OTUS

C# Developer. Basic

Длительность курса: 168 академических часов

1 Знакомство с C#

1 Вводное занятие

Цели занятия:

познакомиться с ключевыми особенностями C#;
понять как C# работает с памятью.

Краткое содержание:

что такое C#?;
объектная ориентация;
типобезопасность;
управление памятью;
поддержка платформы;
что такое CLR;
intermediate language;
почему именно C#?

---

2 Среда разработки VisualStudio: интерфейс, базовый функционал

Цели занятия:

познакомиться со средой разработки VisualStudio и узнать основы работы с ней

Краткое содержание:

установка среды разработки;
базовая структура проекта;
директивы using;
пространства имён;
main() метод;
комментарии.

---

3 **Переменные и операторы**

Цели занятия:

понять, что такое переменная;
понять, что бывают примитивные и сложные типы;
научиться работать с примитивными типами.

Краткое содержание:

что такое переменная?
примитивные и сложные;
int, byte, float, double, decimal, char, bool;
nullable типы;
наименование переменных;
объявление и инициализация переменных;
базовые операторы;
какие ещё есть операторы;
приведение типов.

---

4 **Методы, их перегрузка и расширения**

Цели занятия:

познакомиться с тем, что такое метод;
научиться писать свои методы;
научиться использовать перегрузку методов.

Краткое содержание:

основные виды методов: обычные, статические и локальные;
входные, выходные параметры;
параметры по умолчанию, params;
сигнатуры и перегрузка методов;
методы-расширения.

---

5 **Добавляем выводы и решения // ДЗ**

Цели занятия:

Управление потоком выполнения программы
if
?:
Switch
for
foreach

while
do
jump statements
break
continue
граничные условия циклов"

Краткое содержание:

условные операторы: if, else, тернарный оператор;
операторы цикла: while, for, foreach, break, continue;
оператор switch.

Домашние задания

1 Таблица в консоли

Цель: В этом задании мы закрепим знания, полученные в ходе вебинара, и сможем воспользоваться циклами и условными операторами

Нужно написать программу, которое делает следующее:

1. Вывести текст `введите размерность таблицы: ` после этого считать вводимую строку от пользователя в виде целого числа,
если введенная строка не соответствует формату целого числа (не парсится),
то нужно повторно вывести текст `введите размерность таблицы: `, и выводить
его до тех пор, пользователь не введет корректное число.

Число должно быть не меньше 1, и не больше 6. Обозначим его как `n`.

2. Вывести текст `введите произвольный текст: `, если пользователь введет пустой текст, снова выводим `введите произвольный текст: `,
до тех пор, пока пользователь не введет непустую строку.

3. Нужно вывести таблицу,

[пример работы программы]
(https://pasteboard.co/KgAioak.png)

у которой будут следующие свойства
3.1. Ее ширина не должна превышать 40 символов
3.2. Границы таблицы - символ `+`
3.3. Ширина таблицы (каждой строки) зависит от числа n и длины введенной строки из п.2.
3.4. Вывести 1ю строку таблицы с текстом, введенным в п.2., который находится на расстоянии `n-1` от каждой из границ строки.
3.5. Вывести 2ю строку таблицы. Она имеет ту же высоту, что и строка 1, и представляет собой набор символов `+`, чередующихся в шахматном порядке.
3.6. Вывести 3ю строку таблицы. Она должна быть квадратной, "перечеркнутая" символом `+` по диагоналям

В программе должны использоваться циклы do while, while и for и ?:

6 **Символы и Строки**

Цели занятия:

использовать структуру Char;
использовать класс String и StringBuilder;
объяснить назначение кодировок и работать с ними;
использовать различные функции для обработки строк;
объяснить, что такое иммутабельность и интернирование;

Краткое содержание:

ASCII, Unicode, UTF-16;
Char;
@, $;
StringBuilder;
кодировки;
строковые и символьные функции;
иммутабельность.
интернирование.

---

7 **Делаем программу интерактивной**

Цели занятия:

научиться пользоваться библиотеками и инструментами c# для работы с выводом и вводом данных в консоли.

Краткое содержание:

класс System.Console;
вывод при помощи Console.Write, Console.WriteLine;
форматирование вывода для различных типов данных;
ввод данных при помощи Console.Read, Console.ReadKey, Console.ReadLine;
обработка введенных данных при помощи Read*:
парсинг в числовое значение;
цикл do while.

---

8 **Массив и лист // ДЗ**

Цели занятия:

изучить базовую работу с коллекциями;
познакомиться с методами работы со списками;
рассмотрим типичные кейсы работы со списками и массивами.

Краткое содержание:

массив, его свойства и методы;
лист, его свойства и методы;
значимые и ссылочные типы.

Домашние задания

1 Сравнение коллекций

Цель: Сделать сравнение по скорости работы List<T>, ArrayList и LinkedList.

1. Создать коллекции List<int>, ArrayList и LinkedList<int>.
2. С помощью цикла for добавить в каждую 1 000 000 элементов (1,2,3,...).
3. С помощью Stopwatch.Start() и Stopwatch.Stop() замерить длительность заполнения каждой коллекции и вывести значения на экран.
4. Найти 496753-ий элемент, замерить длительность этого поиска и вывести на экран.
5. Вывести на экран каждый элемент коллекции, который без остатка делится на 777. Вывести длительность этой операции для каждой коллекции.

