**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**по МДК.01.01**

Студент: Поршнев А.Н., Мошников А.В.

Группа: ИСПП-21

Преподаватель: Маломан Ю.С.

Архангельск 2024

**Лабораторная работа №1**

**Оценка сложности алгоритмов сортировки**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Что такое "массив"

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

2.2 Как описывается одномерный массив?

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

2.3 Как обратиться к некоторому элементу одномерного массива?

К элементу массива можно обратиться по индексу:

имяМассива[индекс элемента];

2.4 Как можно задать одномерный массив?

Массив можно задать программно, с клавиатуры или случайными числами.

2.5 Что такое "сортировка"?

Сортировка – процесс перестановки элементов данного множества в определённом порядке.

2.6 Что такое алгоритм сортировки?

Алгоритм сортировки – это алгоритм для упорядочивания элементов в списке.

**3 Вывод**

Мы 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 массива в C#.

**2 Контрольные вопросы**

2.1 Что такое "алгоритм поиска"?

Алгоритм поиска – это алгоритм нахождения индекса заданного значения в массиве.

2.2 Какие виды поиска элементов массивов существуют?

Существует три вид поиска элементов в массиве: линейный, двоичный и алгоритм поиска прыжками.

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

Метод полного перебора массива. Временная сложность – O(N).

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

На каждом шаге множество делится на две части и в работе остаётся та часть множества,где находится искомый объект. Временная сложность - O(log(N)).

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

Аналогичен алгоритму двоичного поиска, но движение только вперёд. Временная сложность – O(sqrt(N))

**3 Вывод**

Мы научились реализовывать и оценивать сложность алгоритмов поиска элементов массива в C#.

**Лабораторная работа №3**

**Оценка сложности рекурсивных алгоритмов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Что такое "рекурсия"?

Рекурсия – функция, которая вызывает саму себя.

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

В рекурсивных методах обязательно должен быть предусмотрен выход из метода, иначе произойдёт переполнение стека.

2.3 Какое определение функции может быть названо рекурсивным? Привести примеры.

Рекурсивным определением функции принято называть такое определение, при котором значения функции для данных аргументов определяются значениями функции для более простых аргументов. Например: числа Фиббоначи, вычисление факториала числа.

2.4 Что такое "глубина рекурсии"?

Глубина рекурсии – количество последовательных вызовов без возврата значения.

2.5 Что такое "рекурсивный спуск"?

Рекурсивный спуск – движение вычислительного процесса в направлении последовательных рекурсивных вызовов.

2.6 Что такое "рекурсивный подъём"?

Рекурсивный подъём – движение вычислительного процесса в направление возврата из ранее вызванных копий рекурсивной процедуры.

**3 Вывод**

Мы 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 и оценивать сложность эвристических алгоритмов на C#.

**2 Контрольные вопросы**

2.1 Что такое "многомерный массив"?

Многомерный массив – массив, который отличается двумя или более измерениями.

2.2 Как описывается двумерный массив?

тип[,] имяМассива = new тип[количество строк, количество столбцов];

2.3 Как обратиться к некоторому элементу двумерного массива?

К элементу массива можно обратиться по индексу:

имяМассива[индекс элемента 1, индекс элемента 2]

2.4 Как узнать количество строк двумерного массива?

имяМассива.GetLength(0);

2.5 Как узнать количество столбцов двумерного массива?

имяМассива.GetLength(1);

2.6 Как вывести двумерный массив на консоль в виде таблицы?

Вывести двумерный массив можно с помощью двух циклов for:

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

{

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

{

Console.Write("${arr[i, j]}\t");

}

Console.WriteLine();

}

**3 Вывод**

Мы научились реализовывать и оценивать сложность эвристических алгоритмов на C#.

**Лабораторная работа №5**

**Работа с классами**

**1 Цель работы**

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

1.2 Изучить реализацию механизма инкапсуляции на языке C#.

**2 Контрольные вопросы**

2.1 Какова общая форма объявления класса?

модификатор class ИмяКласса

{

// тело класса

}

2.2 На какие виды делятся данные класса?

Поля, свойства и методы.

2.3 На какие виды делятся функции класса?

Свойства, методы, конструкторы и деструкторы.

2.4 Для чего применяются конструкторы классов?

Для создания объектов класса.

2.5 Что такое "цепочка конструкторов"?

Цепочка конструкторов – механизм, позволяющий вызывать один конструктор из другого конструктора того же класса при создании объекта.

2.6 Для чего применяются свойства классов?

Для задания собственных значений созданному в программе экземпляру объекта.

2.7 Когда используются автоматически реализуемые свойства классов?

Автоматически реализуемые свойства имеет смысл использовать тогда, когда нет необходимости накладывать какие-либо ограничения на возможные значения неявного поля свойства.

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

Обращение к статическим полям класса происходит по имени класса и не загромождает пространство глобальных имён.

**3 Вывод**

Мы изучили процесс разработки, применения классов и реализацию механизма инкапсуляции на языке C#.

**Лабораторная работа №6**

**Перегрузка методов**

**1 Цель работы**

1.1 Изучить процесс выполнения перегрузки методов на языке C#;

1.2 Изучить различные способы передачи параметров в метод.

**2 Контрольные вопросы**

2.1 Что такое "перегрузка методов"?

Перегрузка методов – использование одного и того же имени метода с разными параметрами.

2.2 К какому виду полиморфизма относится перегрузка методов?

Перегрузка методов относится к статическому полиморфизму.

2.3 Какие типы функций класса можно перегружать?

Перегружать можно функции, отличающиеся по количеству параметров.

2.4 Какие требования предъявляются к сигнатуре перегружаемых функций класса?

Передача в параметрах только типа данных.

2.5 Какие типы функций класса нельзя перегружать?

Нельзя перегружать функции, отличающиеся по типу возвращаемого значения.

**3 Вывод**

Мы изучили процесс выполнения перегрузки методов на языке C# и различные способы передачи параметров в метод.

**Лабораторная работа №7**

**Определение операций в классе**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Какое ключевое слово определяет операторный метод?

operator – ключевое слово, определяющее операторный метод.

2.2 Какие унарные операторы можно перегружать в C#?

Перегружать можно такие операторы, как +, -, !, ++, --, true и false.

2.3 Какие бинарные операторы можно перегружать в C#?

Перегружать можно такие операторы, как &, |, +, -, \*, /, %.

2.4 Какие операторы сравнения можно перегружать в C#?

>= и <=, == и !=, < и >.

