**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**по МДК.01.01**

Студент: Давудова Д.И.

Группа: ИСПП-31

Преподаватель: Маломан Ю.С.

Архангельск 2025

**Лабораторная работа №1**

**Оценка сложности алгоритмов сортировки**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Массив — это структура однотипных элементов, занимающих непрерывную область памяти.

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

2.3 имя\_массива[индекс]

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

2.5 Сортировка – это процесс перестановки элементов данного множества в определенном порядке.

2.6 Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке

2.7 Сортировка методом простого выбора, сортировка методом «пузырька», сортировка вставками (или методом простого включения)

**3 Вывод**

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

**Лабораторная работа №2**

**Оценка сложности алгоритмов поиска**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке

2.2 Линейный поиск, двоичный поиск, алгоритм поиска прыжками

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

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

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

**3 Вывод**

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

**Лабораторная работа №3**

**Оценка сложности рекурсивных алгоритмов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Рекурсия — это подход, при котором функция вызывает саму себя (прямо или косвенно) для решения задачи.

2.2 Переполнение стека, Высокое потребление памяти, Медленная работа

2.3 Функция называется рекурсивной, если Имеет условие выхода и вызывает саму себя

Пример:

int Factorial(int n)

{

if (n == 0)

return 1;

return n \* Factorial(n - 1);

}

2.4 Глубина рекурсии — это количество вложенных вызовов функции до достижения базового случая.

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

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

**3 Вывод**

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

**Лабораторная работа №4**

**Оценка сложности эвристических алгоритмов**

**1 Цель работы**

1.1 Научиться реализовывать, оценивать и применять эвристические алгоритмы.

**2 Контрольные вопросы**

2.1 Эвристика — это метод решения задач, который ориентирован на практическое нахождение приближённого решения, когда точное решение слишком сложно или невозможно получить за разумное время.

* 1. Эвристические алгоритмы применяются в случаях, когда: важна

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

2.3 Жадные алгоритмы, генетические алгоритмы, метод имитации отжига.

**3 Вывод**

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

**Лабораторная работа №5**

**Работа с классами**

**1 Цель работы**

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

1.2 Изучить реализацию механизма инкапсуляции на языке C#.

**2 Контрольные вопросы**

2.1 [модификаторы] class ИмяКласса { }

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

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

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

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

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

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

2.8 Статические члены класса и члены экземпляра в C# отличаются способом вызова, областью хранения и доступом к данным.

**3 Вывод**

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

**Лабораторная работа №6**

**Перегрузка методов**

**1 Цель работы**

1.1 Изучить процесс выполнения перегрузки методов на языке C#;

1.2 Изучить различные способы передачи параметров в метод.

**2 Контрольные вопросы**

2.1 Перегрузка методов — это возможность определять несколько методов с одним и тем же именем в одном классе, но с разными параметрами.

* 1. Перегрузка методов относится к статическому полиморфизму.

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

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

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

**3 Вывод**

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

**Лабораторная работа №7**

**Определение операций в классе**

**1Цель работы**

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

**2 Контрольные вопросы**

2.1 Ключевое слово: operator

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

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

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

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

**3 Вывод**

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

**Лабораторная работа №8**

**Создание наследованных классов**

**1 Цель работы**

1.1 Изучить процесс разработки дочерних классов на языке C#;

1.2 Изучить реализацию механизма наследования на языке C#.

**2 Контрольные вопросы**

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

2.2 Дочерний класс может наследоваться только от одного родительского класса

* 1. Ключевое слово base используется для вызова конструктора

родителя

2.4 Переопределение метода — это замена реализации метода родителя в дочернем классе

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

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

**3 Вывод**

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

**Лабораторная работа №9**

**Работа с объектами через интерфейсы**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

2.2 Главное отличие в том, что интерфейс содержит только объявления методов и свойств без их реализации, тогда как абстрактный класс может включать как абстрактные методы, так и обычные методы с готовой реализацией. Класс может реализовывать множество интерфейсов, но наследоваться только от одного абстрактного класса. В интерфейсах нельзя объявлять поля и конструкторы, в то время как абстрактные классы поддерживают обычные поля и конструкторы.