Укажите сколько времени вам понадобилось, чтобы выполнить это задание.

*Помните, что вы можете записаться на консультацию по дз к ментору:
Артур Хисматуллин - [calendly.com/artur-hismatullin]
Антон Забелин - [https://calendly.com/anton-zabelin]
Распределение студентов между менторами проверяйте в группе Slack -C#-basic-2022-04*

---

9 **Исключения и**

Цели занятия:

их обработка // ДЗ

парсинг значений из ввода в числа;
вызов исключений;
создание собственных исключений;
обработка исключений.

Краткое содержание:

класс System.Exception;
операторы throw, try, catch, finally;
порядок обработки исключений;
условные исключения.

Домашние задания

1 Решение квадратного уравнения с обработкой ошибок

Цель: Цель домашнего задания - закрепить знания о механизме работы с исключениями, полученным в ходе вебинара. Студенты научатся писать код, обрабатывающий исключения познакомятся с различными примерами встроенных исключений.

Нужно написать программу, решающую квадратное уравнение формата

a * x^2 + b * x + c = 0

Пользователю нужно ввести целые значения a, b, c

и на основе введенных значений рассчитать корни/ корень уравнения x

Шаги:
1. Вывести уравнение
a * x^2 + b * x + c = 0
2. Вывести текст

Введите значение a:

И считать значение a

3. Вывести текст
Введите значение b:

И считать значение b

4. Вывести текст
Введите значение c:

И считать значение c

5. Если какое-либо значение не является целым число
выбрасывается исключение, которое обрабатывается функцией FormatData (Приложение1)
с Severity = Error с выводом параметров, которые не прошли парсинг
[пример работы программы]
(https://pasteboard.co/KhJmoZC.png)
и возвращаемся к п.2

6. После этого нужно по формуле [решения квадратных уравнений]
(https://www.berdov.com/docs/equation/quadratic_equations/)
рассчитать все возможные значения x

6.1. Если вещественных решений - два,
вывести ответ в виде
x1 = ответ_1, x2 = ответ_2

6.2. Если решение - одно
вывести ответ в виде
x = ответ

6.3 Если вещественных решений нет - выбросить Exception с текстом "Вещественных значений не найдено",
и обработать функцией FormatData с Severity = Warning (желтый фон)

Требования к коду
- Исключения ввода данных и исключения ненахождения корней уравнения обрабатываются разными catch
- Для исключения ненахождения ответов уравнения нужно использовать собственный класс
-

доп. задание

Нужно вынести код получения данных (ф1) и код расчета (ф2) в отдельные функции,
но обработчики исключений, указанные в задании, оставить в main.

В случае появления ошибки в функции расчета (ф2) корней уравнения,
исключение обрабатывается сначала в функции расчета,
потом в main как общее исключение и текст ошибки выводится при помощи formData с Severity = Error

задание со звездочкой

Вывести исходного уравнение и значения переменных a, b, c, каждое с новой строки
также вывести указатель '>' напротив строки с a

```
a * x^2 + b * x + c = 0

> a:
b:
c:
```

при помощи клавиш стрелок вверх и вниз сделать навигацию между строками параметров a, b, c
при навигации символ '>' смещается вверх или вниз
при нажатии клавиш чисел вводятся значения для выбранного параметра,
а буква параметра заменяется в уравнении на введенное число

пример

```
a * x^2 + b * x + c = 0

> a:
b:
c:


24 * x^2 + b * x + c = 0
```

```
> a: 24
b:
c:



24 * x^2 + b * x + c = 0

a: 24
> b:
c:


24 * x^2 - 10 * x + c = 0

a: 24
> b: -10
c:
```

По нажатию enter переходим к шагам 5-6
Все исключения выводим в самом конце (под строкой "C:"

Приложение1

Функция FormatData(string message, Severity severity, IDictionary data)

где Severity - перечисление (enum)

```
enum Severity{
Warning,
Error
}
```

Функция FormatData выводит данные в следующем виде

```

---
```

message

параметр1 = значение1
параметр2 = значение2

```

где message - сообщение

Разделительная линия имеет длину 50 символов

параметр1, параметр2 - значения из Data

Если Severity = Error, выводится в консоли белый текст на красном фоне
Если Severity = Warning, выводится в консоли черный текст на желтом фоне

---

10 **Консультация общая**

**Цели занятия:**

синхронизироваться по вопросам учёбы.

**Краткое содержание:**

обратная связь;
ответы на вопросы.
```

2 ООП и анонимные типы с методами

1 **Классы как основа C# // ДЗ**

Цели занятия:

что такое ООП?
пишем свой класс
Поля
Свойства
Методы
Конструктор

ключевые слова static, partial.

Краткое содержание:

На занятии будет рассмотрено понятие класс в языке C#;
разберем, как создавать классы и объекты из них;
научимся добавлять методы и свойства,
ограничивать доступность к тем или иным свойства класса;
изучим, какие средства языка позволяют упрощать работу с классами и делать код понятным.

Домашние задания

1 Реализация класс коллекции - Стэк

Цель: Цель домашнего задания - научится проектировать классы (обычные и статические), создавать в них методы и свойства

Стэк - тип данных, представляющий собой коллекцию элементов, организованную по принципу *LIFO* - *Last In - First Out*.
Данные в эту коллекцию могут добавляться только "сверху", и извлекать тоже сверху. Если мы добавили элемент1, а потом элемент2, то доступ к Элементу1 мы получим только после того как извлечем Элемент2.

