данных. Основные функции СУБД включают:

Хранение данных: Обеспечение надежного хранения больших объемов данных.

Манипулирование данными: Вставка, обновление, удаление и выборка данных.

Администрирование данных: Управление пользователями, правами доступа, резервное копирование и восстановление данных.

Обеспечение целостности данных: Поддержание согласованности и целостности данных через механизмы транзакций, ограничений и проверок.

Поддержка многопользовательского доступа: Обеспечение одновременного доступа к данным несколькими пользователями без конфликтов.

* 1. Что такое «база данных»?

Ответ: База данных — это организованная коллекция данных, которая хранится и управляется с помощью СУБД. База данных структурирована таким образом, чтобы обеспечивать эффективное хранение, управление и поиск информации. Данные в базе данных могут быть организованы в таблицы, строки и столбцы (в случае реляционных баз данных), документы, графы и другие структуры в зависимости от типа базы данных.

Основные типы баз данных:

Реляционные базы данных: Организованы в таблицы, которые могут быть связаны друг с другом по ключам. Примеры: SQL Server, MySQL, PostgreSQL.

Документо-ориентированные базы данных: Хранят данные в формате документов (например, JSON). Примеры: MongoDB, CouchDB.

Графовые базы данных: Хранят данные в виде узлов и ребер, представляющих связи между узлами. Примеры: Neo4j, Amazon Neptune.

Ключ-значение базы данных: Хранят данные в парах ключ-значение. Примеры: Redis, Riak

* 1. Какие СУБД являются реляционными (привести примеры ПО)?

Ответ: Реляционные СУБД (РСУБД) — это системы управления базами данных, основанные на реляционной модели данных, предложенной Эдгаром Коддом. В этой модели данные организованы в таблицы, и отношения между данными определяются с помощью ключей. Примеры реляционных СУБД:

Microsoft SQL Server: Разработанная Microsoft, широко используемая в корпоративных приложениях.

MySQL: Открытое программное обеспечение, популярное среди разработчиков веб-приложений. Сейчас поддерживается Oracle Corporation.

PostgreSQL: Открытое программное обеспечение с поддержкой расширенных возможностей и соответствия стандартам SQL.

Oracle Database: Коммерческая СУБД от Oracle Corporation, широко используемая в крупных предприятиях.

SQLite: Легковесная встраиваемая СУБД, часто используемая в мобильных приложениях и небольших десктопных приложениях.

IBM Db2: Коммерческая СУБД от IBM, используемая в крупных предприятиях.

MariaDB: Форк MySQL, также поддерживаемый сообществом и имеющий схожие с MySQL функции.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я изучил процесс создания таблиц и связей между ними в реляционной СУБД.

# Лабораторная работа №45

**Создание приложения с БД для чтения данных**

1. **Цель работы**
   1. Научиться создавать приложения для чтения данных из БД.
2. **Ответы на контрольные вопросы**
   1. Каково назначение элемента SqlConnection?

Ответ: SqlConnection — это класс из пространства имен System.Data.SqlClient, предназначенный для установления и управления подключением к базам данных Microsoft SQL Server. Он используется для выполнения команд, получения данных и управления транзакциями в контексте подключения к базе данных

* 1. Как считать из БД одно значение?

Ответ: Для считывания одного значения из базы данных можно использовать метод ExecuteScalar класса SqlCommand. Этот метод выполняет запрос и возвращает первое значение первой строки результата.

* 1. Каково назначение элемента SqlDataReader?

Ответ: SqlDataReader — это класс из пространства имен System.Data.SqlClient, используемый для быстрого и эффективного чтения данных из базы данных Microsoft SQL Server в режиме только для чтения и только вперед. Он обеспечивает доступ к результатам выполнения команды SQL, позволяя построчно обрабатывать данные

* 1. Как получить данные из БД, используя SqlDataAdapter?