2.5 Какие операторы требуется перегружать в C# попарно?

== и !=, < и >, а также >= и <= и true и false.

**3 Вывод**

Мы 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 Что такое "наследование"?

Наследование – механизм создания нового класса на основе уже существующего.

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

Дочерний класс может иметь несколько родительских.

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

Ключевое слово base.

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

Переопределение метода – метод, который позволяет вызывать функции из другого класса в дочернем. Создание метода в дочернем классе с той же сигнатурой, что и у метода в базовом классе.

2.5 Что такое «абстрактный класс»?

Абстрактный класс – класс, который содержит один или несколько абстрактных методов. Абстрактный класс не может использоваться для создания объектов.

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

Virtual — делает метод виртуальным.

Override — позволяет переопределить виртуальный метод в наследующем классе.

Abstract — используется для объявления абстрактных методов. Абстрактный метод автоматически становится виртуальным.

New — скрывает виртуальный метод в базовом классе.

**3 Вывод**

Мы изучили процесс разработки дочерних классов на языке C# и реализацию механизма наследования на языке C#.

**Лабораторная работа №9**

**Работа с объектами через интерфейсы**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Что такое "интерфейс"?

Интерфейс – ссылочный тип, который может определять набор методов и свойств без реализации.

2.2 Чем отличается интерфейс от абстрактного класса?

В интерфейсе нельзя прописывать поля, интерфейс может наследоваться только от интерфейса, члены интерфейса по умолчанию открытые и не могут иметь модификаторов доступа.

2.3 Есть ли у класса ограничения по количеству реализуемых интерфейсов?

У класса есть ограничение по количеству реализуемых интерфейсов,но оно невероятно велико (224  или более 16000000).

2.4 Какова общая форма объявления интерфейсов и их элементов?

interface IИмяИнтерфейса

{

ТипВозвращаемогоЗначения ИмяМетода(параметры);

ТипВозвращаемогоЗначения ИмяСвойства;

}

**3 Вывод**

Мы 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 Для чего используется интерфейс IComparable?

IComparable определяет обобщенный метод сравнения, который реализуется типом значения или классом для упорядочения или сортировки экземпляров.

2.2 Для чего используется интерфейс IEquatable?

IEquatable определяет обобщенный метод, который реализуется типом значения или классом для создания метода с целью определения экземпляров.

2.3 В чём отличие между обобщёнными и необобщёнными интерфейсами?

В реализации обобщённых интерфейсов используется обобщённый тип T:

interface IИмяИнтерфейса<T>, где T – любой тип данных

Обобщённый интерфейс может содержать методы с универсальными параметрами и универсальные свойства.

В реализации необобщённых интерфейсов используется конкретный тип используемых/возвращаемых данных

**3 Вывод**

Мы 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 Что такое "коллекция"?

Коллекция – динамическая структура данных, возможно, разного типа.

2.2 Какие классы описаны пространством имён System.Collections.Generic?

CollectionExtensions, Comparer<T>, Dictionary<TKey, TValue>, Queue<T>, List<T> и т.д.

2.3 Что такое List<T>?

Список типизированных объектов.

2.4 Как можно обратиться к элементу списка?

К элементу списка можно обратиться по индексу.

2.5 Что такое Dictionary<TKey, TValue>?

Коллекция, представляющая пару ключ – значение.

2.6 Как можно обратиться к элементу словаря?

К элементу словаря можно обратиться по ключу: имяСловаря[ключ].

**3 Вывод**

Мы 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 В какой области памяти хранятся типы значения?

Типы значения хранятся в стеке.

2.2 Что такое «структура»?

Структура – это тип значения, который может содержать поля, методы, константы, конструкторы, свойства, индексаторы, операторы и другие типы структур.

2.3 Чем отличается структура от класса?

Основное отличие структуры (struct) от класса (class) заключается в том, что структура – это тип значений, а класс – это ссылочный тип.

2.4 Что такое «перечисление»?

Перечисление – список однотипных значений.

2.5 Для чего используются перечисления?

Перечисление чаще всего используется в качестве хранилища состояния.

2.6 Какова общая форма объявления перечисления в C#?

enum ИмяПеречисления

{

переменная1, (= 1 – указание значения, по умолчанию – 0)

переменная2,

...

переменнаяN

}

**3 Вывод**

Мы 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 К какому типу переменных относятся переменные типа string?

Переменные типа string относятся к строковому типу данных.

2.2 Какие операции допустимы над строковыми данными?

Сравнение строк (Compare), поиск в строке(IndexOf, LastIndexOf, StartsWith, EndsWith), разделение строк (Split), обрезка строки (Trim, TrimStart, TrimEnd, SubString), вставка (Insert), удаление строк (Remove), замена (Replace), смена регистра (ToUpper, ToLower).

2.3 В чем отличие переменной типа string от массива символов?

Тип string объявляется без указания количества символов, а char[] объявляется как обычный массив и каждый элемент имеет свой индекс.

2.4 Что такое «интерполяция строк»?

Интерполяция строк – это механизм объединения двух или более строк вместе.

2.5 Изменяют ли методы класса String исходную строку?

Только методы обрезки, разделения, замены, вставки и удаления изменяют исходную строку.

**3 Вывод**

Мы изучили процесс обработки строк на языке C# 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 Что такое «регулярное выражение»?

Регулярное выражение - это шаблон, который используется для синтаксического анализа и проверки соответствия данного входного текста заданному шаблону или нет.

2.2 Для чего используются регулярные выражения?

Регулярные выражения используются для упрощения работы с текстом.

2.3 Для чего используется класс Regex?

Класс Regex используется для реализации регулярных выражений, предлагает методы и свойства для анализа большого текста.

2.4 Каков алгоритм поиска подстроки при помощи регулярного выражения?

Regex regex = new(@"\w\*искомаяСтрока\w\*"), где \w\* – любая последовательность алфавитно-цифровых символов.

2.5 Каков алгоритм замены подстроки при помощи регулярного выражения?

Regex regex = new Regex(шаблон);

string result = regex.Replace(текст, заменяемыйСимвол/Слово);

2.6 Для чего в регулярных выражениях применяются escape-символы?

Следующий символ является специальным или реализация паттернов.

2.7 Для чего в регулярных выражениях применяются классы символов?

Класс символов определяет набор символов, любой из которых может присутствовать во входной строке для успешного сопоставления.

2.8 Для чего в регулярных выражениях применяются квантификаторы?

Квантификаторы определяют количество экземпляров символа, группы или класса символов, которое должно присутствовать во входных данных, чтобы было зафиксировано совпадение.