2.3 Класс в C# может реализовывать сколько угодно интерфейсов.

2.4 [модификаторы] interface ИмяИнтерфейса

{

тип Метод(параметры); // Метод

тип Свойство { get; set; } // Свойство

событие Делегат Событие; // Событие

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

}

**3 Вывод**

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

**Лабораторная работа №10**

**Использование стандартных интерфейсов**

**1 Цель работы**

1.1 Изучить процесс реализации стандартных интерфейсов на языке C#

**2 Контрольные вопросы**

2.1 Интерфейс IComparable позволяет объектам определять естественный порядок сортировки

2.2 Интерфейс IEquatable<T> предоставляет метод для проверки равенства объектов

2.3 Ключевое различие между обобщенными и необобщенными интерфейсами заключается в типизации и производительности. Необобщенные интерфейсы работают с типом object, что может приводить к упаковке типов-значений и снижению производительности. Они также требуют приведения типов, что увеличивает риск ошибок времени выполнения. Обобщенные интерфейсы используют строгую типизацию через параметр типа T, что устраняет необходимость приведения типов и предотвращает упаковку для типов-значений, повышая производительность и безопасность кода.

**3 Вывод**

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

**Лабораторная работа №11**

**Коллекции. Параметризованные классы**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Коллекция в C# — это структура данных, предназначенная для хранения набора элементов (объектов или значений) и управления ими.

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

2.3 List<T> — это динамический массив, который автоматически расширяется при добавлении элементов.

2.4 К элементам List<T> обращаются по индексу

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

2.6 К элементам словаря обращаются по ключу

**3 Вывод**

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

**Лабораторная работа №12**

**Работа с типом данных структура**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

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

2.3 Структуры являются типами значений и хранятся в стеке или внутри родительского объекта, что делает их более быстрыми для небольших данных. Они копируются целиком при передаче или присваивании, не поддерживают наследование и не могут быть null. Классы же относятся к ссылочным типам - размещаются в куче, при операциях копируется только ссылка на объект, поддерживают полное наследование и могут принимать значение null.

2.4 Перечисление — это тип, который определяет набор именованных констант.

2.5 Перечисления применяются для: улучшения читаемости кода, ограничения допустимых значений

2.6 [модификаторы] enum ИмяПеречисления [: базовыйТип]

{

ИмяКонстанты1 [= значение],

ИмяКонстанты2 [= значение],

}

**3 Вывод**

В ходе лабораторной работы изучили процесс создания и применения структур на языке C#.

**Лабораторная работа №13**

**Обработка и форматирование строк**

**1 Цель работы**

1.1 Изучить процесс обработки строк на языке C#;

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

**2 Контрольные вопросы**

2.1 string в C# — это ссылочный тип

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

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

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

2.5 Все методы класса String возвращают новую строку, не изменяя исходную.

**3 Вывод**

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

**Лабораторная работа №14**

**Использование регулярных выражений**

**1 Цель работы**

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

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

**2 Контрольные вопросы**

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

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

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

2.4 1 Компиляция шаблона.

2 Сканирование текста.

3 Возврат результата

* 1. 1 Поиск совпадений

1. Замена

2.6 Escape-символы позволяют: экранировать спецсимволы, задавать спецпоследовательности

2.7 Классы символов группируют символы для упрощения шаблонов

2.8 Квантификаторы задают количество повторений

**3 Вывод**

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

**Лабораторная работа №15**

**Разработка делегатов**

**1 Цель работы**

1.1 Научиться разрабатывать и применять делегаты на C#

**2 Контрольные вопросы**

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

2.2 Делегат объявляется с ключевым словом delegate, сигнатурой метода и именем

2.3 Прямой вызов, явный вызов и асинхронный вызов

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

**3 Вывод**

В ходе лабораторной работы научились разрабатывать и применять делегаты на C#

**Лабораторная работа №16**

**Разработка событий**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

* 1. Событие объявляется с ключевым словом event и делегатом

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

2.4 Базовый класс для данных события — EventArgs

2.5 EventHandler, EventHandler<TEventArgs>

**3 Вывод**

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

**Лабораторная работа №17**

**Операции со списками**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

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

* 1. Var позволяет сократить код при работе с сложными типами

2.4 Метод Select преобразует элементы коллекции по заданному правилу.

