**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**по МДК.01.01**

Студент: С.Е. Герасименко 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 данных, позволяющая хранить коллекцию элементов одного типа под одним именем и определенным индексом.

* 1. Как описывается одномерный массив?

Ответ: одномерный массив описывается следующим образом: int[] Array = new int[];

* 1. Как обратиться к некоторому элементу одномерного массива?

Ответ: Для обращения к элементу массива: названиеЭлемента[индекс];

* 1. Как можно задать одномерный массив?

Ответ: тип[] Название = {элемент1, элемент2, элемент3};

* 1. Что такое «сортировка»?

Ответ: Сортировка – это процесс упорядочивания какого-то количества элементов по определенным условиям.

* 1. Что такое «алгоритм сортировки»?

Ответ: Алгоритм сортировки – это набор действий, необходимый для упорядочивания всех элементов.

* 1. Какие виды сортировки массивов существуют?

Ответ: Существуют следующие виды сортировки массивов: метод пузырька (Bubble Sort), простым выбором (Selection Sort), методом вставки (Insertion 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: Существуют следующие виды поиска элементов массивов: алгоритм линейного поиска, алгоритм двоичного поиска элемента, алгоритм поиска прыжками элемента.

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

Ответ: В данном алгоритме проверка элементов массива осуществляется последовательно от начала до конца. Временная сложность составляет O(n).

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

Ответ: В данном алгоритме массив должен быть отсортирован, определяется с помощью деления массива пополам. Временная сложность составляет О(log n).

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

Ответ: В данном алгоритме осуществляются прыжки по блокам массива для сокращения диапазона поиска. Временная сложность в худшем случае: O(√n).

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 – процесс, при котором функция вызывает саму себя, непосредственно или через цепочку других функций.

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

Ответ: При реализации могут возникнуть следующие проблемы: потребление большого объема памяти из-за сохранения промежуточных результатов и возврата из рекурсивных вызовов; возможность переполнения стека из-за чрезмерной глубины рекурсии.

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

Ответ: Определение функции называется рекурсивным, если оно содержит вызов самой себя или через цепочку вызовов.

Примеры:

1. Факториал числа n: factorial(n) = n \* factorial(n-1), где factorial(n-1);
2. Возведение положительного числа x в положительную степень n: Power(x, n) = x\*Power(x, n-1);
   1. Что такое «глубина рекурсии»?

Ответ: Глубина рекурсии — это количество последовательных вызовов функции до достижения базового случая или глубины, на которой вызовы возвращаются из стека вызовов.

* 1. Что такое «рекурсивный спуск»?

Ответ: Рекурсивный спуск - метод реализации рекурсивных алгоритмов, при котором функция вызывает саму себя для решения меньших подзадач.

* 1. Что такое «рекурсивный подъём»?

Ответ: Рекурсивный подъем — это обратный процесс к рекурсивному спуску, при котором результаты подзадач объединяются при возврате из рекурсивных вызовов к базовому случаю.

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, элементы которого сами являются массивами.

* 1. Как описывается двумерный массив?

Ответ: Двумерный массив описывается как массив массивов, то есть массив, в котором каждый элемент является массивом элементов: int[][] array = new int[][];

* 1. Как обратиться к некоторому элементу двумерного массива?

Ответ: array[индекс1][индекс2];

* 1. Как узнать количество строк двумерного массива?

Ответ: Количество строк можно узнать с помощью чтения свойства массив.length.

* 1. Как узнать количество столбцов двумерного массива?

Ответ: Количество строк можно узнать с помощью чтения свойства массив0.length.

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

Ответ: Для вывода двумерного массива на консоль в виде таблицы можно использовать вложенные циклы, перебирая строки и столбцы массива и выводя элементы.

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 ИмяКласса {поля}

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

Ответ: Данные класса делятся на: поля, методы, свойства, события.

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

Ответ: Функции класса делятся на методы и конструкторы.

* 1. Для чего применяются конструкторы классов?

Ответ: Конструкторы классов применяются для инициализации объектов класса, устанавливают начальные значения полей объекта.

* 1. Что такое «цепочка конструкторов»?

Ответ: “Цепочка конструкторов” относится к вызову одного конструктора из другого конструктора того же класса. Таким образом, один конструктор может вызывать другой конструктор класса.

* 1. Для чего применяются свойства классов?

Ответ: Свойства классов используются для обеспечения доступа к полям объекта класса с контролем доступа и позволяют выполнить дополнительную логику при установке или получении значения.

* 1. Когда используются автоматически реализуемые свойства классов?

Ответ: Автоматически реализуемые свойства классов используются, когда не требуется дополнительная логика при установке или получении значения свойства, и можно задать свойство с помощью одной строки.

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

Ответ: Отличие вызова статических членов классов от членов экземпляра класса: статические члены принадлежат классу в целом, а не конкретному экземпляру класса; для доступа к статическим членам не требуется создание объекта класса.

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 - возможность определения нескольких методов с одинаковыми именами в классе, но с разными параметрами (разное количество параметров или разные типы параметров).

* 1. К какому виду полиморфизма относится перегрузка методов?

Ответ: Перегрузка методов относится к статическому (или компиляционному) полиморфизму.

* 1. Какие типы функций класса можно перегружать?

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

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

Ответ: Требования к сигнатуре перегружаемых функций класса следующие: имя метода должно быть одинаковым; количество параметров должно быть разным или типы параметров должны быть разными; параметры должны отличаться по типу, порядку или количеству.

* 1. Какие типы функций класса нельзя перегружать?

Ответ: Нельзя перегружать функции класса, если их сигнатура полностью совпадает, и различие только в возвращаемом типе метода.

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: Операторный метод определяется с помощью ключевого слова operator.

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

Ответ: Можно перегружать следующие унарные операторы:

* Отрицание -
* Инкремент ++
* Декремент --
* Логическое отрицание !
* Побитовое отрицание ~
* Префиксная и постфиксная версии унарных операторов инкремента и декремента (++ и --)
  1. Какие бинарные операторы можно перегружать в C#

Ответ: Можно перегружать следующие бинарные операторы:

* Сложение +
* Вычитание -
* Умножение \*
* Деление /
* Остаток от деления %
* Побитовое И &
* Побитовое ИЛИ |
* Побитовое исключающее ИЛИ ^
* Сдвиг влево <<
* Сдвиг вправо >>
  1. Какие операторы сравнения можно перегружать в C#?