**3 Вывод**

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

**Лабораторная работа №15**

**Разработка делегатов**

**1 Цель работы**

1.1 Научиться разрабатывать и применять делегаты на C#.

**2 Контрольные вопросы**

2.1 Что такое «делегат» в C#?

Делегат в C# — это тип, который представляет ссылки на методы со списком параметров и типом возвращаемого значения.

2.2 Как объявить делегат на C#?

модификатор delegate типВозвращаемогоЗначения ИмяДелегата();

2.3 Какие способы вызова метода через делегат существуют?

Callback имяПеременной = ИмяМетодаДелегата

или

модификатор типВозвращаемогоЗначения ИмяМетода(параметры, Callback callback)

{

callback();

}

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

Action представляет некоторое действие, которое ничего не возвращает, то есть в качестве возвращаемого типа имеет тип void.

Predicate принимает один параметр и возвращает значение типа bool.

Func возвращает результат действия и может принимать параметры.

**3 Вывод**

Мы 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 Что такое «событие» в C#?

Событие – класс, передающий системе о возникновении каких-либо ситуаций.

2.2 Как объявить событие на C#?

delegate void ИмяДелегата(параметры);

event ИмяДелегата ИмяСобытия;

2.3 Как создать обработчик события?

В качестве обработчика событий чаще всего выступают методы юез возврата значения:

void ИмяМетода(параметры) { }

ИмяСобытия += ИмяМетода; // добавление обработчика

2.4 Какой класс является родительским для всех классов данных события?

Класс EventArgs.

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

Delegate и MulticastDelegate.

**3 Вывод**

Мы научились создавать, вызывать и обрабатывать события на C#.

**Лабораторная работа №17**

**Операции со списками**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Что такое «LINQ»?

LINQ (Language Integrated Query) – язык запросов к источнику данных.

2.2 Что такое «анонимный тип»?

Анонимный тип – тип данных, позволяющий создать объект с некоторым набором свойств без определения класса.

2.3 Для чего используется var?

Для объявления и инициализации переменной без определения типа данных.

2.4 Для чего используется метод Select?

Select определяет проекцию выбранных значений.

2.5 Для чего используются методы OrderBy, OrderByDescending?

Сортировка по первостепенному по значимости определённому критерию по возрастанию/убыванию соответственно.

2.6 Для чего используются методы ThenBy, ThenByDescending?

Сортировка по второму по значимости определённому критерию по возрастанию/убыванию соответственно.

**3 Вывод**

Мы 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.

**2 Контрольные вопросы**

2.1 Для чего используется диаграмма классов?

Диаграмма классов используется для изображения логической и физической структуры системы и показывает ее классы.

2.2 Из каких элементов состоит описание типа в диаграмме классов?

модификатор (- – private, + – public) имяПеременной : типПеременной

модификатор ИмяМетода()

2.3 На какие группы делятся взаимосвязи в диаграмме классов?

Наследование, реализация, композиция, агрегация, ассоциация.

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

Реализация – указание связи между интерфейсами и классами.

Наследование – описание отношений между родительскими и дочерними классами.

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

Агрегация –отношение между целым и частью класса, объекты-члены являются частью общего объекта, но объект-член может существовать независимо от общего объекта.

Композиция – отношение между целым и частью класса, а общее и часть имеют согласованное время жизни.

2.6 Что показывает взаимосвязь «ассоциация»?

Ассоциация указывает, что свойство класса содержит ссылку на экземпляр (или экземпляры) другого класса.

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

1..1 – связь один к одному, 0..\* - ноль или больше, 1..\* - связь один ко многим, 0..1 – ноль или один, m..n – не менее m, не более n.

**3 Вывод**

Мы 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 Для чего используются порождающие паттерны?

Для создания классов и объектов.

2.2 Какие паттерны относятся к порождающим?

Абстрактная фабрика, строитель, фабричный метод, прототип, одиночка.

2.3 Для чего используются структурные паттерны?

Для объединения классов в более сложные структуры.

2.4 Какие паттерны относятся к структурным?

Адаптер, мост, компоновщик, декоратор, фасад, приспособленец, заместитель.

2.5 Для чего используются поведенческие паттерны?

Определяют алгоритмы и взаимодействие между классами и объектами.

2.6 Какие паттерны относятся к поведенческим?

Цепочка обязанностей, команда, интерпретатор, итератор, посредник, хранитель, наблюдатель, состояние, стратегия, шаблонный метод, посетитель.

**3 Вывод**

Мынаучились применять паттерны проектирования в разработке программ.

**Лабораторная работа №20**

**Проектирование API**

**1 Цель работы**

1.1 Научиться проектировать API.

**2 Контрольные вопросы**

2.1 Что такое REST?

REST – архитектура с применением методов или типов запросов HTTP.

2.2 Для чего используется метод GET?

GET – получение данных.

2.3 Для чего используется метод POST?

POST – добавление данных.

2.4 Для чего используется метод PUT?

PUT – изменение данных.

2.5 Для чего используется метод DELETE?

DELETE – удаление данных.

**3 Вывод**

Мы научились проектировать API.

**Лабораторная работа №21**

**Оптимизация кода**

**1 Цель работы**

1.1 Изучить методы оптимизации программного кода.

**2 Контрольные вопросы**

2.1 Что такое «оптимизация программного кода»?

Оптимизация программного кода – модификация системы, чтобы заставить некоторые её аспекты работать более эффективно, использовать меньше ресурсов или быть более надёжными.

2.2 Какова цель оптимизации программного кода?

Заставить некоторые аспекты системы работать более эффективно, использовать меньше ресурсов или быть более надёжными.

2.3 Какие методы оптимизации программного кода применяются?

Рефакторинг, использование более низкоуровневой системы, применение менее точного решения, использование более эффективной структуры данных и более эффективных алгоритмов работы программы.

**3 Вывод**

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

Ключевые слова в работе с асинхронными вызовами - async и await.

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

void – без возврата данных, Task – возврат объекта типа Task, Task<T> - метод может вернуть некоторое значение, ValueTask<T> - аналогично Task<T>, но использует больше памяти из-за работы с большим количеством данных.

2.3 Как вызвать метод в асинхронном режиме?

await ИмяМетодаAsync.

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

Прописать после имени метода Async.

2.5 Как обработать исключения, возникшие в асинхронных вызовах

Поместить выражение await в блок try.