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

2.6 Эти методы применяются после OrderBy для дополнительной сортировки по другим критериям

**3 Вывод**

В ходе лабораторной работы изучили процесс обработки списков стандартными методами на языке C#.

**Лабораторная работа №18**

**Проектирование диаграммы классов**

**1 Цель работы**

1.1 Изучить процесс описания типов данных с применением диаграммы классов.

1. **Контрольные вопросы**

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

2.2 Из имени, атрибутов и методов

2.3 На структурные, зависимости и анонимные связи

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

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

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

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

**3 Вывод**

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

**Лабораторная работа №19**

**Использование шаблонов проектирования**

**1 Цель работы**

1.1 Научиться применять паттерны проектирования в разработке программ.

**2 Контрольные вопросы**

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

2.2 Singleton, Factory Method, Abstract Factory, Builder

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

2.4 Adapter, Decorator, Facade, Composite

2.5 Поведенческие паттерны решают задачи взаимодействия между объектами и распределения ответственности

2.6 Observer, Strategy, Command, Iterator

**3 Вывод**

В ходе лабораторной работы научились применять паттерны проектирования в разработке программ.

**Лабораторная работа №20**

**Проектирование API**

**1 Цель работы**

1.1 Научиться проектировать API.

**2 Контрольные вопросы**

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

2.2 GET — запрос данных с сервера без изменения состояния ресурса.

2.3 POST — отправка данных на сервер для создания нового ресурса.

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

2.5 DELETE — удаление ресурса по указанному URI.

**3 Вывод**

В ходе лабораторной работы научились проектировать API.

**Лабораторная работа №21**

**Оптимизация кода**

**1 Цель работы**

1.1 Изучить методы оптимизации программного кода

**2 Контрольные вопросы**

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

2.2 Ускорение работы программы, снижение нагрузки на систему, поддержка масштабируемости

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

**3 Вывод**

В ходе лабораторной работы изучили методы оптимизации программного кода

**Лабораторная работа №22**

**Асинхронное программирование**

**1 Цель работы**

1.1 Научиться реализовывать и запускать асинхронные операции на C#.

1.2 Научиться выполнять вычисления, используя асинхронные операции.

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

**2 Контрольные вопросы**

2.1 async, await, Task / Task<T>

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

2.3 Используйте await для ожидания завершения асинхронной операции

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

2.5 Исключения в асинхронных методах перехватываются через try-catch

**3 Вывод**

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

**Лабораторная работа №23**

**Документирование кода**

**1 Цель работы**

1.1 Изучить процесс документирования программного кода

**2 Контрольные вопросы**

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

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

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

2.4 **<**summary> содержит краткое описание функциональности

2.5 <param> описывает параметры метода

**3 Вывод**

В ходе лабораторной работы изучили процесс документирования программного кода.

**Лабораторная работа №24**

**Рефакторинг кода**

**1 Цель работы**

1.1 Изучить и применить техники рефакторинга программного кода.

**2 Контрольные вопросы**

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

2.2 Рефакторинг методов, классов, работа с данными, упрощение вызовов

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

**3 Вывод**

В ходе лабораторной работы изучили и применили техники рефакторинга программного кода

**Лабораторная работа №25**

**Работа с системой контроля версий**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Репозиторий — это хранилище кода и связанных с ним файлов

2.2 README.md — документация проекта.

2.3 .gitignore — исключает ненужные файлы.

