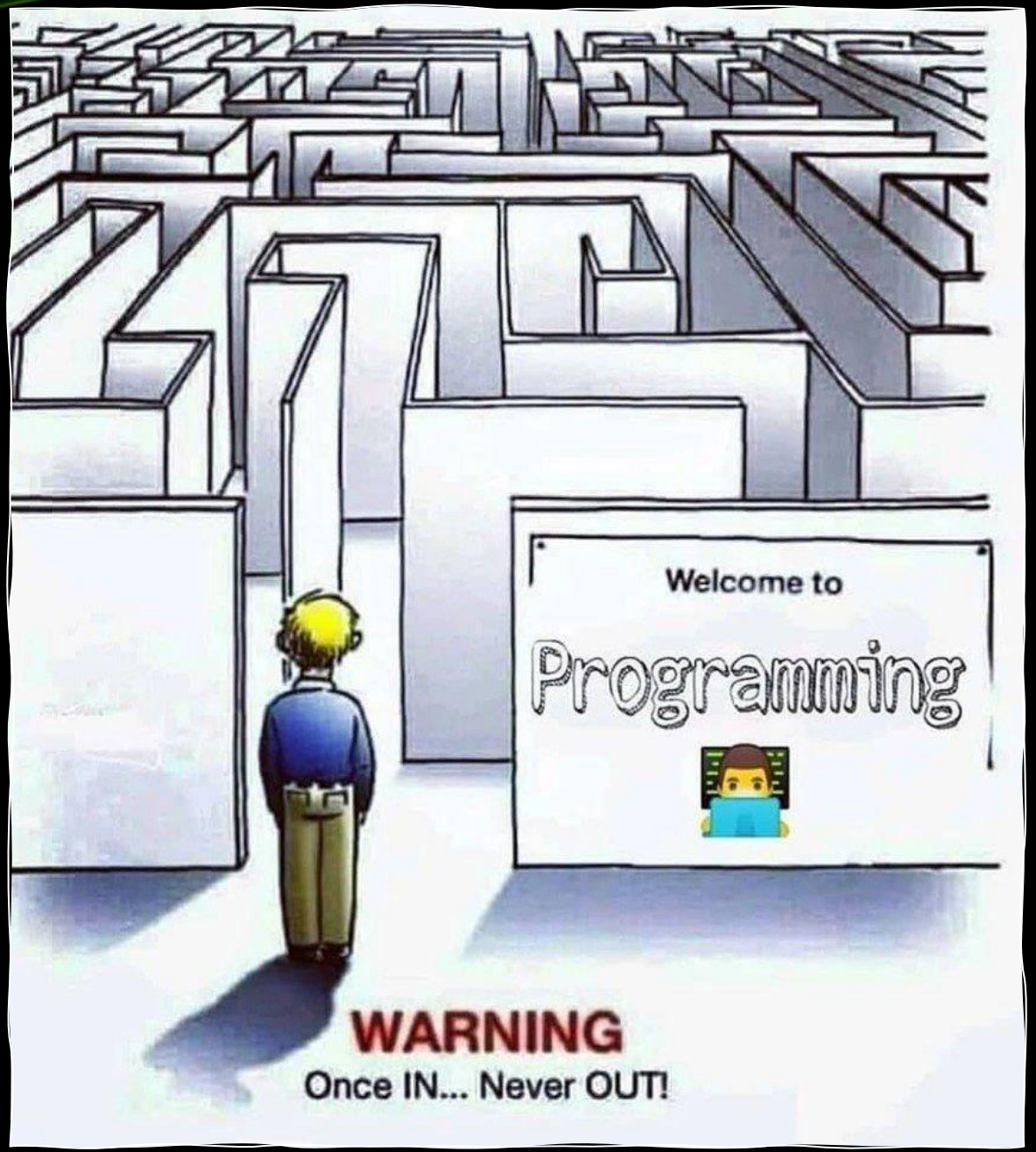
ЛЕКЦИЯ 1

- 06.09.2023
- Знакомство с преподавателями
- Структура и инфраструктура дисциплины
- Современные языки программирования
- Структура программ на C#


Welcome to
Programming
WARNING
Once IN... Never OUT!