Ответ: Можно перегружать следующие операторы сравнения:

* Равенство ==
* Неравенство !=
* Больше >
* Меньше <
* Больше или равно >=
* Меньше или равно <=
  1. Какие операторы требуется перегружать в C# попарно?

Ответ: Операторы == и != требуется перегружать попарно, то есть если перегружен ==, то должен быть также перегружен 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 в объектно-ориентированном программировании - механизм, позволяющий определить новый класс на основе уже существующего класса. Новый класс, называемый дочерним классом, наследует свойства и методы родительского класса.

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

Ответ: Дочерний класс может иметь только один родительский класс (одинарное наследование).

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

Ответ: Для обращения к реализации родительского класса из дочернего класса используется ключевое слово base.

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

Ответ: Переопределение метода - возможность дочернему классу предоставить свою собственную имплементацию метода, который был унаследован от родительского класса. Для переопределения метода используется ключевое слово override.

* 1. Что такое «абстрактный класс»?

Ответ: Абстрактный класс - класс, у которого хотя бы один метод объявлен как абстрактный с ключевым словом abstract.

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

Ответ:

* Virtual: используется для определения метода, который может быть переопределен в дочерних классах.
* Override: используется для переопределения метода из базового класса в дочернем классе.
* Abstract: используется для объявления абстрактного метода в абстрактном классе. Абстрактные методы имеют только сигнатуру и нет реализации.
* New: используется для скрытия метода из базового класса в дочернем классе, если методы совпадают по названию.

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 - контракт, который определяет методы, свойства, события и индексаторы, но не содержит их реализацию. Классы, структуры и другие интерфейсы могут реализовывать интерфейс, обеспечивая конкретную реализацию для методов и других элементов.

* 1. Чем отличается интерфейс от абстрактного класса?

Ответ: Основные отличия между интерфейсом и абстрактным классом:

Интерфейс:

1. Может содержать только сигнатуры методов, свойств, событий и индексаторов.
2. Класс может реализовывать несколько интерфейсов.
3. Интерфейсы не имеют реализации методов.
4. Абстрактный класс:
5. Может содержать как абстрактные методы, так и реализованные методы.
6. Класс может наследоваться только от одного абстрактного класса.
7. Абстрактные классы могут иметь поля и конструктор.
   1. Есть ли у класса ограничения по количеству реализуемых интерфейсов?

Ответ: Нет ограничений по количеству реализуемых интерфейсов у класса. Класс может реализовывать любое количество интерфейсов.

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

Ответ: Общая форма объявления интерфейса в C#:

interface ИмяИнтерфейса

{

//сигнатура

}

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 класса на основе их значений. Классы, которые реализуют интерфейс IComparable, могут использовать метод CompareTo для сравнения экземпляров своих объектов и определения их относительного порядка.

* 1. Для чего используется интерфейс IEquatable?

Ответ: Интерфейс IEquatable используется для определения метода Equals для проверки равенства объектов.

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

Ответ: Обобщенные интерфейсы используются для определения интерфейсов, которые могут работать с различными типами данных, задаваемыми при реализации интерфейса (IEnumerable<T>). Необобщенные интерфейсы определяют интерфейсы без учета типов данных, с которыми они будут работать (IComparable). Обобщенные интерфейсы позволяют создавать более универсальные и гибкие интерфейсы, которые могут быть использованы с различными типами данных.

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 (Collection) в C# представляет собой структуру данных, которая позволяет хранить и управлять группой объектов. Она обеспечивает удобный способ организации данных и предоставляет различные методы для доступа, добавления, удаления и изменения элементов.

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

Ответ: Пространство имен System.Collections.Generic в C# содержит классы для работы с обобщенными коллекциями. Некоторые из классов данного пространства имен включают List<T>, Dictionary<TKey, TValue>, HashSet<T> и другие.

* 1. Что такое List<T>?

Ответ: List<T> - это класс обобщенного списка в C#, который представляет собой динамический массив, позволяющий хранить элементы одного типа и обеспечивающий удобные методы для работы с этими элементами.

* 1. Как можно обратиться к элементу списка?

Ответ: Для доступа к элементу списка List<T>, можно использовать его индекс. Например, чтобы обратиться к элементу по индексу i, нужно написать listi, где list - объект List<T>.

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

Ответ: Dictionary<TKey, TValue> - это класс словаря в C#, который представляет собой коллекцию пар "ключ-значение". Позволяет быстро находить значение по ключу и обеспечивает эффективный поиск элементов.

* 1. Как можно обратиться к элементу словаря?

Ответ: Для доступа к элементу словаря Dictionary<TKey, TValue> необходимо указать ключ, по которому хотите получить значение. Например, чтобы обратиться к элементу с ключом key, можно использовать dictionarykey, где dictionary - объект Dictionary<TKey, TValue>.

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 в C# хранятся в стеке памяти.

* 1. Что такое «структура»?

Ответ: Структура (struct) в C# представляет собой пользовательский тип данных, который позволяет объединить несколько связанных переменных под одним именем.

* 1. Чем отличается структура от класса?

Ответ: Основное отличие между структурой и классом в C# заключается в том, что структуры являются типами значений, хранящимися в стеке, в то время как классы являются типами ссылок, хранящимися в куче памяти. Кроме того, структуры не поддерживают наследование, в отличие от классов.

* 1. Что такое «перечисление»?

Ответ: Перечисление (enum) в C# представляет собой именованный набор константных целочисленных значений. Это позволяет объединить несколько значений в одноцельное множество и использовать их в коде для удобства и читаемости.

* 1. Для чего используются перечисления?

Ответ: Перечисления используются для удобного хранения и использования группы связанных константных значений. Они облегчают чтение и написание кода, делая его более понятным и поддерживаемым.

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

Ответ: Общая форма объявления перечисления в C# выглядит следующим образом:

enum ИмяПеречисления

{

Значение1,

Значение2,

// и т.д.

}

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 контрольные вопросы**
   1. К какому типу переменных относятся переменные типа string?

Ответ: Переменные типа string относятся к ссылочным типам данных в C#. Однако, значение строки (сама строка) хранится в куче памяти, а ссылка на строку хранится на стеке памяти.

* 1. Какие операции допустимы над строковыми данными?

Ответ: Операции над строковыми данными в C# включают конкатенацию (объединение) строк, доступ к символам по индексу, поиск подстроки, замену подстроки, разделение строки на подстроки и другие операции сравнения и обработки строк.

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

Ответ: Переменная типа string представляет собой строку, то есть упорядоченную коллекцию символов. Массив символов же является более примитивным типом данных, представляющим набор символов в последовательной форме. Основное отличие заключается в том, что строка (string) в C# является классом, а массив символов (char) является массивом.