В качестве примера стека может послужить стопка тарелок: мы кладем сверху тарелки, но если мы хотим взять тарелку из середины - надо для начала снять верхние.

Основное задание
Нужно создать класс Stack у которого будут следующие свойства
1. В нем будем хранить строки
2. В качестве хранилища используйте список List<string>
3. Конструктор стека может принимать неограниченное количество входных параметров типа string, которые по порядку добавляются в стек

- Метод Add(string) - добавить элемент в стек
- Метод Pop() - извлекает верхний элемент и удаляет его из стека. При попытке вызова метода Pop у пустого стека - выбрасывать исключение с сообщением "Стек пустой"
- Свойство Size - количество элементов из Стека
- Свойство Top - значение верхнего элемента из стека. Если стек пустой - возвращать null

Пример работы
```csharp
var s = new Stack("a", "b", "c");

// size = 3, Top = 'c'
Console.WriteLine($"size = {s.Size}, Top = '{s.Top}'");

var deleted = s.Pop();

// Извлек верхний элемент 'c' Size = 2
Console.WriteLine($"Извлек верхний элемент '{deleted}' Size = {s.Size}");
s.Add("d");

// size = 3, Top = 'd'
Console.WriteLine($"size = {s.Size}, Top = '{s.Top}'");

s.Pop();
s.Pop();
s.Pop();
// size = 0, Top = null
Console.WriteLine($"size = {s.Size}, Top = {(s.Top == null ? "null" : s.Top)}");
s.Pop();
```

```

# Доп. задание 1
1. Создайте класс расширения StackExtensions и добавьте в него метод расширения Merge, который на вход принимает стек s1, и стек s2. Все элементы из s2 должны добавится в s1 в обратном порядке
Сам метод должен быть доступен в класс Stack

```
var s = new Stack("a", "b", "c")

s.Merge(new Stack("1", "2", "3"))

// в стеке s теперь элементы - "a", "b", "c", "3", "2", "1" <- верхний

```

# Доп. задание 2
1. В класс Stack и добавьте статический метод Concat, который на вход неограниченное количество параметров типа Stack
и возвращает новый стек с элементами каждого стека в порядке параметров, но сами элементы записаны в обратном порядке

```
var s =Stack.Concat(new Stack("a", "b", "c"), new Stack("1", "2", "3"), new Stack("А", "Б", "В"));

// в стеке s теперь элементы - "c", "b", "a" "3", "2", "1", "В", "Б", "А" <- верхний

```


# Доп. задание 3
Вместо коллекции - создать класс StackItem, который
- доступен только для класс Stack (отдельно объект класса StackItem вне Stack создать нельзя)
- хранит текущее значение элемента стека
- ссылку на предыдущий элемент в стеке
- Методы, описанные в основном задании
```

переделаны под работу со StackItem

---

2 **Три кита ООП: Наследование, Полиморфизм и Абстракция**

Цели занятия:

изучим механизм наследования в C#;
разберемся что такое полиморфизм и как он используется;
рассмотрим механизмы языка для работа с ключевыми
терминами ООП.

Краткое содержание:

Наследование
Пишем родительский класс
Пишем дочерний класс
Полиморфизм
GetType(), typeof()
Абстракция
Интерфейс
Модификаторы доступа
public, private, protected, internal
Виртуальные методы

---

3 **Объектно-Ориентированное Программирование (продолжение)**

Цели занятия:

сможете не только правильно описывать структуру классов, но и предоставлять доступ другим программистам к вашей структуре;
научитесь описывать свои программы так, что с ними можно будет удобно работать в команде;
начнете работу с наследованием классов, приведением типов, и применением модификаторов доступа как к классам, так и к полям класса.

Краткое содержание:

полиморфизм;
GetType(), typeof();
абстракция;
интерфейс;
модификаторы доступа;
public, private, protected, internal;
виртуальные методы.

---

4 **Интерфейсы // ДЗ**

Цели занятия:

познакомиться с понятием интерфейса
научиться применять их на практике

Краткое содержание:

понятие интерфейса
состав интерфейса
для чего он нужен
реализация интерфейсов
наследование интерфейсов
реализация интерфейсов с одинаковыми методами
константы и реализация методов по умолчанию

Домашние задания

1 Интерфейсы

Цель: В этом ДЗ вы научитесь явному вызову интерфейсов, их наследованию и реализации методов по умолчанию.

1. Создать интерфейс IRobot с публичным методами string GetInfo() и List<string> GetComponents(), а также string GetRobotType() с дефолтной реализацией, возвращающей значение "I am a simple robot.".
2. Создать интерфейс IChargeable с методами void Charge() и string GetInfo().
3. Создать интерфейс IFlyingRobot как наследник IRobot с дефолтной реализацией GetRobotType(), возвращающей строку "I am a flying robot.".
4. Создать класс Quadcopter, наследующий IFlyingRobot и IChargeable. В нём создать список компонентов List<string> _components = new List<string> {"rotor1","rotor2","rotor3","rotor4"} и возвращать его из метода GetComponents().
5. Реализовать метод Charge() должен писать в консоль "Charging..." и через 3 секунды "Charged!". Ожидание в 3 секунды реализовать через Thread.Sleep(3000).
6. Реализовать все методы интерфейсов в классах. До этого пункта достаточно "throw new NotImplementedException();"

В чат напишите также время, которое вам потребовалось для реализации домашнего задания.