ЦЕЛИ ЛЕКЦИИ

- Познакомиться с преподавателями и общей структурой дисциплины
- Получить общие представления о языке программирования C#
- Начать знакомство с терминологией программирования

ПРЕПОДАВАТЕЛИ ДИСЦИПЛИНЫ

Кто ведёт семинары?

Кот Вася вести семинары отказался ☹

ПОЗНАКОМИМСЯ

Лесовская Ирина Николаевна,

канд.техн.наук

- Доцент департамента программной инженерии ФКН НИУ ВШЭ
- Стаж работы в НИУ ВШЭ 18 лет (Факультет бизнес-информатики, Отделение программной инженерии ФБИ, Факультет бизнеса и менеджмента, ФДП, Лицей НИУ ВШЭ)
- 2020-2022: Microsoft RUS, Академия ITHub

ПОЗНАКОМИМСЯ

Бегичева Антонина Константиновна

- Преподаватель департамента программной инженерии ФКН НИУ ВШЭ (с 2021 г.)
- Аспирант ДПИ (с 2022 г.)
- Стажер-исследователь лаборатории ПОИС (always)

Контакты

- Личная страница - https://www.hse.ru/staff/akbegicheva
- Почта – abegicheva@hse.ru
- tg – @tonyaginger

ПОЗНАКОМИМСЯ

Головин Леонид Олегович

- Разработчик в группе разработки платформы для разработчиков Яндекс.Игр
- Старший преподаватель РТУ “МИРЭА” ИКБ кафедры “Безопасность программных решений”
- Эксперт по интеграции программы обучения Яндекс.Академии для РТУ “МИРЭА”
- Был тимлидом бэкэнд-разработки игр Инди-кот 2, Family Hotel, Family Town, Безумие в компании PlayFlock
- Закончил магистратуру “Информационные системы и технологии” в 2019

ПОЗНАКОМИМСЯ

Резуник Людмила Александровна

- Выпускница ФКН ПИ (2023)
- Младший iOS-разработчик в отделе мобильных приложений Цифрового блока ВШЭ (2022-23)
- Научный сотрудник научно-учебной лаборатории облачных и мобильных технологий (с 2023)

ПОЗНАКОМИМСЯ

Глушко Александр Александрович

- Приглашенный преподаватель департамента программной инженерии ФКН НИУ ВШЭ
- Научный сотрудник международной лаборатории интеллектуальных систем и структурного анализа ФКН НИУ ВШЭ
- Разработчик в Schlumberger Reasearch

ПОЗНАКОМИМСЯ

Соколовский Вацлав Антонович

- Приглашенный преподаватель департамента программной инженерии ФКН НИУ ВШЭ (с 2023 г.)
- Студент магистерской программы "Науки о данных" ФКН НИУ ВШЭ
- Разработчик в Delta Solutions (с 2021 г.)

Контакты:

- Почта - rinokusdev@gmail.com
- Telegram - @RinokuS

ПОЗНАКОМИМСЯ

- Чапкин Николай
- Выпускник прикладной математики
- В разработке 30 лет (а если считать школу - 35)
- Руководство командами разработки с 2005
- Проектное, продуктовое управление с 2007
- Работаю в Большой зеленой компании ;)
- Преподаю с 1998

ПОЗНАКОМИМСЯ

Максименкова Ольга Вениаминовна, канд. техн. наук

- Доцент Департамента программной инженерии
- В ВШЭ с 2009 года
- Старший аналитик студии игровой разработки «Винторог»
- Соавтор и переводчик книг по программированию, информатике и разработке игр