2.4 Командная строка, GUI-клиенты, интеграция в IDE

2.5 Локально — на вашем компьютере или удаленно — на хостингах

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

**3 Вывод**

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

**Лабораторная работа №26**

**Разработка интерфейса пользователя: компоновка элементов**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

2.2 Базовый класс для всех контейнеров — Panel

2.3 HorizontalAlignment или VerticalAlignment

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

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

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

2.6 px, Auto, \*, относительные единицы

**3 Вывод**

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

**Лабораторная работа №27**

**Организация интерфейса пользователя**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

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

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

2.4 Использовать свойство CanGoBack

2.5 Вызвать метод GoBack()

2.6 GroupBox, TabControl, Expander

**3 Вывод**

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

**Лабораторная работа №28**

**Разработка интерфейса пользователя: настройка стилей**

**1 Цель работы**

1.1 Изучить процесс настройки интерфейса с использованием стилей в

приложениях WPF.

**2 Контрольные вопросы**

2.1 Для чего используются стили в приложениях WPF?

Стили упрощают единообразное оформление элементов (шрифты, цвета).

2.2 Какова общая форма локального определения стиля элемента управления?

<Style TargetType="Button">

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

</Style>

2.3 Какова общая форма определения стиля приложения?

Определяется в App.xaml и применяется ко всем окнам.

2.4 Как указать явное использование стилей?

Указать x:Key и применить через StaticResource.

2.5 Как указать наследование стиля?

Использовать BasedOn="{StaticResource BaseStyle}".

2.6 Как добавить новую тему в приложение?

Создать новый ResourceDictionary и подключить в App.xaml.

2.7 Как выполнить переключение между темами?

Можно изменить ресурс словаря в коде или XAML.

**3 Вывод**

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

**Лабораторная работа №29**

**Разработка интерфейса пользователя: применение триггеров**

**1 Цель работы**

1.1 Изучить процесс применения триггеров в приложениях WPF,

1.2 Закрепить навык применения стилей в приложениях на WPF.

**2 Контрольные вопросы**

2.1 Что позволяют делать триггеры в приложениях WPF?

Позволяют автоматически изменять свойства элементов при определенных условиях.

2.2 Какие виды триггеров можно разработать в приложениях WPF?

Свойств, данных и событий.

2.3 Для чего используется и когда срабатывает триггер свойств?

Срабатывает при изменении свойства, например IsMouseOver. Используется для визуальных эффектов.

2.4. Для чего используется и когда срабатывает триггер данных?

Когда меняются данные, например, текст в поле. Можно отобразить ошибку или подсказку.

2.5. Для чего используется и когда срабатывает триггер событий?

Когда происходит событие, например клик. Обычно используется в шаблонах.

**3 Вывод**

В ходе лабораторной работы изучили процесс применения триггеров в приложениях WPF и закрепили навык применения стилей в приложениях на WPF.

**Лабораторная работа №30**

**Изучение особенностей элементов выбора в приложениях WPF**

**1 Цель работы**

1.1 Изучить свойства и процесс обработки событий элементов выбора в

приложениях WPF.

**2 Контрольные вопросы**

2.1. Что такое CheckBox и для чего он используется?

Флажок для выбора одного/нескольких вариантов .

2.2. Что такое RadioButton и для чего он используется?

Переключатель для выбора одного варианта из группы .

2.3. Как проверить, что флажок или переключатель выбран?

Через свойство IsChecked. Если true — выбран.

2.4. Какое событие срабатывает при выборе флажка или переключателя?

Событие Checked.

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

Событие Unchecked.

2.6. Какие значения могут принимать флажки и переключатели?

True, False, у CheckBox ещё может быть null (если не задано точно).

**3 Вывод**

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

**Лабораторная работа №31**

**Разработка приложения с использованием текстовых компонентов**

**1 Цель работы**

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

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

**2 Контрольные вопросы**

2.1. Как задать имя элементам управления в WPF?

x:Name="TextBoxLogin".

2.2. Как создать обработчик события в WPF?

В XAML: Click="Button\_Click", в коде: private void Button\_Click(...).

2.3. Для чего используется Slider в WPF?

Чтобы выбрать числовое значение в диапазоне.