---

5 **Структуры и перечисления**

Цели занятия:

более глубоко изучить перечисления, раскрыть возможности битовых операций с элементами перечислений (флаги);
узнать о типе - Структура, и структур отличиях от классов.

Краткое содержание:

сравнение классов и структур;
описание перечислений, привидение перечислений и основные операции с ними;
основные булевы операции: И, ИЛИ, НЕ, XOR;
флаги, как опция для перечислений.

---

6 **Анонимные типы, кортежи, лямбда-выражения и анонимные методы // ДЗ**

Цели занятия:

использовать анонимные типы;
использовать кортежи;
использовать анонимные методы и лямбда-выражения.

Краткое содержание:

анонимные типы;
кортежи;
анонимные методы;
лямбда-выражения;
объяснение дз.

Домашние задания

1 Анонимные типы, кортежи и лямбда-выражения

Цель: В результате этого ДЗ вы подготовите три программы (проекта).

Тренируемые навыки:
Программы 1 - анонимные типы, их вывод в консоль, сравнение между собой.
Программы 2 - кортежи.
Программы 3 - лямбда-выражения, замыкания

Программа 1.

Создать четыре объекта анонимного типа для описания планет Солнечной системы со свойствами "Название", "Порядковый номер от Солнца", "Длина экватора", "Предыдущая планета" (ссылка на объект - предыдущую планету):
- Венера
- Земля
- Марс
- Венера (снова)
Данные по планетам взять из открытых источников.
Вывести в консоль информацию обо всех созданных "планетах". Рядом с информацией по каждой планете вывести эквивалентна ли она Венере.

Программа 2.

Написать обычный класс "Планета" со свойствами "Название", "Порядковый номер от Солнца", "Длина экватора", "Предыдущая планета" (ссылка на предыдущую Планету). Написать класс "Каталог планет". В нем должен быть список планет - при создании экземпляра класса сразу заполнять его тремя планетами: Венера, Земля, Марс.

Добавить в класс "Каталог планет" метод "получить планету", который на вход принимает название планеты, а на выходе дает порядковый номер планеты от Солнца и длину ее экватора. В случае, если планету по названию найти не удалось, то этот метод должен возвращать строку "Не удалось найти планету" (именно строку, не Exception). На каждый третий вызов метод "получить планету" должен возвращать строку "Вы спрашиваете слишком часто". Таким образом на выходе из метода должны быть три поля: первые два заполнены осмысленными значениями, когда планета найдена, а последнее поле - для ошибки.

В main-методе Вашей программы создать экземпляр "Каталога планет". У этого каталога вызвать метод "получить планету", передав туда последовательно названия Земля, Лимония, Марс. Для найденных планет в консоль выводить их название, порядковый номер и длину экватора. А для ненайденных выводить строку ошибки, которую вернул метод "получить планету".

Программа 3.

Скопировать решение из программы 2, но переделать метод "получить планету" так, чтобы он на вход принимал еще один параметр, описывающий способ защиты от слишком частых вызовов - делегат PlanetValidator (можно вместо него использовать Func), который на вход принимает название планеты, а на выходе дает строку с ошибкой. Метод "получить планету" теперь не должен проверять сколько вызовов делалось ранее. Вместо этого он должен просто вызвать PlanetValidator и передать в него название планеты, поиск

которой производится. И если PlanetValidator вернул ошибку - передать ее на выход из метода третьим полем.

Из main-метода при вызове "получить планету" в качестве нового параметра передавать лямбду, которая делает всё ту же проверку, которая была и ранее - на каждый третий вызов она возвращает строку "Вы спрашиваете слишком часто" (в остальных случаях возвращает null). Результат исполнения программы должен получиться идентичный программе 2.

(*) Дописать main-метод так, чтобы еще раз проверять планеты "Земля", "Лимония" и "Марс", но передавать другую лямбду так, чтобы она для названия "Лимония" возвращала ошибку "Это запретная планета", а для остальных названий - null. Убедиться, что в этой серии проверок ошибка появляется только для Лимонии.

Таким образом, вы делегировали логику проверки допустимости найденной планеты от метода "получить планету" к вызывающему этот метод коду.

В чат напишите также время, которое вам потребовалось для реализации домашнего задания.

---

7 **Консультация общая**

Цели занятия:

синхронизироваться по вопросам учёбы.

Краткое содержание:

обратная связь;
ответы на вопросы.

3 Алгоритмы и стили кодирования

1 Анализ сложности алгоритмов и сортировка

Цели занятия:

познакомиться с O-нотацией;
понять, зачем она нужна и как можно сравнивать алгоритмы;
посмотреть разницу в различных алгоритмах сортировок.

Краткое содержание:

сложность алгоритма;
сортировка;
практика.

2 **Взаимосвязь циклов и рекурсии // ДЗ**

Цели занятия:

познакомиться с видами рекурсии
понять в каких случаях эффективнее применять циклы, а в каких - рекурсии
сравнить алгоритмическую сложность циклов и рекурсий

Краткое содержание:

как эффективнее писать циклы;
внутренняя оптимизация работы циклов;
отличие циклов от рекурсии;
какие есть нюансы работы с рекурсиями;
какие рекурсии можно заменить циклами;
стоит ли заменять рекурсии циклами;
когда без рекурсии не обойтись;
и т.д.