Ответ: SqlDataAdapter — это класс из пространства имен System.Data.SqlClient, который используется для заполнения объекта DataSet или DataTable данными из базы данных и обновления базы данных с помощью данных из DataSet или DataTable. Он работает как мост между базой данных и объектами в памяти

* 1. Какие пространства имен требуется подключить для реализации подключения к СУБД MS SQL Server?

Ответ: Для реализации подключения к СУБД MS SQL Server требуется подключить следующие пространства имен:

System.Data: Содержит классы для работы с данными и управления ими.

System.Data.SqlClient: Содержит классы для работы с Microsoft SQL Server, включая SqlConnection, SqlCommand, SqlDataReader и SqlDataAdapter.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился создавать приложения для чтения данных из БД.

# Лабораторная работа №46

**Создание приложения с БД для записи данных**

1. **Цель работы**
   1. Научиться создавать приложения для записи данных в БД.
2. **Ответы на контрольные вопросы**
   1. Для чего применяется компонент SqlDataAdapter?

Ответ: SqlDataAdapter — это компонент из пространства имен System.Data.SqlClient, который используется для заполнения объектов DataSet или DataTable данными из базы данных и для обновления базы данных на основе изменений в DataSet или DataTable. SqlDataAdapter выполняет функции мостика между источником данных и объектами в памяти, предоставляя функциональность для выборки и сохранения данных.

Основные задачи SqlDataAdapter:

- Заполнение DataSet/DataTable данными из базы данных;

- Синхронизация изменений в DataSet/DataTable с базой данных.

* 1. Для чего применяется компонент SqlCommandBuilder?

Ответ: SqlCommandBuilder — это компонент из пространства имен System.Data.SqlClient, который автоматически генерирует команды SQL для синхронизации изменений в объекте DataSet или DataTable с базой данных. SqlCommandBuilder упрощает создание команд INSERT, UPDATE и DELETE для обновления базы данных.

Основные задачи SqlCommandBuilder:

- Автоматическая генерация команд SQL на основе схемы и изменений в DataSet/DataTable.

- Упрощение работы с SqlDataAdapter для синхронизации данных.

* 1. Как изменить данные в БД, используя SqlCommand?

Ответ: Для изменения данных в базе данных с использованием SqlCommand необходимо создать команду SQL, например, команду UPDATE, и выполнить её с помощью метода ExecuteNonQuery.

* 1. Как изменить данные в БД, используя SqlDataAdapter?

Ответ: Для изменения данных в базе данных с использованием SqlDataAdapter необходимо внести изменения в объект DataSet или DataTable, а затем вызвать метод Update у SqlDataAdapter для синхронизации изменений с базой данных.

* 1. Как связать SqlCommandBuilder и SqlDataAdapter?

Ответ: Для связывания SqlCommandBuilder и SqlDataAdapter достаточно создать экземпляр SqlCommandBuilder, передав ему в конструкторе объект SqlDataAdapter. Это позволит SqlCommandBuilder автоматически генерировать команды INSERT, UPDATE и DELETE для SqlDataAdapter на основе изменений в объекте DataSet или DataTable.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился создавать приложения для записи данных в 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. **Ответы на контрольные вопросы**
   1. Для чего используются параметры в командах?

Ответ: Параметры в командах используются для передачи значений в SQL-запросы или хранимые процедуры. Параметры помогают избежать SQL-инъекций, обеспечивая безопасное выполнение запросов, а также упрощают работу с динамическими данными.

Основные причины использования параметров:

Безопасность: Предотвращение SQL-инъекций путем использования параметризованных запросов.

Удобство: Упрощение управления динамическими значениями в запросах.

Производительность: Повышение производительности за счет повторного использования подготовленных (precompiled) команд с параметрами.

Читаемость и поддерживаемость: Упрощение чтения и поддерживания кода за счет использования параметров вместо динамически построенных строк запросов

* 1. Как добавить параметр в команду, используя Add?