2.4. Для чего используется TextBox в WPF?

Для ввода текста пользователем.

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

TextBlock просто показывает текст. Label можно привязать к элементу

через "for".

2.6. Для чего используется Calendar в WPF?

Для отображения дат.

2.7. Для чего используется DatePicker в WPF?

Для выбора конкретной даты из выпадающего календаря.

2.8. Для чего используется PasswordBox в WPF?

Для скрытого ввода пароля.

**3 Вывод**

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

**Лабораторная работа №32**

**Разработка приложения с использованием элементов отображения списков**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Что такое ComboBox и для чего он используется?

выпадающий список для выбора одного варианта.

2.2. Что такое ListBox и для чего он используется?

Cписок с возможностью выбора одного/нескольких элементов.

2.3. Какое событие срабатывает при выборе элемента в селекторе?

SelectionChanged.

2.4. В каком свойстве хранятся элементы селекторов?

В свойстве Items.

2.5. Какого типа элементы могут быть в селекторе?

Любые — строки, картинки.

2.6. Какое свойство позволяет привязать селектор к набору данных?

ItemsSource.

2.7. Для чего используется свойство DisplayMemberPath в селекторе?

Чтобы указать, какое поле объекта показывать в списке.

**3 Вывод**

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

**Лабораторная работа №33**

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

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Что такое DataGrid и для чего он используется?

Таблица для отображения и редактирования данных.

2.2. Какие типы столбцов поддерживаются в DataGrid?

Текстовые, чекбоксы, гиперссылки, выпадающие списки и кнопки.

2.3. Как добавить кнопку в строки DataGrid?

Использовать DataGridTemplateColumn.

2.4. Как указать источник данных для DataGrid?

Присвоить список в ItemsSource.

2.5. Как указать источник данных для выпадающего списка DataGrid?

ItemsSource в столбце.

1. **Вывод**

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

**Лабораторная работа №34**

**Разработка приложения с меню и панелью инструментов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Что такое Menu и для чего он используется?

Главное меню приложения.

2.2. Что такое ContextMenu и для чего он используется?

Контекстное меню, вызываемое правой кнопкой мыши.

2.3. Что такое ToolBar и для чего он используется?

Панель с кнопками для быстрого доступа к функциям.

2.4. Что такое StatusBar и для чего он используется?

Строка состояния.

1. **Вывод**

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

**Лабораторная работа №35**

**Создание проекта с использованием компонентов стандартных диалогов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Что такое диалоговое окно?

Это вспомогательное окно для взаимодействия с пользователем.

2.2. Для чего используется OpenFileDialog?

Чтобы выбрать файл с диска и открыть его в приложении.

2.3. Для чего используется SaveFileDialog?

Чтобы сохранить данные в файл, указав имя и место.

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

Это окно с сообщением.

2.5. Для чего используется PrintDialog?

Чтобы отправить на печать содержимое окна или элемента.

**3 Вывод**

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

**Лабораторная работа №36**

**Разработка приложения с несколькими формами**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

Нужно создать публичное свойство в окне.

2.2. Какие значения может принимать переменная DialogResult?

True, False.

2.3. Как открыть окно в диалоговом режиме?

окно.ShowDialog().

2.4. Как открыть окно в недиалоговом режиме?

окно.Show().

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

Диалоговое окно блокирует остальные, пока его не закроют. Недиалоговое нет.

**3 Вывод**

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

**Лабораторная работа №37**

**Реализация фильтрации данных**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Для чего используется метод Where?

Фильтрует данные по условию.

2.2. Какие логические операторы могут применяться при составлении условий?

&& (и), || (или), ! (не), а также обычные сравнения: ==, >, < и т.п.

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

С помощью метода StartsWith.

2.4. Как проверить, что строка содержит определенный текст?

Через метод Contains.

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

Применять Where() несколько раз или объединять условия через && и ||.

**3 Вывод**

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

**Лабораторная работа №38**

**Реализация постраничного вывода информации**

**1 Цель работы**

1.1 Научиться применять LINQ-запросы для постраничного вывода данных.