* 1. Что такое «интерполяция строк»?

Ответ: "Интерполяция строк" в C# - это синтаксис, который позволяет встраивать выражения в строку, обрамляя их символами "{ }". Это делает строку более читаемой и удобной для работы с переменными и выражениями. Например:

* 1. Изменяют ли методы класса String исходную строку?

Ответ: Методы класса String в C# не изменяют исходную строку (представленную переменной). Вместо этого они создают и возвращают новую строку, основанную на исходной операции. строки в C# являются неизменяемыми (immutable), поэтому любые операции, изменяющие строку, создают новую строку на основе существующей.

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 (или regex) — это последовательность символов, используемая для поиска и сопоставления шаблонов в тексте.

* 1. Для чего используются регулярные выражения?

Ответ: Регулярные выражения используются для поиска, сопоставления, разделения, замены и обработки текстовой информации по определенным шаблонам. Они позволяют выполнять сложные операции со строками, повторять и комбинировать символы и шаблоны для поиска нужных данных.

* 1. Для чего используется класс Regex?

Ответ: Класс Regex в C# используется для работы с регулярными выражениями. Он предоставляет методы для компиляции, проверки и применения регулярного выражения к строковым данным.

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

Ответ: Алгоритм поиска подстроки при помощи регулярного выражения включает сопоставление шаблона регулярного выражения с текстом и выявление всех вхождений этого шаблона в тексте.

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

Ответ: Алгоритм замены подстроки при помощи регулярного выражения включает поиск всех вхождений шаблона регулярного выражения в тексте и замену их на указанную строку.

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

Ответ: В регулярных выражениях escape-символы используются для обозначения специальных символов, таких как точка (.), звездочка (), скобки и другие. Например, точка в регулярном выражении обозначает любой символ, но если нужно найти именно символ точки, то его следует экранировать с помощью обратного слеша: \.

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

Ответ: Классы символов в регулярных выражениях применяются для описания групп символов, которые могут быть использованы в шаблоне. Например, [a-z] соответствует любой строчной букве от a до z.

* 1. Для чего в регулярных выражениях применяются квантификаторы?

Ответ: Квантификаторы в регулярных выражениях позволяют указать количество вхождений символов или групп символов. Например, звездочка () обозначает ноль или более вхождений, плюс (+) обозначает одно или более вхождений, вопросительный знак (?) обозначает ноль или одно вхождение.

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 (Delegate) в C# представляет собой тип данных, который ссылается на методы с определенной сигнатурой. Делегаты позволяют передавать методы как параметры, хранить ссылку на методы и вызывать методы через делегат.

* 1. Как объявить делегат на C#?

Ответ: Для объявления делегата в C# используется ключевое слово delegate, за которым следует возвращаемый тип, название делегата и параметры, если они есть. Например:

delegate void MyDelegate(int x, int y);

* 1. Какие способы вызова метода через делегат существуют?

Ответ: Существуют несколько способов вызова метода через делегат:

* Присваивание метода делегату и вызов через делегат.
* Операция вызова () делегата, которая автоматически вызывает связанный метод.
* Использование методов BeginInvoke и EndInvoke для асинхронного вызова метода через делегат.
  1. Какие встроенные делегаты имеются в C# и для чего они используются?

Ответ: В C# есть несколько встроенных делегатов, используемых для различных целей:

* Action - делегат без возвращаемого значения с параметрами (может использоваться для передачи метода, который не возвращает результат).
* Func - делегат с возвращаемым значением и параметрами (используется для передачи метода, который возвращает результат).
* Predicate - делегат, принимающий один параметр и возвращающий булево значение (используется для проверки условий).
* Множество других встроенных делегатов, предназначенных для различных целей.

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 (Event) представляет собой механизм, который позволяет классам уведомлять другие классы о том, что произошло определенное действие. Класс, который инициирует событие, называется издателем (publisher), а класс, который обрабатывает событие, называется подписчиком (subscriber).

* 1. Как объявить событие на C#?

Ответ: Для объявления события в C# используется ключевое слово event, за которым идет тип делегата и название события. Например:

public event EventHandler MyEvent;

* 1. Как создать обработчик события?

Ответ: Для создания обработчика события необходимо создать метод с соответствующей сигнатурой делегата события и подписать этот метод на событие. Например:

public void EventHandlerMethod(object sender, EventArgs e)

{

// Обработка события

}

MyEvent += EventHandlerMethod;

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

Ответ: Класс EventArgs является родительским для всех классов данных события в C#. Этот класс предоставляет базовую функциональность для передачи данных от издателя события к подписчику.

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

Ответ: Классы делегатов, которые являются стандартными для создания событий в C#, это Action и EventHandler. Action используется для событий без передачи дополнительных данных, а EventHandler используется для событий, которые могут передавать данные в виде цкласса EventArgs.

1. **Вывод**
   1. В ходе проделанной лабораторной работы я научился создавать, вызывать и обрабатывать события на C#.

# 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 интегрированный языковой запрос в C#, который позволяет создавать выражения, выполняющие запросы к источникам данных, таким как коллекции объектов, базы данных и другие. LINQ предоставляет возможность писать запросы, которые похожи на SQL, но написанные на языке программирования.

* 1. Что такое «анонимный тип»?

Ответ: Анонимный тип (Anonymous Type) - это тип данных в C#, который создается автоматически компилятором на основе значений, переданных в выражении инициализации. Анонимные типы используются для представления сложных типов данных без явного объявления нового класса или структуры.

* 1. Для чего используется var?

Ответ: Ключевое слово var используется в C# для неявного объявления переменной, когда тип переменной выводится компилятором на основе выражения. var упрощает код, делая его более читаемым и позволяет избежать дублирования типов данных в коде.

* 1. Для чего используется метод Select?

Ответ: Метод Select в LINQ используется для проекции (выбора) определенных значений или свойств объектов из коллекции. Он позволяет создавать новую последовательность, которая содержит только выбранные данные из исходной коллекции.

* 1. Для чего используются методы OrderBy, OrderByDescending?

Ответ: Методы OrderBy и OrderByDescending в LINQ используются для сортировки элементов в коллекции по возрастанию или убыванию соответственно. OrderBy сортирует элементы по возрастанию, а OrderByDescending - по убыванию.

* 1. Для чего используются методы ThenBy, ThenByDescending?