Домашние задания

1 Циклы и рекурсии

Цель: В этом задании вы сами попробуете создать алгоритм использующий рекурсию и его аналог с использованием цикла. Опционально вы можете проверить скорость работы этих двух алгоритмов.

1. Реализовать метод нахождения n-го члена последовательности Фибоначчи по формуле F(n) = F(n-1) + F(n-2) с помощью рекурсивных вызовов.
2. Реализовать метод нахождения n-го члена последовательности Фибоначчи по формуле F(n) = F(n-1) + F(n-2) с помощью цикла.
3. Добавить подсчёт времени на выполнение рекурсивного и итеративного методов с помощью Stopwatch и написать сколько времени для значений 5, 10 и 20.

---

3 **Деревья и кучи // ДЗ**

Цели занятия:

рассказать базовую теорию работы с деревьями и кучей;
построить и обойти дерево в C#;

построить на C# бинарное дерево поиска и найти в нем значение.

Краткое содержание:

общая теория деревьев;
представление деревьев;
обход дерева;
бинарное дерево поиска;
куча.

Домашние задания

1 Деревья и кучи

Цель: Тренируемые навыки:
- построение бинарного дерева поиска
- его обход в симметричном порядке
- поиск элемента по бинарному дереву

Напишите программу, которая:
- принимает на вход из консоли информацию о сотрудниках: имя + зарплата (имя в первой строке, зарплата в виде целого числа во второй строке; и так для каждого сотрудника, пока пользователь не введет пустую строку в качестве имени сотрудника)
- попутно при получении информации о сотрудниках строится бинарное дерево с этой информацией, где в каждом узле хранится имя сотрудника, а его зарплата является значением, на основе которого производится бинарное разделение в дереве
- после окончания ввода пользователем программа выводит имена сотрудников и их зарплаты в порядке возрастания зарплат (в каждой строчке формат вывода "Имя - зарплата"). Использовать для этого симметричный обход дерева.
- после этого программа запрашивает размер зарплаты, который интересует пользователя. В построенном бинарном дереве программа находит сотрудника с указанной зарплатой и выводит его имя. Если сотрудник не найден - выводится "такой сотрудник не найден"
- после этого программа предлагает ввести цифру 0 (переход к началу программы) или 1 (снова поиск зарплаты). При вводе 0 должен произойти переход к началу работы программы, т.е. опять запрашивается список сотрудников и строится

		новое дерево. При вводе 1 должны снова запросить зарплату, которую хочется поискать в дереве - см.предыдущий пункт.
4	**Системы контроля версий**	**Цели занятия:** познакомиться с понятием системы контроля версий научиться пользоваться ей на примере Git + GitHub **Краткое содержание:** системы контроля версий; Git; CLI and GUI; Git + GitHub (Demo).
5	**Code style от Майкрософт, DRY/DIE, Yagni, KISS**	**Цели занятия:** познакомиться с наиболее популярными конвенциями и подходами написания кода на C# **Краткое содержание:** Text formatting; Microsoft coding convention; DRY/DIE, boilerplate, KISS; заключение.
6	**Консультация общая**	**Цели занятия:** синхронизироваться по вопросам учёбы. **Краткое содержание:** обратная связь; ответы на вопросы.

4 Пользовательский интерфейс

1 **Знакомство с Telegram API**

2 **Различные виды клавиатур**

3 **Занятие по внутренней организации приложения**

4 **Делегаты, Event-ы, добавляем асинхронное выполнение // ДЗ**

Цели занятия:

научиться использовать паттерн "наблюдатель" через events;
запускать таски и треды и понимать базовые принципы их работы.

Краткое содержание:

делегаты;
observer, events;
параллелизм;
async / await.

Домашние задания

1 Делегаты, Event-ы, добавляем асинхронное выполнение

Цель: Тренируемые навыки:
работа с событиями
избегание блокировок через асинхронные вызовы
базовая работа с формой в WPF

1) Напишите класс ImageDownloader. В этом классе должен быть метод Download, который скачивает картинку из интернета. Для загрузки картинки

можно использовать примерно такой код:
https://dotnetfiddle.net/5oT1Hi

```
// Откуда будем качать
string remoteUri = "https://effigis.com/wp-content/uploads/2015/02/Iunctus_SPOT5_5m_8bit_RGB_DRA_torngat_mountains_national_park_8bits_1.jpg";
// Как назовем файл на диске
string fileName = "bigimage.jpg";

// Качаем картинку в текущую директорию
var myWebClient = new WebClient();
Console.WriteLine("Качаю \"{0}\" из \"{1}\" .....\n\n", fileName, remoteUri);
myWebClient.DownloadFile(remoteUri, fileName);
Console.WriteLine("Успешно скачал \"{0}\" из \"{1}\"", fileName, remoteUri);
```

2) Создайте экземпляр этого класса и вызовите скачивание большой картинки, например, https://effigis.com/wp-content/uploads/2015/02/Iunctus_SPOT5_5m_8bit_RGB_DRA_torngat_mountains_national_park_8bits_1.jpg

В конце работы программы выведите в консоль "Нажмите любую клавишу для выхода" и ожидайте нажатия любой клавиши.

3) Добавьте события: в классе ImageDownloader в начале скачивания картинки и в конце скачивания картинки выкидывайте события (event) ImageStarted и ImageCompleted соответственно.

В основном коде программы подпишитесь на эти события, а в обработчиках их срабатываний выводите соответствующие уведомления в консоль: "Скачивание файла началось" и "Скачивание файла закончилось".