**2 Контрольные вопросы**

2.1. Почему может потребоваться выводить данные постранично?

Чтобы не грузить сразу огромный список.

2.2. Что такое «пагинация»?

Это деление списка данных на страницы, чтобы их можно было пролистывать.

2.3. Для чего используется метод Take?

Он берёт заданное количество элементов из коллекции.

2.4. Для чего используется метод Skip?

Он пропускает заданное количество элементов — например, чтобы начать со второй страницы.

2.5. Для чего используется метод TakeWhile?

Он берёт элементы до тех пор, пока выполняется условие.

2.6. Для чего используется метод SkipWhile?

Он пропускает элементы, пока выполняется условие, и потом возвращает остальные.

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

Он должен быть упорядочен, иначе страницы будут отображать данные в случайном порядке.

**3 Вывод**

В ходе лабораторной работы научились применять LINQ-запросы для постраничного вывода данных.

**Лабораторная работа №39**

**Реализация группировки и соединения данных**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Какие агрегатные функции поддерживаются в LINQ?

Sum, Min, Max, Average, Count.

2.2. Что возвращает метод Distinct?

Возвращает уникальные элементы.

2.3. Для чего используется метод GroupBy?

Группирует данные по ключу.

2.4. Для чего используется метод Join?

Чтобы соединить данные из двух коллекций по общему полю.

2.5. Для чего используется метод GroupJoin?

Группирует элементы одной коллекции по другой.

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

Concat — объединяет коллекции.

Union — объединяет без дубликатов.

Except — возвращает разницу.

Intersect — возвращает пересечение.

**3 Вывод**

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

**Лабораторная работа №40**

**Разработка приложения для работы с графикой**

**1 Цель работы**

1.1 Изучить процесс рисования и трансформации объектов в приложениях WPF.

**2 Контрольные вопросы**

2.1. Какие графические примитивы доступны в приложениях WPF?

Линии, прямоугольники, эллипсы, полигоны, пути.

2.2. Для чего используется Path?

Чтобы рисовать сложные фигуры — дуги, кривые, комбинированные формы.

2.3. Какие виды трансформаций объектов доступны в WPF?

Масштабирование, поворот, сдвиг, наклон.

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

Через свойства Fill (заливка) и Stroke (контур), можно использовать цвета, градиенты или картинки.

**3 Вывод**

В ходе лабораторной работы изучили процесс рисования и трансформации объектов в приложениях WPF.

**Лабораторная работа №41**

**Разработка приложения с анимацией**

**1 Цель работы**

1.1 Изучить процесс анимации объектов в приложениях WPF.

**2 Контрольные вопросы**

2.1. Как задать анимацию размеров в WPF?

Через DoubleAnimation, указав начальное и конечное значение, и длительность.

2.2. Как задать анимацию цвета в WPF?

Использовать ColorAnimation для свойства Background.

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

Покадровая задаёт чёткие ключевые кадры, а плавная изменяет значение по времени.

2.4. Какие свойства управляют анимацией и зачем они нужны?

Duration (длительность),

RepeatBehavior (повтор),

AutoReverse (возврат к началу).

**3 Вывод**

В ходе лабораторной работы изучили процесс анимации объектов в приложениях WPF.

**Лабораторная работа №42**

**Разработка мультимедиа-приложения**

**1 Цель работы**

1.1 Изучить процесс создания мультимедиа-приложений на WPF.

**2 Контрольные вопросы**

2.1. Что такое «мультимедиа»?

Это работа с видео, аудио, изображениями.

2.2. Какие элементы отображают изображения в WPF?

Image, InkCanvas.

2.3. Какие элементы воспроизводят аудио в WPF?

MediaElement и SoundPlayer.

2.4. Какие элементы воспроизводят видео в WPF?

MediaElement.

**3 Вывод**

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

**Лабораторная работа №43**

**Разработка игрового приложения**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Как подключить таймер к приложению на WPF?  
Используется DispatcherTimer.