**3 Вывод**

Мынаучились реализовывать и запускать асинхронные операции на C#, выполнять вычисления, используя асинхронные операции и 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 Что такое «XML-документация»?

XML (eXtensible Markup Language) – расширяемый язык разметки.

2.2 Как сгенерировать XML-комментарий?

Сгенерировать комментарий можно в классе, поставив перед ним «///».

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

Нужно установить зависимость от того проекта, в котором используется XML-документация.

2.4 Что пишется в разделе summary?

Пояснение к тому, что делает тот или иной метод.

2.5 Что пишется в разделе param?

Пояснение параметров

2.6 Что пишется в разделе returns?

Комментарий о возвращаемом значении.

**3 Вывод**

Мы изучили процесс документирования программного кода.

**Лабораторная работа №24**

**Рефакторинг кода**

**1 Цель работы**

1.1 Изучить техники рефакторинга программного кода.

**2 Контрольные вопросы**

2.1 Что такое «рефакторинг»?

Рефакторинг — это процесс изменения кода, призванный упростить его обслуживание, понимание и расширение, при этом не изменяя его поведение.

2.2 Какие группы техник рефакторинга существуют?

Составление методов, перемещение функций между объектами, организация данных, упрощение условных выражений, упрощение вызовов методов, решение задач обобщения.

2.3 Как выполнить рефакторинг в Visual Studio?

Комбинация «Ctrl + . » → выбрать нужную функцию рефакторинга.

**3 Вывод**

Мы изучили техники рефакторинга программного кода.

**Лабораторная работа №25**

**Работа с системой контроля версий**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Что такое «репозиторий»?

Репозиторий – удалённое хранилище данных, доступное по ссылке.

2.2 Что указывается в файле readme.md?

readme.md – руководство, которое предоставляет другим разработчикам подробное описание вашего проекта.

2.3 Что указывается в файле .gitignore?

Файлы и папки, которые не будут загружаться в репозиторий.

2.4 Какое программное обеспечение может применяться для управления git-репозиторием?

git bash, GitHub и другие.

2.5 Где может располагаться репозиторий?

Репозиторий может располагаться на GitHub и других подобных ему площадках удалённых репозиториев.

**3 Вывод**

Мы 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 Что такое «компоновка» в WPF?

Компоновка – процесс размещения элементов внутри контейнера.

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

Object – родительский класс.

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

Указать в свойствах VerticalAlignment и HorizontalAlignment параметр Center, Left или Right.

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

Для размещения элементов в Grid нужно создавать столбцы и строки, а у элементов прописывать Grid.Row и/или Grid.Column, а в Canvas у каждого элемента прописывается его положение относительно краёв (Canvas.Top, Canvas.Left, Canvas.Right и Canvas.Bottom, в параметрах пишется смещение. Например: Canvas.Left = “100”).

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

StackPanel: элементы располагаются сверху вниз не во всю ширину окна.

DockPanel: привязка элементов осуществляется к краям контейнера через свойство DockPanel.Dock.

WrapPanel: элементы располагаются слева направо, если уменьшать размер окна, то элементы будут смещаться друг под друга.

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

Размеры элементов могут задаваться числами, автоматически, в пикселях и \*(для заполнения оставшейся части формы).

**3 Вывод**

Мы 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 Для чего используется элемент управления Frame?

Для расположения страницы и её элементов внутри Frame.

2.2 Для чего используется элемент управления Page?

Для реализации постраничного перехода на форме.

2.3 Как перейти к определенной странице, используя фрейм?

Создать кнопку и событие для перехода на определённую страницу.

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

if (ИмяКадра.CanGoBack())

{ }

2.5 Как перейти к предыдущей странице, используя фрейм?

ИмяКадра.GoBack();

2.6 Какие элементы позволяют сгруппировать содержимое?

Для группировки элементов управления используются элементы-контейнеры.

**3 Вывод**

Мы 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 используются для упрощения написания кода и настройки группы элементов.

2.2 Какова общая форма локального определения стиля элемента управления?

<Window.Resources>

<Style TargetType "НазваниеЭлементаУправления">

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

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

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

…

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

</Style>

</Window.Resources>

2.3 Какова общая форма определения стиля приложения?

<Window.Resources>

<Style x:Key="ИмяСтиля">

<Setter Property = "Control.Свойство" Value = "…" />

<Setter Property = "Control.Свойство" Value = "…" />

<Setter Property = "Control.Свойство" Value = "…" />

…

<Setter Property = "Control.Свойство" Value = "…" />

</Style>

</Window.Resources>

2.4 Как указать явное использование стилей?

Привязать стиль к элементу по имени элемента.

2.5 Как указать наследование стиля?

У элемента, наследующего стиль, указать в свойствах BasedOn = "{StaticResource НазваниеСтиля}"

2.6 Как добавить новую тему в приложение?

Создать словарь ресурсов и прописать всё, что должно быть в теме приложения. Затем у прописать строку, отвечающую за тему Style = "{DynamicResource НазваниеСтиля}"

2.7 Как выполнить переключение между темами?

Создать два словаря ресурсов с разными наборами значений. В файле инициализации формы прописать метод на смену и создать список со значениями тем.

В WPF создать элемент ComboBox с именем styleBox

 public MainWindow()

{

InitializeComponent();

List<string> styles = new List<string> { "light", "dark" };

styleBox.SelectionChanged += ThemeChange;

styleBox.ItemsSource = styles;

styleBox.SelectedItem = "dark";

}

private void ThemeChange(object sender, SelectionChangedEventArgs e)

{

string style = styleBox.SelectedItem as string;

var uri = new Uri(style + ".xaml", UriKind.Relative);

ResourceDictionary dictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(dictionary);

}

**3 Вывод**

Мы 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?

Тригерры позволяют декларативно задать некоторые действия, которые выполняются при изменении свойств стиля.

2.2 Какие виды триггеров можно разработать в приложениях WPF?

Триггер свойств, триггер данных, триггер событий и мультитриггеры.

2.3 Для чего используется и когда срабатывает триггер свойств?

Они следят за значением свойств и в случае их изменения с помощью объекта Setter устанавливают значение других свойств.

2.4 Для чего используется и когда срабатывает триггер данных?

Отслеживает изменение свойств, которые необязательно должны представлять свойства зависимостей.

2.5 Для чего используется и когда срабатывает триггер событий?

Реагирует на определенные события, как и обработчики событий.

**3 Вывод**

Мы изучили 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**

8.1 Что такое CheckBox и для чего он используется?

CheckBox – флажок, используется для выбора одного или более значений.