4) Сделайте метод ImageDownloader.Download асинхронным. Если Вы скачивали картинку с использованием WebClient.DownloadFile, то используйте теперь WebClient.DownloadFileTaskAsync - он возвращает Task.

В конце работы программы выводите теперь текст "Нажмите клавишу A для выхода или любую другую клавишу для проверки статуса скачивания" и ожидайте нажатия любой клавиши. Если нажата клавиша "A" - выходите из программы. В противном случае выводите состояние загрузки картинки (True - загружена, False - нет). Проверить состояние можно через вызов Task.IsCompleted.

Поздравляю! Ваша загрузка картинки работает асинхронно с основным потоком консоли.

5) Создайте WPF приложение и перенесите туда эту же логику:
- начало скачивания картинки по нажатии на кнопку на форме
- вывод текущего состояния загрузки по нажатии на другую кнопку на форме
- события начала и окончания загрузки выводятся текстом ниже кнопок

5 **Работа с файлами // ДЗ**

Цели занятия:

научиться работать с файлами и каталогами в своих приложениях

Краткое содержание:

основные методы работы с файлами и директориями;
потоки для записи в файлы;
асинхронная работа с файлами.

Домашние задания

1 Создание консольного приложение, записывающее и считывающее информацию в\из файл(а).

Цель: компилирующееся без ошибок приложение, файлы по заданному пути, консоль со значениями файлов.

1. Создать директории c:\Otus\TestDir1 и c:\Otus\TestDir2 с помощью класса DirectoryInfo.
2. В каждой директории создать несколько файлов File1...File10 с помощью класса File.
3. В каждый файл записать его имя в кодировке UTF8. Учесть, что файл может быть удален, либо отсутствовать права на запись.
4. Каждый файл дополнить текущей датой (значение DateTime.Now) любыми способами: синхронно и\или асинхронно.
5. Прочитать все файлы и вывести на консоль: имя_файла: текст + дополнение.

---

6 **Консультация общая**

Цели занятия:

синхронизироваться по вопросам учёбы.

Краткое содержание:

обратная связь;
ответы на вопросы.

5 Работа с коллекциями

1 Основные коллекции: массив, список, связный список

2 **Основные коллекции: очередь, стек, словарь, хешсет // ДЗ**

Краткое содержание:

внутреннее устройство FIFO, LIFO O(n) нотация;
различие скорости записи и чтения;
хеш-функция;
механизм разрешения коллизий.

Домашние задания

1 Реализуем свой собственный словарь

Цель: Реализуйте класс OtusDictionary, который позволяет оперировать int-овыми значениями в качестве ключей и строками в качестве значений. Для добавления используйте метод void Add(int key, string value), а для получения элементов - string Get(int key). Внутреннее хранилище реализуйте через массив. При нахождении коллизий, создавайте новый массив в два раза больше и не забывайте пересчитать хеши для всех уже вставленных элементов. Метод GetHashCode использовать не нужно и массив/список объектов по одному адресу создавать тоже не нужно (только один объект по одному индексу). Словарь не должен давать сохранить null в качестве строки, соответственно, проверять заполнен элемент или нет можно сравнивая строку с null.

1. Реализуйте метод Add с неизменяемым массивом размером 32 элемента (исключение, если уже занято место).
2. Реализуйте метод Get.
3. Реализуйте увеличение массива в два раза при нахождении коллизий.
4. Убедитесь, что класс работает без ошибок (например, Get несуществующего элемента) не бросает исключений, помимо заданных вами. Если это не так, то доработайте.
5. Добавьте к классу [возможность работы с индексатором](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/indexers/).

3 **Generic коллекции**

Цели занятия:

узнать, для чего нам нужны Generics;
узнать, как создавать собственные Generic классы.

Краткое содержание:

предназначение Generics;
создание своей дженерик-коллекции;
создание Generic интерфейсов;
создание Generic методов.

---

4 **Observable, Immutable и Concurrent коллекции // ДЗ**

Цели занятия:

использовать рассмотренные коллекции.

Краткое содержание:

паттерн Observer;
ObservableCollection;
Systems.Collections.Immutable;
Systems.Collections.Concurrent;
объяснение дз.

Домашние задания

1 Observable, Immutable и Concurrent коллекции

Цель: Тренируемые навыки: работа с
- ObservableCollection
- Immutable коллекциями
- ConcurrentDictionary

(1) Напишите программу "Постоянный покупатель" с двумя классами:
- Shop (Магазин)
- Customer (Покупатель)

В классе Shop должна храниться информация о списке товаров (экземпляры классов Item). Также в классе Shop должны быть методы Add (для добавления товара) и Remove (для удаления товара).
В классе Item должны быть свойства Id

(идентификатор товара) и Name (название товара).

В классе Customer должен быть метод OnItemChanged, который будет срабатывать, когда список товаров в магазине обновился. В этом методе надо выводить в консоль информацию о том, какое именно изменение произошло (добавлен товар с таким-то названием и таким-то идентификатором / удален такой-то товар).

В основном файле программы создайте Магазин, создайте Покупателя. Реализуйте через ObservableCollection возможность подписки Покупателем на изменения в ассортименте Магазина - все изменения сразу должны отображаться в консоли (должен срабатывать метод Customer.OnItemChanged).