Ответ: Методы ThenBy и ThenByDescending в LINQ используются для добавления дополнительных условий сортировки после OrderBy или OrderByDescending. Они позволяют сортировать элементы по вторичному и последующим ключам после первичной сортировки.

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 программе, их свойств и методов, а также взаимосвязей между классами.

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

Ответ: Описание типа в диаграмме классов состоит из имени типа, атрибутов (поля и свойства), методов и их типов доступа.

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

Ответ: Взаимосвязи в диаграмме классов делятся на три группы: наследование, ассоциация и общий интерфейс.

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

Ответ: Взаимосвязь "наследование" означает, что один класс наследует свойства и методы другого класса, при этом обеспечивается иерархия классов. В отличие от этого, взаимосвязь "реализация" указывает на то, что класс реализует интерфейс, предоставляя конкретную реализацию его методов.

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

Ответ: Взаимосвязь "агрегация" обозначает, что один объект содержит или ссылается на другой объект, но они могут существовать независимо друг от друга. В отличие от этого, взаимосвязь "композиция" подразумевает, что один объект является частью другого объекта, и они живут в тесной зависимости друг от друга.

* 1. Что показывает взаимосвязь «ассоциация»?

Ответ: Взаимосвязь "ассоциация" показывает связь или отношение между двумя классами, но без указания наличия четкой зависимости или вложенности.

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

Ответ: В диаграмме классов могут быть указаны виды мощности отношений, такие как ассоциация, композиция, агрегация, наследование, использование. Каждый вид отношений имеет свои специфические характеристики.

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 создания объектов с определенными свойствами и методами, сосредотачиваясь на упрощении процесса создания объектов, управлении зависимостями и их конфигурацией.

* 1. Какие паттерны относятся к порождающим?

Ответ: К порождающим паттернам относятся: Builder, Factory Method, Abstract Factory, Prototype, Singleton.

* 1. Для чего используются структурные паттерны?

Ответ: Структурные паттерны используются для организации классов и объектов с целью формирования более сложных структур данных.

* 1. Какие паттерны относятся к структурным?

Ответ: К структурным паттернам относятся: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy.

* 1. Для чего используются поведенческие паттерны?

Ответ: Поведенческие паттерны используются для организации взаимодействия между объектами, определяя способы реализации коммуникации и распределения обязанностей.

* 1. Какие паттерны относятся к поведенческим?

Ответ: К поведенческим паттернам относятся: Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, 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 стиль архитектуры программного обеспечения для распределенных систем, основанный на принципах описанных в докладе Роя Филдинга "О проектировании сетевой архитектуры программного обеспечения".

* 1. Для чего используется метод GET?

Ответ: Метод GET используется для запроса данных или информации с сервера. Он используется для получения ресурсов в RESTful веб-приложениях.

* 1. Для чего используется метод POST?

Ответ: Метод POST используется для отправки данных на сервер для обработки. Он обычно используется при создании новых ресурсов на сервере в RESTful веб-приложениях.

* 1. Для чего используется метод PUT?

Ответ: Метод PUT используется для обновления или создания ресурса с определенным URI на сервере. Он обновляет или создает ресурс по указанному URI в RESTful веб-приложениях.

* 1. Для чего используется метод DELETE?

Ответ: Метод DELETE используется для удаления ресурса с сервера по указанному URI. Он удаляет ресурс, указанный в запросе, в RESTful веб-приложениях.

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 внесения изменений в код с целью улучшения его производительности, эффективности использования ресурсов и/или улучшения читаемости.

* 1. Какова цель оптимизации программного кода?

Ответ: Цель оптимизации программного кода - повышение производительности, уменьшение потребления ресурсов (памяти, процессорного времени), улучшение отзывчивости приложения.

* 1. Какие методы оптимизации программного кода применяются?

Ответ: Методы оптимизации программного кода могут включать в себя использование более эффективных алгоритмов, уменьшение сложности кода, избегание избыточных операций, оптимизацию циклов и т.д.

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 в C# используются ключевые слова async и await.

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

Ответ: Типы возврата у асинхронных методов могут быть void, Task и Task<T>. Void используется, когда метод не возвращает значение, Task используется для асинхронной операции без возвращаемого значения, а Task<T> - для асинхронной операции с возвращаемым значением.

* 1. Как вызвать метод в асинхронном режиме?

Ответ: Метод можно вызвать в асинхронном режиме, добавив ключевое слово await перед вызовом асинхронного метода.

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

Ответ: Для указания, что в методе могут быть асинхронные вызовы, метод должен содержать ключевое слово async в своем объявлении.

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

Ответ: Исключения, возникшие в асинхронных вызовах, можно обработать с помощью блока try-catch вокруг асинхронного вызова или с помощью обработки исключения внутри асинхронного метода.

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 коде, содержащие информацию о типах, методах, свойствах и т.д., которая автоматически извлекается в виде документации.

* 1. Как сгенерировать XML-комментарий?

Ответ: Для генерации XML-комментариев в Visual Studio можно использовать специальные теги перед определением классов, методов, свойств и т.д., начинающиеся с символа "///".

* 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 XML-файл документации и предоставить его вместе с библиотекой.

* 1. Что пишется в разделе summary?

Ответ: В разделе summary описывается краткое описание элемента, например, метода или свойства.

* 1. Что пишется в разделе param?

Ответ: В разделе param описываются параметры метода, их ожидаемые значения и использование.

* 1. Что пишется в разделе returns?

Ответ: В разделе returns описывается возвращаемое значение 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 программного кода с целью улучшения его читаемости, понимания и поддержки без изменения внешнего поведения.

* 1. Какие группы техник рефакторинга существуют?

Ответ: Группы техник рефакторинга включают в себя техники упрощения кода, улучшения архитектуры, выделения повторяющихся фрагментов кода и т.д.

* 1. Как выполнить рефакторинг в Visual Studio?

Ответ: В Visual Studio можно выполнить рефакторинг с помощью встроенных инструментов, таких как CodeRush, ReSharper, CodeMaid и др.

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, обычно используемое для управления версиями кода и совместной работы над проектами.

* 1. Что указывается в файле readme.md?

Ответ: В файле readme.md обычно указывается описание проекта, инструкции по установке, использованию и взаимодействию с проектом.

* 1. Что указывается в файле .gitignore?

Ответ: В файле .gitignore указываются файлы и папки, которые не должны быть отслеживаемыми системой контроля версий Git.

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

Ответ: Для управления git-репозиторием может использоваться программное обеспечение, такое как GitKraken, SourceTree, GitHub Desktop и др.

