OTUS

C# Developer. Basic

Длительность курса: 178 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 Code: интерфейс, базовый функционал

Цели занятия:

"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 типы примитивные и сложные
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 Массив и лист

Цели занятия:

массив
Его свойства и методы
Лист
Его свойства и методы
Значимые и ссылочные типы

Домашние задания

1 Ступенчатый массив из двумерных массивов

Цель: Создать ступенчатый массив из двумерных массивов со списками всех возможных ходов шахматных фигур.

Создать перечисление для шахматных фигур: ладья, конь, слон, ферзь, король.
Создать структуру для хранения координат полей шахматной доски (x, y).
Создать ступенчатый массив:
первый аргумент - фигура.
второй аргумент - матрица из двух координат (x, y) полей шахматной доски.

Сгенерировать список всех возможных ходов указанной шахматной фигуры
с указанного поля, поместить эти координаты в список и
сохранить в соответствующем элементе массива.

Используя сгенерированный массив, написать код для ответа на вопросы:
1. Сколько всего возможных ходов у всех фигур со всех полей?
2. Сколько всего возможных ходов на поле a1, координата (0, 0)
3. Выписать координаты для каждой фигуры, с которого меньше всего возможных ходов.

* Сгенерировать c# код с непосредственным описанием элементов массива.

5 **Делаем программу интерактивной**

Цели занятия:

научится 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

---

6 **Добавляем выводы и решения**

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 ?:

7 **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.

---

8 **Исключения и их обработка**

Цели занятия:

"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

---

9 **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

---

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#**

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**	**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)**	

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)
Рассказать 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**

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	**Циклы и рекурсия**	**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** 1ДЗ 8
2	**Анализ сложности алгоритмов и сортировка**	**Домашние задания** 1ДЗ
3	**Деревья и кучи**	**Домашние задания** 1ДЗ 9
4	**Системы контроля версий**	**Домашние задания** 1ДЗ
5	**Code style от Майкрософт, DRY/DIE, Yagni, KISS**	

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

4 Пользовательский интерфейс

1	**Windows Presentation Foundation #1**	**Цели занятия:** переводим программу в UI, добавляем интерфейс на WPF **Домашние задания** 1ДЗ
2	**Windows Presentation Foundation #2**	
3	**Занятие по UI-потоку и внутренней организации приложения**	**Цели занятия:** разделение на UI и остальные потоки
4	**Делегаты, Event-ы, добавляем асинхронное выполнение**	**Домашние задания** 1ДЗ
5	**Работа с файлами**	**Цели занятия:** чтение и запись в CSV файл Увеличиваем объёмы данных **Домашние задания** 1ДЗ

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**	**Цели занятия:** "Внутреннее устройство FIFO, LIFO O(n) нотация Различие скорости записи и чтения Хеш-функция Механизм разрешения коллизий" **Домашние задания** 1ДЗ
3	**Generic коллекции**	**Цели занятия:** создание своей дженерик-коллекции
4	**Observable, Immutable и Concurrent коллекции**	**Домашние задания** 1ДЗ

5 **LINQ запросы**

Цели занятия:

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**

Домашние задания

1 ДЗ

6 Работа с базами данных

1	**Введение в базы данных**	**Цели занятия:** таблицы (CRUD), столбцы (CRUD), типы данных, insert, select *
2	**Выборки данных**	**Цели занятия:** - SELECT, JOIN, GROUP BY, HAVING **Домашние задания** 1ДЗ
3	**Хранимые процедуры и функции**	**Цели занятия:** плюс, свои типы данных, временные таблицы # и @
4	**Индексы: кластерный и не кластерный**	**Домашние задания** 1ДЗ
5	**Linq2DB, Dapper**	**Домашние задания** 1ДЗ
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 Проект

---

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.