8.2 Что такое RadioButton и для чего он используется?

RadioButton – переключатель, используется для выбора одного значения (нужно назначить GroupName).

8.3 Как проверить, что флажок или переключатель выбран?

У RadioButton и у CheckBox – IsChecked.

8.4 Какое событие срабатывает при выборе флажка или переключателя?

Событие Checked

8.5 Какое событие срабатывает при снятии выбора флажка или переключателя?

Событие Unchecked.

8.6 Какие значения могут принимать флажки и переключатели?

У RadioButton –True, False, у CheckBox – {x:Null}, True, False.

**3 Вывод**

Мы изучили свойства и процесс обработки событий элементов выбора в приложениях WPF.

**Лабораторная работа №31**

**Разработка приложения с использованием текстовых компонент**

**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**

8.1 Как задать имя элементам управления в WPF?

x:Name/Name = "ИмяЭлементаУправления"

8.2 Как создать обработчик события в WPF?

У кнопки, например:

Click = "ИмяКнопки\_Click"

8.3 Для чего используется Slider в WPF?

Для создания ползунка и использования его для выбора значений на форме.

8.4 Для чего используется TextBox в WPF?

Для ввода текста на форме.

8.5 Для чего используется TextBlock в WPF?

Для отображения подсказок на форме.

8.6 Для чего используется Calendar в WPF?

Для просмотра текущей даты на форме.

8.7 Для чего используется DatePicker в WPF?

Для выбора даты на форме.

8.8 Для чего используется PasswordBox в WPF?

Для ввода текста, скрытого специальными символами (\* или подобными ей).

**3 Вывод**

Мы изучили различные типы полей ввода, применяющихся в приложениях WPF, 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?

ComboBox – выпадающий список для выбора одного значения.

2.2 Что такое ListBox и для чего он используется?

ListBox – обычный список для выбора одного значения.

2.3 Какое событие срабатывает при выборе элемента в селекторе?

SelectionIndexChanged срабатывает при выборе элемента в селекторе.

2.4 В каком свойстве хранятся элементы селекторов?

Элементы селекторов хранятся в Items.

2.5 Какого типа элементы могут быть в селекторе?

В селекторе могут быть данные любого стандартного типа.

2.6 Какое свойство позволяет привязать селектор к набору данных?

ItemsSource позволяет привязать селектор к набору данных.

2.7 Для чего используется свойство DisplayMemberPath в селекторе?

Для отображения только значения, имя которого указано в DisplayMemberPath.

**3 Вывод**

Мы 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 в приложениях WPF.

**2 Контрольные вопросы**

2.1 Что такое DataGrid и для чего он используется?

Элемент табличного представления данных, позволяющий редактировать содержимое.

2.2 Какие типы столбцов поддерживаются в DataGrid?

DataGridTextColumn – текст, DataGridHyperlinkColumn – гиперссылка, DataGridCheckBoxColumn – флажок, DataGridComboBoxColumn – выпадающий список, DataGridTemplateColumn – шаблон для отображения столбца.

2.3 Как добавить кнопку в строки DataGrid?

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="…"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

2.4 Как указать источник данных для DataGrid?

Прописать ItemsSource для DataGrid.

2.5 Как указать источник данных для выпадающего списка DataGrid?

Прописать ItemsSource для выпадающего списка.

**3 Вывод**

Мы изучили свойства и процесс настройки внешнего вида элемента DataGrid в приложениях WPF.

**Лабораторная работа №34**

**Разработка приложения с меню и панелью инструментов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Что такое Menu и для чего он используется?

Menu – меню с подпунктами настройки. Данный элемент служит для создания стандартных меню.

2.2 Что такое ContextMenu и для чего он используется?

ContextMenu – контекстное меню. Используется для создания контекстного меню по нажатию ПКМ.

2.3 Что такое ToolBar и для чего он используется?

ToolBar – панель инструментов. Применяется для обеспечения быстрого доступа к наиболее часто используемым операциям.

2.4 Что такое StatusBar и для чего он используется?

StatusBar – строка состояния. Используется для вывода информации о работе приложения.

**3 Вывод**

Мы 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?

Диалоговое окно – окно, предназначенное для вывода информации и/или получения ответа от пользователя

2.2 Для чего используется OpenFileDialog?

Для открытия диалогового окна открытия файла.

2.3 Для чего используется SaveFileDialog?

Для открытия диалогового окна сохранения файла.

2.4 Что такое MessageBox и какие настройки можно ему задать?

MessageBox – окно сообщения. Ему можно задать текст заголовка, текст внутри окна, картинку и кнопки.

2.5 Для чего используется PrintDialog?

Для открытия диалогового окна печати файла.

**3 Вывод**

Мы 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 пользовательских окон в приложениях WPF.

**2 Контрольные вопросы**

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

Обращаться к диалоговому окну через DialogResult.

2.2 Какие значения может принимать переменная DialogResult?

True и False.

2.3 Как открыть окно в диалоговом режиме?

Window.ShowDialog().

2.4 Как открыть окно в недиалоговом режиме?

Window.Show().

2.5 В чем отличие между диалоговым и недиалоговым режимами работы?

Диалоговые окна блокируют работу с недиалоговыми окнами до тех пор, пока они не будут закрыты. Недиалоговые окна не блокируют выполнение текущего потока, когда он активен, поэтому, если это дочернее окно, оно позволяет вам взаимодействовать с его родительским.

**3 Вывод**

Мы изучили процесс создания пользовательских окон в приложениях WPF.

**Лабораторная работа №37**

**Разработка приложения с несколькими формами**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Для чего используется метод Where?

Метод Where в LINQ используется для фильтрации последовательностей (коллекций) данных. Он позволяет выбрать только те элементы, которые удовлетворяют заданному условию.

2.2 Какие логические операторы могут применяться при составлении условий?

В LINQ-запросах, как и в C# в целом, используются стандартные логические операторы: && (И) объединяет два условия, оба из которых должны быть истинными; || (ИЛИ) объединяет два условия, хотя бы одно из которых должно быть истинным; ! (НЕ) инвертирует результат условия; == (Равно) проверяет равенство; != (Не равно) проверяет неравенство; <, >, <=, >= операторы сравнения (меньше, больше, меньше или равно, больше или равно соответственно).

2.3 Как выполнить регистронезависимый поиск?

Для регистронезависимого поиска в LINQ используются методы ToLower или ToUpper для приведения строк к одному регистру.

2.4 Как проверить, что строка начинается с определенного текста?