* 1. Где может располагаться репозиторий?

Ответ: Репозиторий может располагаться на локальной машине или на удаленном сервере, например, на GitHub, GitLab, Bitbucket и др.

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" в WPF — это способ организации элементов управления на пользовательском интерфейсе.

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

Ответ: Класс Grid является родительским для всех элементов-контейнеров в WPF.

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

Ответ: Элементы внутри контейнера можно выровнять по высоте и ширине с помощью свойств HorizontalAlignment и VerticalAlignment.

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

Ответ: Особенность компоновки с использованием Grid в том, что элементы могут быть размещены в ячейках таблицы, а с использованием Canvas - абсолютное позиционирование элементов.

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

Ответ: StackPanel располагает элементы в стеке, DockPanel позволяет привязывать элементы к краям контейнера, а WrapPanel автоматически переносит элементы на новую строку или столбец при достижении границы.

* 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, проценты, дюймы, точки и т.д.

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: Элемент управления Frame используется для отображения содержимого страниц в WPF-приложениях.

* 1. Для чего используется элемент управления Page?

Ответ: Элемент управления Page представляет собой отдельную страницу приложения, которая может содержать свое собственное содержимое.

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

Ответ: Для перехода к определенной странице с использованием фрейма можно использовать метод Navigate() и передать URI или объект страницы.

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

Ответ: Чтобы проверить возможность вернуться к предыдущей странице в фрейме, можно использовать метод CanGoBack.

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

Ответ: Для перехода к предыдущей странице с использованием фрейма можно использовать метод GoBack().

* 1. Какие элементы позволяют сгруппировать содержимое?

Ответ: Элементы, позволяющие сгруппировать содержимое, включают в себя контейнеры, такие как StackPanel, Grid, WrapPanel и др.

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 приложениях WPF используются для унификации внешнего вида элементов управления.

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

Ответ: Общая форма локального определения стиля элемента управления включает в себя присвоение ключа стилю и описание его свойств.

* 1. Какова общая форма определения стиля приложения?

Ответ: Общая форма определения стиля приложения включает в себя объявление стиля в ресурсах приложения без ключа.

* 1. Как указать явное использование стилей?

Ответ: Явное использование стилей осуществляется путем применения стиля к элементу с помощью его ключа.

* 1. Как указать наследование стиля?

Ответ: Наследование стиля можно указать с помощью свойства BasedOn, указав ключ стиля, от которого нужно наследоваться.

* 1. Как добавить новую тему в приложение?

Ответ: Для добавления новой темы в приложение можно создать новый ресурсный словарь с описанием стилей для элементов.

* 1. Как выполнить переключение между темами?

Ответ: Для переключения между темами в приложении можно изменять ресурсные словари, содержащие стили элементов.

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 приложениях WPF позволяют реагировать на изменения свойств, данных или событий элементов управления.

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

Ответ: В приложениях WPF можно разработать триггеры свойств, триггеры данных и триггеры событий.

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

Ответ: Триггер свойств используется для изменения внешнего вида элемента при изменении его свойств.

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

Ответ: Триггер данных используется для реакции на изменения данных, например, в элементе управления.

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

Ответ: Триггер событий используется для выполнения действий при возникновении определенных событий.

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, представляющий флажок, который можно выбирать или снимать.

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

Ответ: RadioButton - это элемент управления, представляющий переключатель, который позволяет выбрать один из нескольких вариантов.

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

Ответ: Чтобы проверить, что флажок или переключатель выбран, можно использовать свойство IsChecked.

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

Ответ: При выборе флажка или переключателя срабатывает событие Checked.

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

Ответ: При снятии выбора флажка или переключателя срабатывает событие Unchecked.

* 1. Какие значения могут принимать флажки и переключатели?

Ответ: Флажки и переключатели могут принимать значения true, false и Intermediate.

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: Для указания имени элемента управления в WPF используется атрибут Name или x:Name.

* 1. Как создать обработчик события в WPF?

Ответ: Для создания обработчика события в WPF можно использовать атрибуты XAML или подключать обработчики событий в коде C#.

* 1. Для чего используется Slider в WPF?

Ответ: Slider в WPF используется для выбора значения из диапазона.

* 1. Для чего используется TextBox в WPF?

Ответ: TextBox в WPF используется для ввода и отображения текстовой информации.

* 1. Для чего используется TextBlock в WPF?

Ответ: TextBlock в WPF используется для отображения текстовой информации без возможности редактирования.

* 1. Для чего используется Calendar в WPF?

Ответ: Calendar в WPF используется для отображения и выбора даты.

* 1. Для чего используется DatePicker в WPF?

Ответ: DatePicker в WPF используется для выбора даты из календаря с автоматическим заполнением текстового поля.

* 1. Для чего используется PasswordBox в WPF?

Ответ: PasswordBox в WPF используется для ввода и отображения пароля, текст скрыт и неотображаем.

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 в WPF — это комбинированное поле, которое позволяет пользователю выбрать одну опцию из выпадающего списка.

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

Ответ: ListBox в WPF используется для отображения списка элементов, из которых пользователь может выбрать один или несколько.

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

Ответ: При выборе элемента в селекторе срабатывает событие SelectionChanged.

* 1. В каком свойстве хранятся элементы селекторов?

Ответ: Элементы селекторов хранятся в свойстве Items.

* 1. Какого типа элементы могут быть в селекторе?

Ответ: Элементы в селекторе могут быть различных типов, включая строки, объекты, изображения и другие элементы управления.

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

Ответ: Для привязки селектора к набору данных используется свойство ItemsSource.

* 1. Для чего используется свойство DisplayMemberPath в селекторе?

Ответ: Свойство DisplayMemberPath в селекторе используется для задания пути к полю, которое будет отображаться в элементе выбора.

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 в WPF используется для отображения и редактирования табличных данных.

* 1. Какие типы столбцов поддерживаются в DataGrid?

Ответ: В DataGrid поддерживаются типы столбцов: DataGridTextColumn, DataGridCheckBoxColumn, DataGridHyperlinkColumn и другие.

* 1. Как добавить кнопку в строки DataGrid?

Ответ: Для добавления кнопки в строки DataGrid можно использовать шаблоны столбцов (TemplateColumn) и разместить в них кнопку.

* 1. Как указать источник данных для DataGrid?

Ответ: Источник данных для DataGrid можно указать через свойство ItemsSource.

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

Ответ: Источник данных для выпадающего списка в DataGrid также указывается через свойство ItemsSource.

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 в WPF используется для создания меню приложения.