Часы консультаций:
Понедельник 9:30-10:50
Пятница 14:30 – 15:30

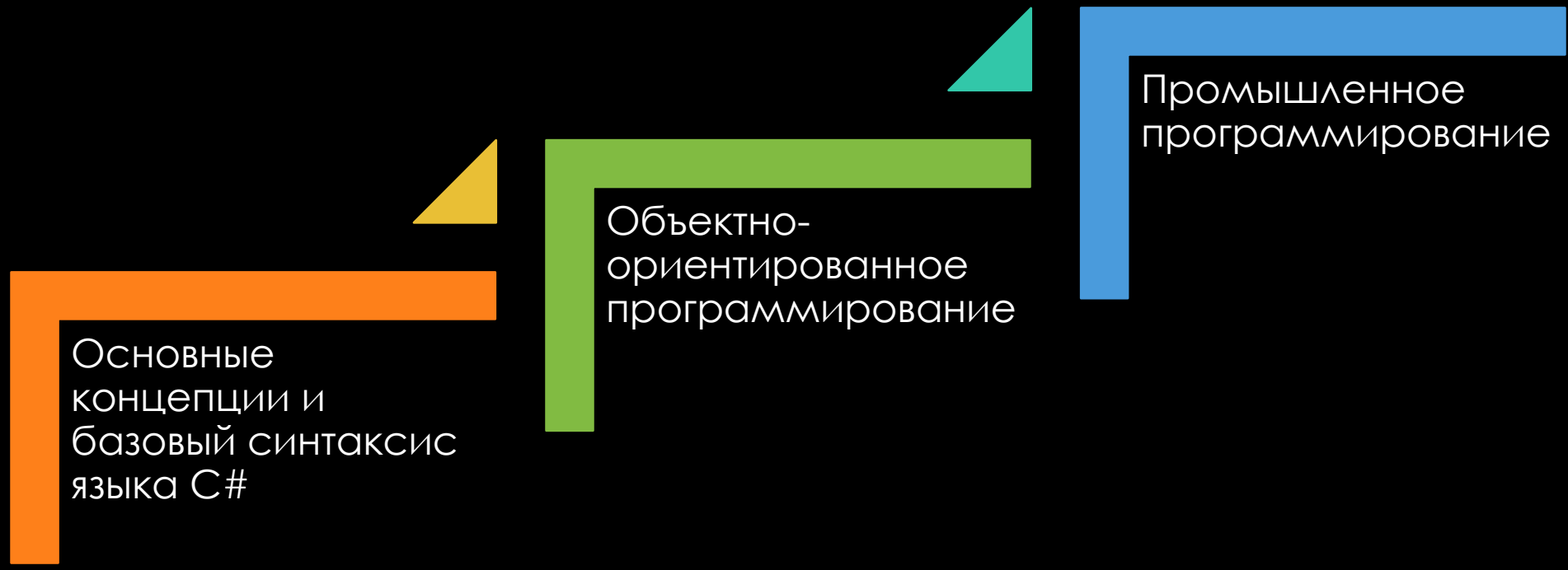
О ДИСЦИПЛИНЕ

Структура и календарный план

Формы контроля

Оценивание

ОБЩАЯ СТРУКТУРА ДИСЦИПЛИНЫ



ВИДЫ ПРОВЕРОЧНЫХ РАБОТ

- **Самостоятельная**
  - **В классе, 20-30 минут**
- **Контрольная**
  - **В классе 60-80 минут**
- **Контрольное домашнее задание (КДЗ) оно же Проект**
  - **Дома 2-3 недели**
- **Тест**
  - **На экзамене, 40-60 минут**

ГРАФИК КОНТРОЛЯ, 1 МОДУЛЬ

Неделя	Контроль		
1			
2		СР_1_1	
3			
4		СР_1_2	КД3_1
5			
6		СР_1_3	
7	КР_1		
ЭКЗ	ЭТ1		

Накоп за модуль:
ОКРУГЛЕНИЕ (
(СР_1_1+ СР_1_2+ СР_1_1+КД3_1_1+КР_1)
)

Итог за модуль:
ОКРУГЛЕНИЕ (0,8 * Н1+ 0,2 * ЭТ1)

Экзаменационный тест блокирующий

КРИТЕРИИ ВЫСТАВЛЕНИЯ ОЦЕНКИ ЗА ПРОГРАММУ

- **НЕУДОВЛЕТВОРИТЕЛЬНО**
- 1 балл:
  - 1. Разработка программы не завершена.
  - 2. Программа имеет синтаксические ошибки (не компилируется).
- 2 балла:
  - 1. Программа не решает основную задачу или не соответствует спецификации.
  - 2. В программе обнаруживаются не обработанные исключения при решении основных и второстепенных подзадач.
- 3 балла:
  - 1. Программа не решает основную задачу при некоторых вариантах исходных данных.
  - 2. Программа завершается аварийно при некоторых вариантах исходных данных.
-

КРИТЕРИИ ВЫСТАВЛЕНИЯ ОЦЕНКИ ЗА ПРОГРАММУ

- **УДОВЛЕТВОРИТЕЛЬНО**
- 4 балла:
- 1. Программа решает основную задачу, но имеет отклонения от спецификации.
- 5 баллов:
- 1. Программа соответствует критериям получения оценки 4 балла.
- 2. Программа соответствует отдельным дополнительным критериям.