Ответ: Для добавления параметра в команду можно использовать метод Add коллекции Parameters объекта SqlCommand. Метод Add позволяет настроить тип данных, размер и другие свойства параметра.

* 1. Как добавить параметр в команду, используя AddWithValue?

Ответ: Метод AddWithValue является упрощенным способом добавления параметров в команду. Он автоматически определяет тип данных на основе переданного значения.

* 1. Какие свойства можно указать у параметра?

Ответ: При создании параметра и добавлении его в команду можно указать несколько свойств, которые определяют его поведение и тип данных. Основные свойства параметра включают:

ParameterName: Имя параметра, используемое в SQL-запросе.

SqlDbType: Тип данных SQL, который соответствует типу данных в базе данных (например, SqlDbType.Int, SqlDbType.NVarChar).

Size: Размер данных для типов данных переменной длины (например, nvarchar, varchar).

Value: Значение параметра, которое будет передано в запрос.

Direction: Направление параметра (например, Input, Output, InputOutput, ReturnValue).

IsNullable: Указывает, может ли параметр принимать значение NULL.

Scale: Масштаб параметра для числовых типов данных с фиксированной точностью.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился выполнять запросы к БД из клиентского приложения,
   2. В ходе проделанной лабораторной работы я научился передавать параметры в запросы.

# Лабораторная работа №48

**Создание хранимых процедур**

1. **Цель работы**
   1. Научиться создавать и использовать хранимые процедуры в MS SQL Server;
   2. Закрепить навык создания запросов на выборку и модификацию данных в MS SQL Server
2. **Ответы на контрольные вопросы**
   1. Что такое хранимые процедуры и для чего они применяются?

Ответ: Хранимая процедура — это предварительно скомпилированный набор инструкций SQL, который сохраняется в базе данных и может быть вызван из клиентского приложения. Она является частью базы данных и хранится в системных таблицах базы данных.

Хранимые процедуры применяются для выполнения различных операций в базе данных. Они могут использоваться для выполнения сложных запросов, манипуляций с данными, обновлений, вставок, удалений и других операций. Преимущества хранимых процедур включают повышение производительности, улучшение безопасности, повторное использование кода и упрощение администрирования базы данных.

* 1. Как вызвать выполнение хранимой процедуры в клиентском приложении?

Ответ: Для вызова выполнения хранимой процедуры в клиентском приложении можно использовать объект SqlCommand, указав в свойстве CommandType значение StoredProcedure и указав имя хранимой процедуры в свойстве CommandText

* 1. Как задать входные параметры хранимой процедуры?

Ответ: Для задания входных параметров хранимой процедуры в объекте SqlCommand можно использовать объекты SqlParameter, указав их в свойстве Parameters. Каждый параметр должен быть добавлен в коллекцию параметров и указан соответствующий тип данных, имя и значение

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился создавать и использовать хранимые процедуры в MS SQL Server;
   2. В ходе проделанной лабораторной работы я закрепил навык создания запросов на выборку и модификацию данных в MS SQL Server

# Практическая работа №1

**Разработка приложений для обработки файлов**

1. **Цель работы**
   1. Научиться применять классы для работы с файлами в приложениях на C#.
2. **Ответы на контрольные вопросы**
   1. В чем отличие между классами Directory и DirectoryInfo?

Ответ: Классы Directory и DirectoryInfo выполняют схожие функции по работе с каталогами (директориями), но различаются по способу использования и архитектуре:

**Directory**: Это статический класс, предоставляющий методы для работы с каталогами. Поскольку он статический, его методы вызываются напрямую, без создания экземпляра класса. Например, для создания каталога используется метод Directory.CreateDirectory(path).

**DirectoryInfo**: Это класс, экземпляры которого создаются для представления конкретного каталога. Экземпляры класса DirectoryInfo содержат информацию о каталоге и позволяют выполнять операции над ним. Например, для создания каталога используется метод экземпляра Create(): new DirectoryInfo(path).Create().