2.2. Как создать обработчик события для таймера в приложении WPF?  
Создают метод и привязывают его к событию Tick.

2.3. Как изменить интервал таймера в приложении WPF?  
Установить новое значение свойства Interval.

**3 Вывод**

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

**Лабораторная работа №44**

**Создание БД**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. Что такое «система управления базами данных»?

Это программа, которая помогает создавать, хранить и управлять базами данных.

2.2. Что такое «база данных»?

Это структурированный набор данных.

2.3. Какие СУБД являются реляционными (примеры)?

SQLite, MSSQL, MySQL.

**3 Вывод**

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

**Лабораторная работа №45**

**Создание приложения с БД для чтения данных**

**1 Цель работы**

1.1 Научиться создавать приложения для чтения данных из БД.

**2 Контрольные вопросы**

2.1. Каково назначение элемента SqlConnection?

Он устанавливает соединение между приложением и базой данных.

2.2. Как считать из БД одно значение?

Выполнить SQL-команду с ExecuteScalar()

2.3. Каково назначение элемента SqlDataReader?

Он построчно читает данные из результата запроса — удобно для списков.

2.4. Какие пространства имён подключать для MS SQL?

System.Data, System.Data.SqlClient.

**3 Вывод**

В ходе лабораторной работы научились создавать приложения для чтения данных из БД.

**Лабораторная работа №46**

**Создание приложения с БД для записи данных**

**1 Цель работы**

1.1 Научиться создавать приложения для чтения данных из БД.

**2 Контрольные вопросы**

2.1. Как изменить данные в БД, используя SqlCommand?

Написать команду UPDATE и вызвать ExecuteNonQuery().

2.2. Как передать параметры в SqlCommand?

Добавить параметры через command.Parameters.AddWithValue().

2.3. Для чего используются параметры в SqlCommand?

Чтобы защититься от SQL-инъекций и сделать запросы гибкими.

2.4. Как изменить данные в БД, используя SqlDataAdapter?

Внести изменения в DataTable и вызвать Update() у адаптера.

2.5. Для чего применяется SqlCommandBuilder?

Он сам создаёт SQL-команды (INSERT, UPDATE, DELETE) для SqlDataAdapter.

2.6. Как связать SqlCommandBuilder и SqlDataAdapter?

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

**3 Вывод**

В ходе лабораторной работы научились создавать приложения для чтения данных из БД.

**Лабораторная работа №47**

**Создание запросов к БД**

**1 Цель работы**

1.1 Научиться выполнять запросы к БД из клиентского приложения,

1.2 Научиться передавать параметры в запросы.

**2 Контрольные вопросы**

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

Они отделяют работу с данными от остального кода.

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

Они управляют бизнес-логикой и используют репозитории для доступа к данным.

2.3. Для чего используются ORM при работе с БД?

Упрощает работу с БД, маппинг объектов на таблицы.

2.4. Какие свойства можно указать у параметра?

Тип, имя, значение, направление.

**3 Вывод**

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

**Лабораторная работа №48**

**Создание сервисов**

**1 Цель работы**

1.1 Научиться выполнять запросы к БД из клиентского приложения,

1.2 Научиться передавать параметры в запросы.

**2 Контрольные вопросы**

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

Чтобы выполнять CRUD-операции изолированно.

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

Добавляют логику поверх репозитория

2.3. Для чего используются ORM при работе с БД?

Ускоряет разработку, минимизируя SQL-код.

2.4. Какие свойства можно указать у параметра?

Имя, тип, значение, размер.

**3 Вывод**

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

**Лабораторная работа №1**

**Разработка приложений для обработки файлов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1. В чём отличие между классами Directory и DirectoryInfo?

Directory — это статический класс, DirectoryInfo — создаётся как объект. Первый подходит для быстрого доступа, второй — для работы с конкретной папкой.

2.2. В чём отличие между классами File и FileInfo?

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

2.3. Как получить список файлов и папок определённого каталога?

Через методы Directory.GetFiles() и Directory.GetDirectories().