По нажатии клавиши A добавляйте новый товар в магазин. Товар должен называться "Товар от <текущее дата и время>", где вместо <текущее дата и время> подставлять текущую дату и время. По нажатии клавиши D спрашивайте какой товар надо удалить. Пользователь должен ввести идентификатор товара, после чего товар необходимо удалить из ассортимента магазина. По нажатии клавиши X выходите из программы.

Добавьте в Магазин несколько товаров, удалите какие-то из них - убедитесь, что сообщения выводятся в консоль.

(2) Напишите программу "Библиотекарь". Суть:
- Пользователю в консоли показывают меню: "1 - добавить книгу; 2 - вывести список непрочитанного; 3 - выйти"
- Если он вводит 1, то далее ему пишут "Введите название книги:". Пользователь вводит название - книга запоминается в коллекции. В качестве коллекции стоит использовать ConcurrentDictionary<string, int> (для чего нужен int - см.далее). Если книга с таким названием уже была добавлена ранее - не добавлять и не обновлять ее. Автоматически возвращаемся в меню (снова выводим его в консоль).
- Если вводит 2 - на экран выводится список всех ранее введенных книг и в конце - опять меню
- Если вводит 3 - выходим из программы.

В выводимом списке книг надо выводить не только

их названия, но и вычисленный процент, насколько она прочитана. Например: "Остров сокровищ - 15%".

Для расчета процентов создаем второй поток, который в фоне постоянно перевычисляет проценты. Между каждой итерацией перевычисления он спит 1 секунду. Во время итерации перевычисления он берет коллекцию всех книг и по каждой вычисляет новый процент путем прибавления 1% к предыдущему значению (изначально 0%). Если дошли до 100% - удаляем эту книгу из списка.

Таким образом, когда пользователь вызовет вывод списка он может получить что-то вроде:
Любовь к жизни - 45%
Приключения Мюнхгаузена - 17%
Незнайка в Солнечном городе - 4%

(3) Напишите программу "Дом, который построил Джек".
- В программе должна быть коллекция строк. Каждая строка - строка стихотворения "Дом, который построил Джек".
- Изначально коллекция пустая
- Также в программе есть 9 классов - Part1, Part2, Part3, ..., Part9
- В каждом классе PartN есть метод AddPart, который на вход принимает коллекцию строк, добавляет в нее новые строки и сохраняет получившуюся коллекцию в свойство "Poem". Требуется это делать так, чтобы исходная коллекция не изменилась. Какие именно строки добавляет каждый класс посмотрите здесь - https://russkaja-skazka.ru/dom-kotoryiy-postroil-dzhek-stihi-samuil-marshak/ (например Part3 добавляет третий параграф стихотворения)
- Надо создать экземпляры этих классов, а затем последовательно вызвать каждый из методов AddPart, передавай в него результат вызова предыдущего метода, примерно так: 'MyPart3(MyPart2.Poem)'
- В конце работы программы надо вывести значение свойства "Poem" у каждого из классов и убедиться, что изменяя коллекцию в одном классе Вы не затрагивали коллекцию в предыдущем.

5 **LINQ запросы**

Краткое содержание:

fluent syntax;
выражения запросов;
отложенное выполнение;
подзапросы;
два синтаксиса;
подзапросы;
стратегии композиций;
стратегии проекций;
=== LINQ операторы ===
фильтрация;
проектирование;
объединение;
упорядочивание;
группировка;
преобразования;
агрегация;
квантификация;
генерация.

6 **LINQ операторы // ДЗ**

Домашние задания

1 Метод для LINQ

Цель: 1. Напишите свой метод расширения с названием "Top" для коллекции IEnumerable<T>, принимающий значение X от 1 до 100 и возвращающий заданное количество процентов от выборки с округлением количества элементов в большую сторону.
То есть для списка var list = new List<int> {1,2,3,4,5,6,7,8,9};
list.Top(30) должно вернуть 30% элементов от выборки по убыванию значений, то есть [9,8,7] (33%), а не [9,8] (22%).
Если переданное значение больше 100 или меньше 1, то выбрасывать ArgumentException.

2. Напишите перегрузку для метода "Top", которая принимает ещё и поле, по которому будут отбираться топ X элементов. Например, для var list = new List<Person>{...}, вызов list.Top(30, person => person.Age) должен вернуть 30% пользователей с наибольшим возрастом в порядке убывания оного.

1. Создайте дженерик метод расширения для IEnumerable<T>, возвращающий коллекцию, на которой был вызван;
2. Ограничьте количество элементов выходной коллекции;
3. Создайте дженерик перегрузку метода Top, добавив для этого одним из параметров функцию, принимающую T и возвращающую int;
4. Сделайте код-ревью (напишите свой отзыв) на одну из работ других студентов. Ссылки можете попросить в слаке. Для первого студента этот пункт опциональный (хотя и желательный), так как пока нет других работ.

7	**Консультация общая**	**Цели занятия:** синхронизироваться по вопросам учёбы. **Краткое содержание:** Краткое содержание обратная связь; ответы на вопросы.

6 Работа с базами данных

1 **Введение в базы данных**

Цели занятия:

таблицы (CRUD), столбцы (CRUD), типы данных, insert, select *

2 **Выборки данных // ДЗ**

Краткое содержание:

PostgreSQL;
графические клиенты для СУБД;
бесплатный хостинг PostgreSQL;
восстановление БД на локальном компьютере;
операторы запросов SQL: SELECT, JOIN, GROUP BY, HAVING;
порядок выполнения SQL запросов.