* 1. В чем отличие между классами File и FileInfo?

Ответ: Классы File и FileInfo также выполняют схожие функции по работе с файлами, но различаются по способу использования и архитектуре:

* **File**: Это статический класс, предоставляющий методы для работы с файлами. Поскольку он статический, его методы вызываются напрямую, без создания экземпляра класса. Например, для создания файла используется метод File.Create(path).
* **FileInfo**: Это класс, экземпляры которого создаются для представления конкретного файла. Экземпляры класса FileInfo содержат информацию о файле и позволяют выполнять операции над ним. Например, для создания файла используется метод экземпляра Create(): new FileInfo(path).Create().
  1. Как получить список файлов и папок определенного каталога?

Ответ: Для получения списка файлов и папок определенного каталога можно использовать классы Directory и DirectoryInfo.

string path = @"C:\ExampleDirectory";

string[] files = Directory.GetFiles(path);

string[] directories = Directory.GetDirectories(path);

DirectoryInfo directoryInfo = new DirectoryInfo(@"C:\ExampleDirectory");

FileInfo[] files = directoryInfo.GetFiles();

DirectoryInfo[] directories = directoryInfo.GetDirectories();

* 1. Какие свойства класса FileInfo позволяют получить информацию о файле?

Ответ: Класс FileInfo предоставляет различные свойства, которые позволяют получить информацию о файле.

**Name**: Имя файла

string fileName = fileInfo.Name;

**FullName**: Полный путь к файлу.

string fullPath = fileInfo.FullName;

**Extension**: Расширение файла.

string extension = fileInfo.Extension;

**DirectoryName**: Путь к каталогу, в котором находится файл.

string directoryName = fileInfo.DirectoryName;

**Length**: Размер файла в байтах.

long fileSize = fileInfo.Length;

1. **Вывод**
   1. В ходе проделанной практической работы я научился применять классы для работы с файлами в приложениях на C#.

# Практическая работа №2

**Работа с табличными файлами**

1. **Цель работы**
   1. Научиться выполнять создание и редактирование табличных документов на C#.
2. **Ответы на контрольные вопросы**
   1. Какое пространство имен требуется подключить для работы с Excel?

Ответ: Для работы с Excel в программе на C# необходимо подключить пространство имен Microsoft.Office.Interop.Excel. Это пространство имен предоставляется библиотекой Microsoft.Office.Interop.Excel.dll, которую можно добавить через NuGet пакет Microsoft.Office.Interop.Excel.

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

Ответ: Через данный код: Application excelApp = new Application();

* 1. Что такое Workbooks?

Ответ: Workbooks — это коллекция всех открытых рабочих книг (файлов Excel) в экземпляре приложения Excel. Она позволяет управлять открытыми книгами, создавать новые или открывать существующие.

* 1. Что такое Worksheets?

Ответ: Worksheets — это коллекция всех рабочих листов (таблиц) в определенной рабочей книге (Workbook). Позволяет обращаться к отдельным листам, добавлять новые или удалять существующие.

* 1. Что такое Range?

Ответ: Range — это объект, представляющий ячейку или диапазон ячеек на рабочем листе Excel. С помощью объекта Range можно управлять содержимым, форматированием и свойствами ячеек.

* 1. Что такое Cells?

Ответ: Cells — это свойство рабочего листа (Worksheet), представляющее собой все ячейки на листе. Оно позволяет обращаться к отдельным ячейкам по их индексам (строка и столбец).

* 1. Как получить доступ к значению ячейки и диапазона?

Ответ: Для получения и изменения значения ячейки или диапазона ячеек используются свойства Value или Value2 объекта Range.

1. **Вывод**
   1. В ходе проделанной практической работы я научился выполнять создание и редактирование табличных документов на C#.