Метод StartsWith().

2.5 Как проверить, что строка содержит определенный текст?

Метод Contains().

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

В LINQ можно использовать комбинацию методов Where и логических операторов для составления динамических запросов, основанных на пользовательских критериях.

**3 Вывод**

Мы 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?

Оптимизация производительности: при работе с большими объемами данных загрузка всех данных сразу может быть ресурсоемкой и привести к медленной работе приложения. Постраничный вывод позволяет загружать только ограниченное количество данных на каждой странице, что ускоряет загрузку и снижает нагрузку на систему.

Удобство использования: вывод данных в виде страниц делает интерфейс приложения более удобным для пользователя. Пользователь может просматривать данные по частям, без необходимости прокручивать длинный список.

2.2 Что такое «пагинация»?

Пагинация – это способ разделения большого объема данных на отдельные страницы, чтобы пользователь мог просматривать их по частям.

2.3 Для чего используется метод Take?

Для получения заданного количества элементов из последовательности данных.

2.4 Для чего используется метод Skip?

Для пропуска заданного количества элементов из последовательности данных.

2.5 Для чего используется метод TakeWhile?

Для получения элементов из последовательности, пока не будет достигнут заданный предикат (условие).

2.6 Для чего используется метод SkipWhile?

Для пропуска элементов из последовательности, пока не будет достигнут заданный предикат (условие).

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

Методы Take и Skip работают с любыми последовательностями данных, которые реализуют интерфейс IEnumerable<T>, например: массивы, списки.

**3 Вывод**

Мы научились применять LINQ-запросы для постраничного вывода данных.

**Лабораторная работа №39**

**Разработка приложения с несколькими формами**

**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?

Count(): Подсчитывает количество элементов в последовательности.

Sum(): Суммирует числовые значения элементов последовательности.

Average(): Вычисляет среднее значение числовых элементов последовательности.

Min(): Возвращает минимальное значение из элементов последовательности.

Max(): Возвращает максимальное значение из элементов последовательности.

First(): Возвращает первый элемент последовательности.

FirstOrDefault(): Возвращает первый элемент последовательности или значение по умолчанию (null для ссылочных типов и 0 для числовых типов), если последовательность пуста.

Last(): Возвращает последний элемент последовательности.

LastOrDefault(): Возвращает последний элемент последовательности или значение по умолчанию (null для ссылочных типов и 0 для числовых типов), если последовательность пуста.

Single(): Возвращает единственный элемент последовательности.

SingleOrDefault(): Возвращает единственный элемент последовательности или значение по умолчанию (null для ссылочных типов и 0 для числовых типов), если последовательность пуста или содержит более одного элемента.

2.2 Что возвращает метод Distinct?

Метод возвращает новую последовательность, содержащую только уникальные элементы исходной последовательности. Дубликаты удаляются.

2.3 Для чего используется метод GroupBy?

Для группировки элементов последовательности по ключу. Метод возвращает новую последовательность, состоящую из групп, каждая из которых содержит элементы с одинаковым ключом.

2.4 Для чего используется метод Join?

Метод Join используется для объединения элементов из двух последовательностей на основе условия. Метод возвращает новую последовательность, содержащую элементы, соответствующие условию.

2.5 Для чего используется метод GroupJoin?

Для группового объединения элементов из двух последовательностей. Метод возвращает новую последовательность, где каждый элемент исходной последовательности (левой) сопоставлен с группой элементов из другой последовательности (правой) на основе заданного ключа.

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

Concat: объединяет две последовательности, сохраняя порядок элементов из исходных последовательностей.

Union: объединяет две последовательности, удаляя дубликаты.

Except: возвращает последовательность, содержащую элементы из первой последовательности, которые отсутствуют во второй.

Intersect: возвращает последовательность, содержащую элементы, которые присутствуют как в первой, так и во второй последовательности.

**3 Вывод**

Мы 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?

Элементарные фигуры: Ellipse (Эллипс), Rectangle (Прямоугольник), Line (Линия), Polygon (Многоугольник), Polyline (Ломаная линия)

Текстовые элементы: TextBlock (Текст), Label (Метка), TextBox (Поле ввода), RichTextBox (Текстовый редактор)

Геометрические объекты: Path (Путь) - для создания сложных фигур, Geometry (Геометрия) - базовый класс для геометрических фигур

Другие примитивы: Image (Изображение), Video (Видео), VisualBrush (Кисть для визуального содержимого), DrawingBrush (Кисть для рисования)

2.2 Для чего используется Path?

Для создания сложных геометрических фигур, которые не могут быть представлены простыми примитивами. Он позволяет задать путь рисования с помощью Data свойства, которое принимает объект типа Geometry (или его производные).

2.3 Какие виды трансформаций объектов доступны в приложениях WPF?

TranslateTransform: перемещение объекта по горизонтали и вертикали.

ScaleTransform: изменение размера объекта.

RotateTransform: поворот объекта.

SkewTransform: искажение объекта (перекос).

MatrixTransform: применяет матрицу преобразования к объекту (более продвинутый вариант).

2.4 Как указать заливку и контур графических объектов в WPF?

Заливка: В WPF используется свойство Fill для задания заливки графических объектов. В качестве значения Fill можно задать объект типа Brush.

Контур: для задания контура объекта используется свойство Stroke. В качестве значения Stroke можно задать объект типа Brush, чтобы указать цвет контура, и свойство StrokeThickness для установки толщины

**3 Вывод**

Мы 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 приложениях WPF?

Для анимации размеров в WPF можно использовать DoubleAnimation или SizeAnimation в сочетании с Storyboard.

2.2 Как задать анимацию цвета в приложениях WPF?

Для анимации цвета в WPF можно использовать ColorAnimation в сочетании с Storyboard.

2.3 Чем отличается покадровая анимация от плавной анимации в WPF?

Покадровая анимация: используется для создания анимации с помощью последовательности статических изображений (кадров). В WPF это можно реализовать с помощью BitmapImage и Image элемента. Каждый кадр отображается на определенный промежуток времени, создавая иллюзию движения.

Плавная анимация: используется для создания плавного изменения свойств объекта во времени. В WPF для этого используются анимации типа DoubleAnimation, ColorAnimation, ThicknessAnimation и другие. Они позволяют задать начальное и конечное значение свойства и время анимации.

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

Duration: задает длительность анимации.

From: задает начальное значение анимируемого свойства.

To: задает конечное значение анимируемого свойства.

AutoReverse: определяет, будет ли анимация воспроизведена в обратном порядке после завершения.