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

Ответ: ContextMenu в WPF используется для создания контекстного меню, которое появляется при щелчке правой кнопкой мыши на элементе управления.

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

Ответ: ToolBar в WPF используется для создания панели инструментов с кнопками и другими элементами управления для быстрого доступа к функциям.

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

Ответ: StatusBar в WPF используется для отображения информации о состоянии приложения, как правило, 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 специальное окно, которое появляется для взаимодействия с пользователем или для получения информации от пользователя в процессе выполнения программы.

* 1. Для чего используется OpenFileDialog?

Ответ: OpenFileDialog используется для открытия диалогового окна выбора файла, позволяет пользователю выбрать файл на компьютере.

* 1. Для чего используется SaveFileDialog?

Ответ: SaveFileDialog используется для открытия диалогового окна сохранения файла, в котором пользователь может указать путь и имя файла для сохранения.

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

Ответ: MessageBox — это стандартное диалоговое окно, используемое для отображения сообщений пользователю. Можно задать текст сообщения, заголовок окна, кнопки (например, Да/Нет), иконку и другие опции.

* 1. Для чего используется PrintDialog?

Ответ: PrintDialog используется для открытия диалогового окна печати, где пользователь может выбрать принтер, настроить параметры печати и отправить документ на печать.

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 этого можно использовать свойства и методы диалогового окна, например, передать данные через параметры методов, или обращаться к свойствам окна после его закрытия.

* 1. Какие значения может принимать переменная DialogResult?

Ответ: Переменная DialogResult может принимать значения OK, Cancel, Yes, No или другие, в зависимости от результата диалогового окна.

* 1. Как открыть окно в диалоговом режиме?

Ответ: Для открытия окна в диалоговом режиме нужно вызвать метод ShowDialog() вместо Show().

* 1. Как открыть окно в не диалоговом режиме?

Ответ: Для открытия окна в не диалоговом режиме нужно вызвать метод Show().

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

Ответ: Диалоговый режим блокирует другие окна пользователя, пока диалоговое окно открыто, в то время как не диалоговый режим позволяет пользователю продолжать взаимодействие с другими окнами при открытом окне.

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 данных по определенному условию.

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

Ответ: Логические операторы AND, OR, NOT.

* 1. Как выполнить регистронезависимый поиск?

Ответ: Для выполнения регистронезависимого поиска можно использовать методы ToLower() или ToUpper(), чтобы привести все символы к одному регистру перед сравнением.

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

Ответ: Можно использовать метод StartsWith().

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

Ответ: Можно использовать метод Contains().

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

Ответ: Можно использовать операторы AND и OR для объединения условий фильтрации.

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: Постраничный вывод данных используется для улучшения производительности и удобства пользователей при работе с большим объемом данных.

* 1. Что такое «пагинация»?

Ответ: Пагинация - это процесс разделения данных на страницы для упрощения навигации и отображения.

* 1. Для чего используется метод Take?

Ответ: Метод Take используется для выбора определенного количества элементов из источника данных.

* 1. Для чего используется метод Skip?

Ответ: Метод Skip используется для пропуска определенного числа элементов из источника данных.

* 1. Для чего используется метод TakeWhile?

Ответ: Метод TakeWhile используется для выбора элементов из источника по условию, пока это условие выполняется.

* 1. Для чего используется метод SkipWhile?

Ответ: Метод SkipWhile используется для пропуска элементов из источника по условию, пока это условие выполняется.

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

Ответ: Набор данных должен быть упорядоченным или иметь определенный порядок элементов, чтобы использование методов Take и Skip было корректным.

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: LINQ поддерживает различные агрегатные функции, такие как Count, Sum, Min, Max, Average и т. д.

* 1. Что возвращает метод Distinct?

Ответ: Метод Distinct возвращает уникальные элементы из последовательности.

* 1. Для чего используется метод GroupBy?

Ответ: Метод GroupBy используется для группировки элементов по определенному ключу.

* 1. Для чего используется метод Join?

Ответ: Метод Join используется для объединения двух коллекций на основе определенного ключа.

* 1. Для чего используется метод GroupJoin?

Ответ: Метод GroupJoin также используется для объединения двух коллекций, но с возможностью группировки результатов.

* 1. В чем отличие результатов, полученных при вызове LINQ-методов Concat, Union, Except, Intersect?
  2. Ответ: Результаты методов Concat, Union, Except и Intersect отличаются следующим образом:
* Concat объединяет две последовательности без удаления дубликатов.
* Union объединяет две последовательности и удаляет дубликаты.
* Except возвращает элементы первой последовательности, которых нет во второй.
* Intersect возвращает общие элементы из двух последовательностей.

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: Графические примитивы, доступные в приложениях WPF, включают в себя такие элементы как Ellipse (эллипс), Rectangle (прямоугольник), Line (линия), Polyline (полилиния), Polygon (полигон), Path (траектория) и другие.

* 1. Для чего используется Path?

Ответ: Path в WPF используется для создания комплексных геометрических фигур по заданным контурам. Он позволяет описывать любую форму, комбинируя линии, дуги, кривые и другие геометрические элементы.

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

Ответ: В приложениях WPF доступны различные виды трансформаций объектов, такие как TranslateTransform (перенос), ScaleTransform (масштабирование), RotateTransform (поворот), SkewTransform (наклон) и другие, позволяющие изменять расположение, размер и ориентацию объектов.

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

Ответ: Для указания заливки и контура графических объектов в WPF можно использовать свойства Fill (для заливки) и Stroke (для контура) в сочетании с SolidColorBrush для указания цвета. Например, для заливки объекта используется свойство Fill, а для контура - Stroke, где можно указать цвет, толщину и стиль контура.

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 приложениях WPF можно использовать объекты классов DoubleAnimation или SizeAnimation, а также привязку к свойству Width и Height элементов интерфейса.

* 1. Как задать анимацию цвета в приложениях WPF?

Ответ: Анимацию цвета в приложениях WPF можно задать с помощью объектов классов ColorAnimation или SolidColorBrushAnimation, а также привязки к свойству Color элементов интерфейса.

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

Ответ: Покадровая анимация в WPF обновляет состояние объекта на каждом кадре анимации, что может быть менее плавным и требовать больше вычислительных ресурсов, в то время как плавная анимация использует сглаживание и интерполяцию между кадрами для более плавного и непрерывного эффекта.