Домашние задания

1 SQL запросы

Цель: 1. Установить на локальный компьютер PostgreSQL (см. ссылки в презентации).
2. Скачать и восстановить БД dvdrental с помощью pgAdmin (см. ссылки в презентации). Приложить скриншот с pgAdmin.
3. В pgAdmin (или Navicat) написать SQL запрос, который вернет названия и описания всех фильмов, продолжительность которых больше 100. Приложить скриншот с SQL запросом и результирующей выборкой.
4. В pgAdmin (или Navicat) написать SQL запрос, который вернет уникальные имена (без фамилий) актеров. Приложить скриншот с SQL запросом и результирующей выборкой.
5. В pgAdmin (или Navicat) написать SQL запрос, который вернет рейтинг фильма и количество фильмов с таким рейтингом, но только для тех рейтингов, которые содержат букву "G". Приложить скриншот с SQL запросом и результирующей выборкой.
6. В pgAdmin (или Navicat) написать SQL запрос, который вернет имена и фамилии только тех актеров, которые снимались менее, чем в 20 фильмах. Приложить скриншот с SQL запросом и результирующей выборкой.

См. вебинар.

3 **Хранимые процедуры и функции**

Краткое содержание:

хранимые процедуры и функции;
триггеры;
кастомные типы данных;
временные таблицы;
представления и материализованные представления.

4 **Индексы: кластерный и не кластерный // ДЗ**

Цели занятия:

рассказать, зачем нужны индексы;
рассказать о разнице между кластерным и некластерным индексами;
использовать индексы на практике.

Краткое содержание:

индексы и зачем они нужны;
как добавить индекс;
составной индекс;
кластерный и некластерный индекс;
B-tree.

Домашние задания

1 Индексы: кластерный и не кластерный

Цель: Тренируем навык работы с таблицами, связями между ними индексами

1. В СУБД PostgreSQL создать БД Shop
2. Создать таблицы Customers (ID, FirstName, LastName, Age), Products (ID, Name, Description, StockQuantity, Price) и Orders (ID, CustomerID, ProductID, Quantity)
3. Установить между ними соответствующие связи по внешним ключам (в каждой таблице поле ID является первичным ключом)
4. Заполнить таблицы произвольными значениями (с корректными значениями для внешних ключей). В каждой таблице не менее 10 записей.
5. Написать запрос, который возвращает список всех пользователей старше 30 лет, у которых есть заказ на продукт с ID=1. Используйте alias, чтобы дать столбцам в результирующей выборке понятные названия. В результате должны получить таблицу:
CustomerID, FirstName, LastName, ProductID, ProductQuantity, ProductPrice
6. Убедитесь, что вы повесили необходимый некластерный индекс (он не особо нужен, когда у вас 10 записей, но пригодится, если бы их было 1000)

5 **Linq2DB, Dapper // ДЗ**

Краткое содержание:

ORM;
Dapper;
Linq2SQL.

Домашние задания

1 Dapper / Linq2sql

Цель: Получить навык работы с БД из программы, получения выборки и последующей ее обработки. Получить опыт работы с ORM Dapper (Linq2Db)

0. Определиться с ORM: Dapper (Linq2Db).
1. Выбрать какую БД использовать (из задания "Sql запросы" или "Кластерный индекс"), написать строку подключения к БД и использовать ее для подключения. (опираться можно на пример из материалов)
2. Создать классы, которые описывают таблицы в БД
3. Используя ORM выполнить простые запросы к каждой таблице, выполнить параметризованные запросы к каждой таблице (без JOIN) - 2-3 запроса на таблицу.
Значения параметров для фильтрации можно как задавать из консоли, так и значениями переменных в коде. (пример GetStudent)
4. Выполнить все запросы, из выбранного ранее задания с передачей параметров.

---

6 **Консультация общая**

Цели занятия:

синхронизироваться по вопросам учёбы.

Краткое содержание:

обратная связь;
ответы на вопросы.

7 Итоговый модуль

1 Консультация по проектам

Цели занятия:

получить ответы на вопросы по проекту, ДЗ и по курсу.

Краткое содержание:

вопросы по улучшению и оптимизации работы над проектом;
затруднения при выполнении ДЗ;
вопросы по программе.

Домашние задания

1 Телеграмм бот

Цель: В этом проекте вы создадите приложение-бота, спроектируете базу данных для него, а также подключите его к Телеграмм API.

1. Определитесь с темой.
2. Напишите какие в вашем боте будут роли и их функции.
3. Выпишите все запросы и ответы для функций из пункта 2.
4. Составьте минимальный набор ролей, функций и запросов (MVP).
5. Реализуйте все запросы из минимального списка с помощью тех инструментов, что вы уже прошли (может не быть базы данных (вместо неё List<>) и подключения к телеграмму, достаточно ввода данных пользователем через консоль.
6. С помощью интерфейсов добавьте реализации классов, позволяющих работать с базами данных вместо списков и телеграмм API вместо консоли.
7. Реализуйте полностью по одному все запросы из списка, составленного в пункте 3.

2 **Защита проектов**

Цели занятия:

защитить проект и получить рекомендации экспертов.

Краткое содержание:

презентация проектов перед комиссией;
вопросы и комментарии по проектам.

*В защите могут участвовать и студенты, не выполняющие собственного проекта, но желающие принять участие в обсуждении проектов своих коллег.