RepeatBehavior: задает количество повторов анимации.

SpeedRatio: устанавливает скорость анимации.

FillBehavior: определяет, как обрабатывается оставшееся время анимации после ее завершения.

EasingFunction: задает функцию изменения скорости анимации во времени (для более плавного движения).

**3 Вывод**

Мы 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»?

Мультимедиа – это информация, представленная в виде комбинации различных типов носителей, таких как текст, изображения, звук, видео, анимация и т.д. Мультимедиа делает информацию более интересной, привлекательной и доступной для восприятия.

2.2 Какие элементы позволяют отображать изображения в приложениях WPF?

Image: основной элемент для отображения изображений. Он принимает в качестве источника объект ImageSource, который может быть BitmapImage, BitmapSource, DrawingImage и др.

BitmapImage: класс, представляющий растровое изображение.

DrawingImage: класс, представляющий изображение, созданное с помощью Drawing объекта.

VisualBrush: кисть, использующая Visual (визуальный элемент) в качестве шаблона для заливки.

2.3 Какие элементы позволяют воспроизводить аудио в приложениях WPF?

MediaElement: универсальный элемент для воспроизведения медиа-файлов (аудио и видео). Он поддерживает различные форматы аудио и видео, а также предоставляет возможность управления воспроизведением.

SoundPlayer: класс, предназначенный для воспроизведения звуковых файлов (например, .wav).

2.4 Какие элементы позволяют воспроизводить видео в приложениях WPF?

В WPF для воспроизведения видео также используется элемент MediaElement. Он поддерживает различные форматы видео и предоставляет функции управления воспроизведением, такие как: Play, Pause, Stop, Volume, Position.

**3 Вывод**

Мы 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?

В WPF таймер создается с помощью класса DispatcherTimer.

2.2 Как создать обработчик события для таймера в приложении WPF?

Чтобы выполнить код при каждом срабатывании таймера, необходимо создать обработчик события Tick.

2.3 Как изменить интервал таймера в приложении WPF?

Интервал таймера можно изменить в любое время, задав новое значение для свойства Interval.

**3 Вывод**

Мы 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»?

Система управления базами данных (СУБД) — это программное обеспечение, которое обеспечивает создание, хранение, управление и доступ к данным в базе данных.

2.2 Что такое «база данных»?

База данных — это организованная коллекция данных, хранящаяся в электронном виде и предназначенная для эффективного поиска, обновления и управления информацией.

2.3 Какие СУБД являются реляционными (привести примеры ПО)?

Реляционные СУБД — это системы, которые хранят данные в таблицах, где каждая строка представляет запись, а каждый столбец — атрибут записи.Например:Oracle Database: одна из самых распространенных СУБД, используемая во многих корпоративных приложениях; Microsoft SQL Server: популярная СУБД от Microsoft, используемая для различных целей, от небольших приложений до масштабных корпоративных систем; MySQL: бесплатная и открытая СУБД, широко используемая в веб-разработке; PostgreSQL: бесплатная и открытая СУБД, отличающаяся своей надежностью и богатыми возможностями; SQLite: встраиваемая СУБД, используемая в приложениях, где не требуется отдельного сервера.

**3 Вывод**

Мы 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?

SqlConnection – это класс в .NET, который представляет собой соединение с базой данных SQL Server. Он используется для: установки соединения, выполнения команд, управления транзакциями, получения информации о соединении.

2.2 Как считать из БД одно значение?

string query = "SELECT TOP 1 Name FROM Customers";

command.ExecuteScalar();

2.3 Каково назначение элемента SqlDataReader?

SqlDataReader – это класс в .NET, который позволяет читать данные из базы данных SQL Server, полученные в результате выполнения SQL-запроса. Он используется для: построчного чтения данных, перемещения по результатам запроса, закрытия соединения.

2.4 Как получить данные из БД, используя SqlDataAdapter?

SqlDataAdapter используется для получения данных из базы данных SQL Server и заполнения набора данных (DataSet).

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

using System.Data;

using System.Data.SqlClient;

using Microsoft.Data.SqlClient;

**3 Вывод**

Мы 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 записи данных в БД.

**2 Контрольные вопросы**

2.1 Для чего применяется компонент SqlDataAdapter?

Заполнение набора данных: чтение данных из базы данных и их помещение в DataSet с помощью метода Fill().

Обновление базы данных: обновление данных в базе данных на основе изменений в DataSet с помощью метода Update().

Создание команд: создание SqlCommand объектов для выполнения операций с базой данных (SELECT, INSERT, UPDATE, DELETE).

Управление транзакциями: управление транзакциями для обновления данных в базе данных.

2.2 Для чего применяется компонент SqlCommandBuilder?

Автоматически генерирует SQL-команды (INSERT, UPDATE, DELETE) для обновления данных в базе данных на основе схемы DataSet. Он используется в сочетании с SqlDataAdapter для упрощения процесса обновления данных.

2.3 Как изменить данные в БД, используя SqlCommand?

SqlCommand command = new SqlCommand(query, connection);

int rowsAffected = command.ExecuteNonQuery();

Console.WriteLine($"Обновлено {rowsAffected} строк.");

2.4 Как изменить данные в БД, используя SqlDataAdapter?

using (SqlDataAdapter adapter = new SqlDataAdapter(selectQuery, connection))

{

SqlCommandBuilder builder = new SqlCommandBuilder(adapter);

DataSet dataSet = new DataSet();

adapter.Fill(dataSet);

dataSet.Tables[0].Rows[0]["…"] = "…";

adapter.Update(dataSet);

}

2.3 Как связать SqlCommandBuilder и SqlDataAdapter?

SqlCommandBuilder автоматически связывается с SqlDataAdapter при создании.

**3 Вывод**

Мы научились создавать приложения для записи данных в 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 параметры в командах?

Параметры в SQL-командах используются для: безопасной передачи данных в SQL-запросы, предотвращая возможность злоумышленников внедрять вредоносный код в запросы, улучшения производительности, упрощения кода.

2.2 Как добавить параметр в команду, используя Add?

command.Parameters.Add(“@Параметр”);

2.3 Как добавить параметр в команду, используя AddWithValue?

command.Parameters.AddWithValue(“@Параметр”, значение);

2.4 Какие свойства можно указать у параметра?

ParameterName: имя параметра (например, @Name или @CustomerID).

SqlDbType: тип данных параметра (например, SqlDbType.VarChar, SqlDbType.Int, SqlDbType.DateTime и т.д.).