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

Ответ: Свойства, позволяющие управлять анимацией в WPF:

* Duration: определяет продолжительность анимации.
* BeginTime: задает время начала анимации.
* AutoReverse: определяет, должна ли анимация воспроизводиться в обратном порядке после завершения.
* FillBehavior: определяет, как анимация влияет на объект после завершения.
* RepeatBehavior: определяет, сколько раз анимация должна воспроизводиться или когда она должна прекратиться.

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

**Лабораторная работа No42**

**Разработка мультимедиаприложения**

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

Ответ: Мультимедиа — это технология, которая объединяет текст, графику, звук, анимацию и видео для создания интерактивного контента.

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

Ответ: Для отображения изображений в приложениях WPF используется элемент Image. Также можно использовать DrawingImage для векторной графики и BitmapImage для растровых изображений.

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

Ответ: Для воспроизведения аудио в WPF можно использовать элемент MediaElement или класс SoundPlayer.

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

Ответ: Для воспроизведения видео в WPF также используется элемент MediaElement, который поддерживает как аудио, так и видео.

1. **Вывод**
   1. Процесс создания мультимедиа-приложений на WPF.

**Лабораторная работа No43**

**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:

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += Timer\_Tick;

timer.Start();

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

Ответ: При подписке на событие у Timer.Tick нажать Tab, или же создать метод в ручную

* 1. Как изменить интервал таймера в приложении WPF?

Ответ: timer.Interval = TimeSpan.FromMilliseconds(500);

1. **Вывод:**
   1. Процесс разработки игровых приложений WPF, использующих графику, обработчики событий и таймеры изучен.

**Лабораторная работа No44**

**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: СУБД — это комплекс программных средств, предназначенных для создания, модификации, управления и организации доступа к базам данных. СУБД обеспечивает безопасность данных, поддерживает целостность, обеспечивает многопользовательский доступ и позволяет выполнять различные операции с данными.

* 1. Что такое «база данных»?

Ответ: База данных — это организованная коллекция структурированных данных, хранящихся в электронном виде в компьютерной системе. Базы данных обычно управляются СУБД и позволяют легко доступать, управлять, изменять, обновлять и анализировать данные.

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

Ответ: Реляционные СУБД используют реляционную модель данных, основанную на представлении данных в виде таблиц. Примеры реляционных СУБД:

* Oracle Database
* Microsoft SQL Server
* MySQL
* PostgreSQL
* SQLite

1. **Вывод:**
   1. Процесс создания таблиц и связей между ними в реляционной СУБД изучен.

**Лабораторная работа No45**

**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 в ADO.NET, который предоставляет подключение к базе данных MS SQL Server. Он используется для установления соединения с базой данных, выполнения команд и управления транзакциями.

* 1. Как считать из БД одно значение?

Ответ: Чтобы считать одно значение из БД, можно использовать метод ExecuteScalar объекта SqlCommand.

SqlConnection connection = new SqlConnection(connectionString)

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

command.ExecuteScalar();

* 1. Каково назначение элемента SqlDataReader?

Ответ: SqlDataReader — это класс, который предоставляет высокопроизводительный поток для чтения данных из базы данных. Он используется для последовательного чтения данных в прямом направлении.

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

Ответ:

SqlDataAdapter adapter = new SqlDataAdapter (query, connectionString);

DataTable table = new();

adapter.Fill(table)

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

Ответ: System.Data.SqlClient

1. **Вывод:**
   1. Мы научились создавать приложения для чтения данных из БД

**Лабораторная работа No46**

**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 используется для заполнения DataSet данными из базы данных и обновления базы данных изменениями, сделанными в DataSet. Это позволяет работать с данными в базе данных без необходимости постоянного подключения к ней.

* 1. Для чего применяется компонент SqlCommandBuilder?

Ответ: SqlCommandBuilder автоматически генерирует команды SQL для синхронизации изменений, сделанных в DataSet с базой данных. Это упрощает процесс обновления базы данных, так как не требует ручного создания команд INSERT, UPDATE и DELETE.

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

Ответ:

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

command.ExecuteNonQuery();

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

Ответ:

SqlDataAdapter adapter = new SqlDataAdapter();

adapter.UpdateCommand = new SqlCommand(query, connection);

adapter.Update(data);

* 1. Как связать SqlCommandBuilder и SqlDataAdapter?

Ответ:

SqlDataAdapter adapter = new SqlDataAdapter(query, connectionString);

SqlCommandBuilder commandBuilder = new (adapter);

1. **Вывод:**
   1. Мы научиться создавать приложения для записи данных в БД.

**Лабораторная работа No47**

**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 SQL-командах используются для передачи данных в запрос, что позволяет избежать SQL-инъекций и упрощает повторное использование запросов с различными значениями. Они также обеспечивают типизацию данных и повышают производительность приложения.

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

Ответ:

SqlCommand command = new SqlCommand("SELECT \* FROM table WHERE column = @ Value ", connection);

SqlParameter parameter = new SqlParameter("@Value", SqlDbType. Int);

parameter.Value = 10

command.Parameters.Add(parameter)

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

Ответ:

SqlCommand command = new SqlCommand("SELECT \* FROM table WHERE column = @parameter", connection);

command.Parameters.AddWithValue("@parameter", 10);

* 1. Какие свойства можно указать у параметра?

Ответ:

* ParameterName — имя параметра в команде.
* SqlDbType — тип данных параметра.
* Size — размер данных для параметров типа string или binary.
* Value — значение параметра.
* Direction — направление параметра (входной, выходной, возвращаемое значение).

1. **Вывод**
   1. Мы научились выполнять запросы к БД из клиентского приложения,
   2. Мы научились передавать параметры в запросы.

**Лабораторная работа No48**

**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: Хранимые процедуры — это набор SQL-команд, сохраненных в базе данных. Они применяются для выполнения повторяющихся операций, улучшения производительности за счет предварительной компиляции, обеспечения безопасности данных и упрощения сложных операций.

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

Ответ:

connection.Open();

SqlCommand command = new SqlCommand("PROCEDURE", connection);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

* 1. Как задать входные параметры хранимой процедуры?

Ответ:

connection.Open();

SqlCommand command = new SqlCommand("PROCEDURE", connection);

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@ParameterName", SqlDbType.Int)).Value = 10;

command.ExecuteNonQuery();

1. **Вывод**
   1. Мы научились создавать и использовать хранимые процедуры в MS SQL Server;
   2. Навык создания запросов на выборку и модификацию данных в MS SQL Server закреплен.

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: Класс Directory представляет статические методы для работы с каталогами, а класс DirectoryInfo представляет экземпляр каталога, что позволяет работать с определенным каталогом.