# Практическая работа №3

**Работа с текстовыми файлами**

1. **Цель работы**
   1. Научиться выполнять создание и редактирование текстовых документов на C#.
2. **Ответы на контрольные вопросы**
   1. Какое пространство имен требуется подключить для работы с Word?

Ответ: Для работы с Word в программе на C# необходимо подключить пространство имен Microsoft.Office.Interop.Word. Это пространство имен предоставляется библиотекой Microsoft.Office.Interop.Word.dll, которую можно добавить через NuGet пакет Microsoft.Office.Interop.Word.

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

Ответ: Через данный код: Application wordApp = new Application();

* 1. Что такое Documents?

Ответ: Documents — это коллекция всех открытых документов в экземпляре приложения Word. Она позволяет управлять открытыми документами, создавать новые или открывать существующие.

* 1. Что такое Range?

Ответ: Range — это объект, представляющий непрерывный диапазон текста в документе Word. С помощью объекта Range можно управлять содержимым, форматированием и свойствами текста.

* 1. Что такое Selection?

Ответ: Selection — это объект, представляющий текущий выделенный текст или позицию курсора в документе Word. С помощью объекта Selection можно управлять выделением текста, изменять его содержимое и форматирование.

* 1. Что такое Paragraphes?

Ответ: Paragraphs — это коллекция всех абзацев в документе Word или в определенном объекте Range. Позволяет обращаться к отдельным абзацам, добавлять новые или удалять существующие.

* 1. Что такое Tables?

Ответ: Tables — это коллекция всех таблиц в документе Word или в определенном объекте Range. Позволяет управлять таблицами, добавлять новые или удалять существующие.

1. **Вывод**
   1. В ходе проделанной практической работы я научиться выполнять создание и редактирование текстовых документов на C#

# Практическая работа №4

**Сохранение настроек приложения**

1. **Цель работы**
   1. Научиться сохранять настройки в клиентском приложении на C#.
2. **Ответы на контрольные вопросы**
   1. Как добавить настройки в приложение на C#?

Ответ:

* 1. Открыть проект в Visual Studio.
  2. Добавить файл настроек:

1. Щелкните правой кнопкой мыши на проекте в Solution Explorer.
2. Выберите "Properties".Перейдите на вкладку "Settings".
3. Если вкладка "Settings" отсутствует, создайте новый файл настроек:
   1. Щелкните правой кнопкой мыши на проекте.
   2. Выберите "Add" -> "New Item...".
   3. Выберите "Settings File" и назовите его, например, Settings.settings.

3) Добавить параметры в файл настроек:

На вкладке "Settings" добавьте параметры, задав имя, тип и значение по умолчанию для каждого параметра.

* 1. Как программно считать значение параметра из настроек?

Ответ: Используется данный код:

var settingValue = Properties.Settings.Default.YourSettingName;

* 1. Как программно изменить значение параметра в настройках?

Ответ: Используется данный код:

Properties.Settings.Default.YourSettingName = newValue;

* 1. Как выполнить сохранение значений параметров в настройках?

Ответ: Используется данный код:

Properties.Settings.Default.Save();

1. **Вывод**
   1. В ходе проделанной практической работы я научился сохранять настройки в клиентском приложении на C#.

# Практическая работа №5

**Создание пользовательских элементов управления**

1. **Цель работы**
   1. Научиться создавать пользовательские элементы управления в приложении WPF.
2. **Ответы на контрольные вопросы**
   1. Для чего применяется ControlTemplate?

Ответ: ControlTemplate применяется для определения или изменения визуального представления элемента управления в WPF (Windows Presentation Foundation). С его помощью можно полностью переопределить визуальный вид элемента управления, при этом сохранив его функциональность. Например, можно изменить внешний вид кнопки, текстового поля и других элементов управления, не затрагивая их логику.

* 1. Где может быть описан ControlTemplate?