Size: размер параметра (для текстовых типов данных).

Direction: гаправление параметра: Input (входной), Output (выходной), InputOutput (входной/выходной).

Value: значение параметра.

**3 Вывод**

Мы научились выполнять запросы к БД из клиентского приложения, а также научились передавать параметры в запросы.

**Лабораторная работа №48**

**Создание хранимых процедур**

**1 Цель работы**

1.1 Научиться создавать и использовать хранимые процедуры в MS SQL Server,

1.2 Закрепить навык создания запросов на выборку и модификацию данных в MS SQL Server.

**2 Контрольные вопросы**

2.1 Что такое хранимые процедуры и для чего они применяются?

Хранимые процедуры — это предкомпилированные блоки SQL-кода, которые хранятся на сервере базы данных и могут быть вызваны из приложения. Используются для упрощения написания SQL-запросов.

2.2 Как вызвать выполнение хранимой процедуры в клиентском приложении?

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command =

new SqlCommand("MyStoredProcedureName", connection))

{

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

}

}

2.3 Как задать входные параметры хранимой процедуры?

using (SqlCommand command =

new SqlCommand("MyStoredProcedureName", connection))

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.AddWithValue("@InputParameter1", "Value1");

command.Parameters.AddWithValue("@InputParameter2", 123);

}

**3 Вывод**

Мы научились создавать и использовать хранимые процедуры в MS SQL Server, а также закрепили навык создания запросов на выборку и модификацию данных в MS SQL Server.

**Практическая работа №1**

**Разработка приложений для обработки файлов**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 В чем отличие между классами Directory и DirectoryInfo?

Класс Directory предоставляет статические методы для работы с каталогами, а DirectoryInfo предоставляет информацию об определённом каталоге.

2.2 В чем отличие между классами File и FileInfo?

Класс File предоставляет статические методы для работы с файлами, а FileInfo предоставляет информацию об определённом файле.

2.3 Как получить список файлов и папок определенного каталога?

DirectoryInfo directory = new("путь");

FileInfo files = directory.GetFiles();

DirectoryInfo directories = directory.GetDirectories();

2.4 Какие свойства класса FileInfo позволяют получить информацию о файле?

Length, Extension, Name, FullName.

**3 Вывод**

Мы научились применять классы для работы с файлами в приложениях на C#.

**Практическая работа №2**

**Разработка приложения с несколькими формами**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Какое пространство имен требуется подключить для работы с Excel?

using Excel = Microsoft.Office.Interop.Excel;

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

Excel.Application excelApp = new();

2.3 Что такое Workbooks?

Свойство Application для создания/открытия рабочей книги таблицы Excel.

2.4 Что такое Worksheets?

Свойство рабочей книги для создания/переключения рабочего листа таблицы Excel.

2.5 Что такое Range?

Свойство рабочего листа для получения доступа к диапазону ячеек таблицы Excel.

2.6 Что такое Cells?

Свойство рабочего листа для получения доступа к ячейке таблицы Excel.

2.7 Как получить доступ к значению ячейки и диапазона?

Excel.WorkBook workBook = excelApp.WorkBook.Load("ИмяФайла.xlsx");

Excel.Worksheet workSheet= workBook.WorkSheets.First();

var range = workSheet["B2:B2"];

var cell = workSheet.Cells[1, 1];

**3 Вывод**

Мы научились выполнять создание и редактирование табличных документов на C#.

**Практическая работа №3**

**Разработка с текстовыми файлами**

**1 Цель работы**

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

**2 Контрольные вопросы**

2.1 Какое пространство имен требуется подключить для работы с Word?

using Word = Microsoft.Office.Interop.Word;

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

Word.Application wordApp = new();

2.3 Что такое Documents?

Свойство Application для создания/открытия документа Word.

2.4 Что такое Range?

Свойство документа для получения диапазона данных документа.

2.5 Что такое Selection?

Свойство документа, представляющее текущий выделенный фрагмент в окне или области.

2.6 Что такое Paragraphs?

Свойство, включающее все абзацы в выделенном фрагменте, диапазоне или документе

2.7 Что такое Tables?

Коллекция Table объектов, представляющих таблицы в выделенном фрагменте, диапазоне или документе.

**3 Вывод**

Мы научились выполнять создание и редактирование текстовых документов на C#.

**Практическая работа №4**

**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#?

Свойства проекта → Параметры → Создать или открыть параметры приложения.

2.2 Как программно считать значение параметра из настроек?

Properties.Settings.Default.ИмяПараметра;

2.3 Как программно изменить значение параметра в настройках?

Properties.Settings.Default.ИмяПараметра = "Значение";

2.4 Как выполнить сохранение значений параметров в настройках?

Properties.Settings.Default.Save();

**3 Вывод**

Мы научились сохранять настройки в клиентском приложении на C#.

**Лабораторная работа №5**

**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?

Для создания шаблона (статического ресурса) стиля для элементов управления.

2.2 Где может быть описан ControlTemplate?

Как внутри элементов управления, так и в Window.Resources.

2.3 Для чего применяется UserControl?

Для создания пользовательского элемента управления.

2.4 Каков алгоритм создания пользовательского элемента управления?

Добавление элемента UserControl в окно XAML → создание необходимых свойств элемента, таких как заголовок, описание, цветная и чёрно-белая версии изображения → установка контекста данных (DataContext) для привязки элементов разметки к свойствам → добавление элемента-контейнера для размещения элементов управления.

2.5 Как программно создать обработчик события?

Например, у кнопки прописать событие Click, название обработчика события и перейти к коду (среда разработки IDE сделает оставшуюся часть работы за пользователя).

2.6 Для чего используется свойство зависимости?

Для привязки данных, создания стилей, анимации и т.д.

**3 Вывод**

Мы научились создавать пользовательские элементы управления в приложении WPF.

**Практическая работа №6**

**Привязка данных**

**1 Цель работы**

1.1 Научиться выполнять привязку данных в приложении WPF.

**2 Контрольные вопросы**

2.1 Что такое «привязка данных»?

Привязка данных в WPF – это простой и последовательный способ представления данных и взаимодействия с ними для приложений.

2.2 Каков шаблон настройки привязки данных?

Binding = "{Binding Свойство="Значение"}"

2.3 Какой интерфейс надо реализовать для создания конвертера значений?

IValueConverter.

2.4 Какой интерфейс надо реализовать для валидации данных при привязке?

IDataErrorInfo.

**3 Вывод**

Мы научились выполнять привязку данных в приложении WPF.