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

Ответ: Класс File представляет статические методы для работы с файлами, а класс FileInfo представляет экземпляр файла, что позволяет работать с определенным файлом.

* 1. Как получить список файлов и папок определенного каталога?

Ответ: Для получения списка файлов и папок определенного каталога можно использовать методы класса Directory, например, Directory.GetFiles() и Directory.GetDirectories().

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

Ответ: Свойства класса FileInfo, которые позволяют получить информацию о файле, включают FileInfo.Name (имя файла), FileInfo.Length (размер файла в байтах), FileInfo.CreationTime (время создания файла) и другие.

1. **Вывод**
   1. В ходе проделанной практической работы были получены умения 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 требуется подключить пространство имен Microsoft.Office.Interop.Excel.

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

Ответ: Для создания объекта типа "приложение Excel" в программе на C# можно использовать следующий код:

using Excel = Microsoft.Office.Interop.Excel;

Excel.Application excelApp = new Excel.Application();

* 1. Что такое Workbooks?

Ответ: Workbooks — это коллекция рабочих книг в программе при работе с Excel.

* 1. Что такое Worksheets?

Ответ: Worksheets — это коллекция листов в программе при работе с Excel.

* 1. Что такое Range?

Ответ: Range — это диапазон ячеек в программе при работе с Excel.

* 1. Что такое Cells?

Ответ: Cells — это коллекция ячеек в программе при работе с Excel.

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

Ответ: Для доступа к значению ячейки можно использовать свойство Value. Например:

Excel.Range cell = (Excel.Range)worksheet.Cells[1, 1];

string value = cell.Value.ToString();

1. **Вывод**
   1. В ходе проделанной практической работы были получены умения 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 требуется подключить пространство имен Microsoft.Office.Interop.Word.

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

Ответ: Для создания объекта типа "приложение Word" в программе на C# необходимо использовать следующий код:

using Word = Microsoft.Office.Interop. Word;

Word.Application wordApp = new Word.Application();

* 1. Что такое Documents?

Ответ: Documents — это коллекция документов в приложении Word, к которым можно обращаться и с которыми можно взаимодействовать.

* 1. Что такое Range?

Ответ: Range — это диапазон символов в документе Word, который можно выделить и работать с ним (изменять форматирование, текст и т.д.).

* 1. Что такое Selection?

Ответ: Selection — это выделенная часть текста в документе Word, с которой можно производить различные операции (вырезать, копировать, форматировать и т.д.).

* 1. Что такое Paragraphes?

Ответ: Paragraphes - это коллекция параграфов (абзацев) в документе Word, к которым можно обращаться для изменения текста и его форматирования.

* 1. Что такое Tables?

Ответ: Tables - это коллекция таблиц в документе Word, которые можно создавать, редактировать и форматировать с помощью программного кода.

1. **Вывод**
   1. В ходе проделанной практической работы были получены умения 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: Для добавления настроек в приложение на C# можно воспользоваться классом ConfigurationManager из пространства имен System.Configuration. Для этого нужно добавить раздел appSettings в файле конфигурации AppConfig, где можно указать необходимые параметры и их значения.

* 1. Как программно считать значение параметра из настроек?

Ответ: Для программного считывания значения параметра из настроек можно воспользоваться методом ConfigurationManager.AppSettings.Get("Имя параметра"), где "Имя параметра" - это ключ параметра в разделе appSettings.

* 1. Как программно изменить значение параметра в настройках?

Ответ: Для программного изменения значения параметра в настройках можно воспользоваться методом Configuration.AppSettings.Set("Имя параметра", "Значение параметра"), чтобы установить новое значение для указанного параметра.

* 1. Как выполнить сохранение значений параметров в настройках?

Ответ: Для сохранения значений параметров в настройках можно воспользоваться методом ConfigurationManager.RefreshSection("appSettings"), чтобы обновить значения параметров в разделе appSettings и сохранить их изменения.

1. **Вывод**
   1. В ходе проделанной практической работы были приобретены умения 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 внешнего вида элемента управления в WPF.

* 1. Где может быть описан ControlTemplate?

Ответ: ControlTemplate может быть описан в ресурсах приложения, в стилях или прямо в разметке элемента управления.

* 1. Для чего применяется UserControl?

Ответ: UserControl применяется для создания пользовательских элементов управления, объединяющих другие элементы и логику в единое целое.

* 1. Каков алгоритм создания пользовательского элемента управления?

Ответ:

1. Добавить в проект пользовательский элемент управления.
2. Разместитть необходимые элементы управления, настроить внешний вид
3. Пересобрать проект, чтобы пользовательский элемент управления стал доступен в панели элементов
4. Для передачи события из пользовательского элемента управления в основную программу нужно создать обработчик:
5. public event RoutedEventHandler GoToFirst
   1. Как программно создать обработчик события?

Ответ: Для создания обработчика события в C# программно нужно сначала объявить делегат, который будет представлять событие, затем создать само событие и назначить ему методы, которые будут обрабатывать это событие.

public delegate void MyEventHandler(object sender, EventArgs e);

public event MyEventHandler MyEvent;

public void HandleMyEvent(object sender, EventArgs e)

{

(обработка события)

}

Добавление обработчика к событию:

MyEvent += HandleMyEvent;

* 1. Для чего используется свойство зависимости?

Ответ: Свойство зависимости используется для обеспечения возможности управления значением свойства другими свойствами и уведомлении об изменениях свойства.

1. **Вывод**
   1. В ходе проделанной практической работы были приобретены умения 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 - это процесс установления связи между элементом пользовательского интерфейса (например, текстовым полем) и определенными данными (например, переменной или свойством объекта), таким образом, что изменения в одном автоматически отражаются в другом.

* 1. Каков шаблон настройки привязки данных?

Ответ: Шаблон настройки привязки данных включает в себя указание источника данных, целевого элемента интерфейса, типа привязки (односторонняя или двусторонняя), а также другие параметры, такие как формат отображения данных.

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

Ответ: Для создания конвертера значений необходимо реализовать интерфейс IValueConverter, который содержит методы Convert и ConvertBack для преобразования данных в обоих направлениях (из целевого элемента в данные и из данных в целевой элемент).

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

Ответ: Для валидации данных при привязке необходимо реализовать интерфейс IDataErrorInfo, который содержит свойства для обозначения ошибок валидации и методы для их обработки.

1. **Вывод**
   1. В ходе проделанной практической работы были приобретены умения выполнять привязку данных в приложении WPF.