Ответ: Может быть описан в различных местах:

1. Ресурсы элемента управления
2. Ресурсы окна (Window) или пользовательского элемента управления (UserControl)
3. **Ресурсы приложения (App.xaml)**
4. **В отдельных файлах ресурсов (ResourceDictionary)**
   1. Для чего применяется UserControl?

Ответ: UserControl применяется для создания составных элементов управления в WPF, которые могут состоять из других базовых элементов управления. Это позволяет инкапсулировать сложную функциональность и повторно использовать её в разных частях приложения. Например, можно создать пользовательский элемент управления для адресной строки или панели инструментов.

* 1. Каков алгоритм создания пользовательского элемента управления?

Ответ: Алгоритм:

1. Создать новый UserControl:
2. В Visual Studio щелкните правой кнопкой мыши на проекте.
3. Выберите "Add" -> "New Item...".
4. Выберите "UserControl (WPF)" и задайте имя, например, MyUserControl.
5. Определить визуальное представление:
6. Откройте созданный XAML-файл (например, MyUserControl.xaml).
7. Определите визуальные элементы и их размещение внутри UserControl.
8. Определить логику:
9. Откройте код-бехайнд файл (например, MyUserControl.xaml.cs).
10. Определите логику пользовательского элемента управления.
11. Использовать пользовательский элемент управления в других частях приложения:
12. Импортируйте пространство имен и используйте UserControl в XAML других окон или элементов управления.
    1. Как программно создать обработчик события?

Ответ: Для программного создания обработчика события в C# необходимо выполнить следующие шаги:

1. **Создать метод-обработчик события**:
2. **Привязать метод-обработчик к событию**:
   1. Для чего используется свойство зависимости?

Ответ:

1. **Привязки данных (Data Binding):** Позволяет привязывать значения свойств к источникам данных, обеспечивая автоматическое обновление интерфейса пользователя при изменении данных.
2. **Анимации:** Позволяет анимировать значения свойств.
3. **Стилей**: Позволяет задавать и применять стили и шаблоны к элементам управления.
4. **Изменения значений по умолчанию**: Установка значений свойств по умолчанию через стили и шаблоны.
5. **Эффективного использования памяти**: Свойства зависимости реализованы так, чтобы занимать память только при необходимости (при установке значения, отличного от значения по умолчанию).
6. **Вывод**
   1. В ходе проделанной практической работы я научился создавать пользовательские элементы управления в приложении WPF.

# Практическая работа №6

**Привязка данных**

1. **Цель работы**
   1. Научиться выполнять привязку данных в приложении WPF.
2. **Ответы на контрольные вопросы**
   1. Что такое «привязка данных»?

Ответ: Привязка данных (data binding) в контексте WPF (Windows Presentation Foundation) — это механизм, который позволяет синхронизировать свойства элементов пользовательского интерфейса (UI) с данными из различных источников, таких как объекты, коллекции или базы данных. Привязка данных упрощает обновление интерфейса при изменении данных и наоборот, обеспечивая автоматическую передачу данных между источником и целевым объектом.

* 1. Каков шаблон настройки привязки данных?

Ответ: Привязка данных в WPF настраивается с использованием объекта Binding. Привязка может быть настроена как в XAML, так и в коде. Вот базовый шаблон настройки привязки данных:

* 1. Какой интерфейс надо реализовать для создания конвертера значений?

Ответ: Для создания конвертера значений в WPF необходимо реализовать интерфейс IValueConverter. Этот интерфейс содержит два метода: Convert и ConvertBack, которые используются для преобразования данных из одного типа в другой и обратно.

* 1. Какой интерфейс надо реализовать для валидации данных при привязке?

Ответ: Для валидации данных при привязке в WPF необходимо реализовать интерфейс IDataErrorInfo или INotifyDataErrorInfo.

#### IDataErrorInfo:

Этот интерфейс предоставляет простой способ для валидации данных. Он содержит два свойства: Error и this[string columnName].

1. **Вывод**
   1. В ходе проделанной практической работы я научился выполнять привязку данных в приложении WPF.