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

Name, Length, CreationTime, Extension, DirectoryName.

**3 Вывод**

В ходе практической работы научились применять классы для работы с файлами в приложениях на C#.

**Лабораторная работа №2**

**Работа с табличными файлами**

**1 Цель работы**

1.1 Научиться выполнять создание и редактирование табличных документовна C#.

**2 Контрольные вопросы**

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

Microsoft.Office.Interop.Excel.

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

var excel = new Excel.Application();

2.3. Что такое Workbooks?

Коллекция всех открытых Excel-файлов (рабочих книг) в приложении.

2.4. Что такое Worksheets?

Коллекция всех листов в одной книге.

2.5. Что такое Range?

Это диапазон ячеек, к которым можно обратиться, чтобы получить или изменить данные.

2.6. Что такое Cells?

Это доступ к отдельным ячейкам по строкам и столбцам.

2.7. Как получить доступ к значению ячейки и диапазона?

worksheet.Cells[1,1].Value или range.Value.

**3 Вывод**

В ходе практической работы научились выполнять создание и редактирование табличных документовна C#.

**Лабораторная работа №3**

**Работа с текстовыми файлами**

**1 Цель работы**

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

**2 Контрольные вопросы**

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

Microsoft.Office.Interop.Word.

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

var word = new Word.Application();

2.3. Что такое Documents?

Коллекция всех документов, открытых в Word.

2.4. Что такое Range?

Это участок текста — можно с ним работать: форматировать, заменять, вставлять.

2.5. Что такое Selection?

Это то, что сейчас выделено пользователем или программой.

2.6. Что такое Paragraphs?

Это список всех абзацев в документе.

2.7. Что такое Tables?

Таблицы в документе.

**3 Вывод**

В ходе практической работы научились выполнять создание и редактирование текстовых документов на C#.

**Лабораторная работа №4**

**Привязка данных**

**1 Цель работы**

1.1 Научиться выполнять привязку данных в приложении WPF.

**2 Контрольные вопросы**

2.1. Что такое «привязка данных»?

Это когда элемент управления сам получает и отображает данные из объекта или коллекции.

2.2. Каков шаблон настройки привязки данных?

Указывается путь к свойству и источник.

2.3. Какой интерфейс надо реализовать для создания конвертера значений?

IValueConverter.

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

IDataErrorInfo или INotifyDataErrorInfo.

**3 Вывод**

**Лабораторная работа №5**

**Сохранение настроек приложения**

**1 Цель работы**

1.1 Научиться сохранять настройки в клиентском приложении на C#.

**2 Контрольные вопросы**

2.1 Как добавить настройки в приложение на C#?

Нужно открыть файл конфигурации приложения (например, App.config или Settings.settings) и добавить необходимые параметры.

2.2 Как программно считать значение параметра из настроек?

var mySetting = Properties.Settings.Default.MySetting;

2.3 Как программно изменить значение параметра в настройках?

Properties.Settings.Default.MySetting = newValue;

2.4 Как выполнить сохранение значений параметров в настройках?

Properties.Settings.Default.Save();

**3 Вывод**

В ходе практической работы научились сохранять настройки в клиентском приложении на C#.

**Практическая работа №6**

**Создание пользовательских элементов управления**

**1 Цель работы**

1.1 Научиться создавать пользовательские элементы управления в приложении WPF.

**2 Контрольные вопросы**

2.1. Для чего применяется ControlTemplate?

Чтобы изменить внешний вид стандартного элемента.

2.2. Где может быть описан ControlTemplate?

В ресурсах окна, страницы, приложения или в словаре ресурсов.

2.3. Для чего применяется UserControl?

Чтобы создать свой элемент управления из уже готовых компонентов.

2.4. Каков алгоритм создания пользовательского элемента управления?

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

2.5. Как программно создать обработчик события?

Пример: кнопка.Click += НазваниеМетода;

2.6. Для чего используется свойство зависимости?

Для привязки данных.

**3 Вывод**

В ходе практической работы научились создавать пользовательские элементы управления в приложении WPF.