КРИТЕРИИ ВЫСТАВЛЕНИЯ ОЦЕНКИ ЗА ПРОГРАММУ

- **ХОРОШО**
- 6 баллов:
- 1. Программа решает поставленную задачу и соответствует спецификации. Отклонения от спецификации допущены при реализации второстепенных подзадач.
- 2. Исходный текст документирован.
- 7 баллов:
- 1. Программа соответствует критериям получения оценки 6 баллов.
- 2. Программа в целом соответствует дополнительным критериям.

КРИТЕРИИ ВЫСТАВЛЕНИЯ ОЦЕНКИ ЗА ПРОГРАММУ

- **ОТЛИЧНО**
- 8 баллов:
  - 1. Программа решает все поставленные задачи и полностью соответствует спецификации.
  - 2. Студент в комментариях обосновал принятые конструктивные решения.
  - 3. Исходный текст документирован. Присутствуют сведения о назначении используемых переменных, параметров, методов, классов, объектов.
  - 4. Программа остается работоспособной при вводе неверных исходных данных.
  - 5. Предусмотрено повторное решение задачи без повторного запуска программы.
  - 6. Программа реализована по модульному принципу и хорошо декомпозирована.

КРИТЕРИИ ВЫСТАВЛЕНИЯ ОЦЕНКИ ЗА ПРОГРАММУ

- **ОТЛИЧНО+** и **ОТЛИЧНО++**
- 9 баллов:
  - 1. Программа соответствует критериям получения оценки 8 баллов.
  - 2. Программа соответствует некоторым дополнительным критериям.

- 10 баллов:
  - 1. Программа соответствует критериям получения оценки 8 баллов.
  - 2. Программа полностью соответствует всем дополнительным критериям.
  - 3. Студент отразил в комментариях возможность альтернативных вариантов решения задачи.

КОПИЯ МАТЕРИАЛОВ ЛЕКЦИЙ И СЕМИНАРОВ

Временный канал для информирования

http://t.me/+l0gwLqzWepsxYWVi

Войти можно только с учёткой вышки

@edu.hse.ru

ИНФРАСТРУКТУРА

.net 6.0

Установить на личный компьютер интегрированную среду разработки VS Code по инструкциям. При выборе дополнительных задач установщика – выбрать ВСЕ!

ССЫЛКА НА ВИДЕО https://youtu.be/tC6nTO6zBfQ?si=VkBC8ASFZ1l-fpxR

Дополнительно установить из запущенного VS Code

- C# (https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp): позволяет подсвечивать синтаксис, отлаживать и запускать код; для корректной работы проверить установку .Net

Полезные расширения для работы с кодом и тестами:

- C# namespace autocompletion (https://marketplace.visualstudio.com/items?itemName=adrianwilczynski.namespace

- Auto-using for C# (https://marketplace.visualstudio.com/items?itemName=Fudge.auto-using)

- .Net Core Test Explorer (https://marketplace.visualstudio.com/items?itemName=formulahendry.dotnet-test-explorer)

ЧТО ПОЧИТАТЬ?

- Г. Шилдт C# Полное руководство (ищем самое свежее издание)
- Д. Абахари C# 9.0 Справочник. Полное описание языка (8.0 тоже подойдёт, подлинник называется C# 9.0 in a Nutshel. The Definitive Reference)
- Richter, J. CLR via C# (подлинник предпочтительнее, в переводе есть ошибки)
- Документация по C# (https://learn.microsoft.com/ru-ru/dotnet/csharp/)
- Материалы лекций (SmartLMS) и ссылки к ним

ПЕРЕХОДИМ К ПРОГРАММИРОВАНИЮ

- Программирование часто контринтуитивно
  - При **обучении программированию** многие **концепции**, для которых мозг автоматом подбирает аналогии, **никак не связаны** друг с другом
  - Аналогия часто не просто лжива – она **антисистемна**!
- Очень помогает навык рационального мышления
  - LessWrong на русском (http://lesswrong.ru)
    - Особенно это помогает при тестировании ☺
  - 97 вещей, которые должен знать каждый программист (http://copist.ru/books/97things-dev)

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

Парадигмы

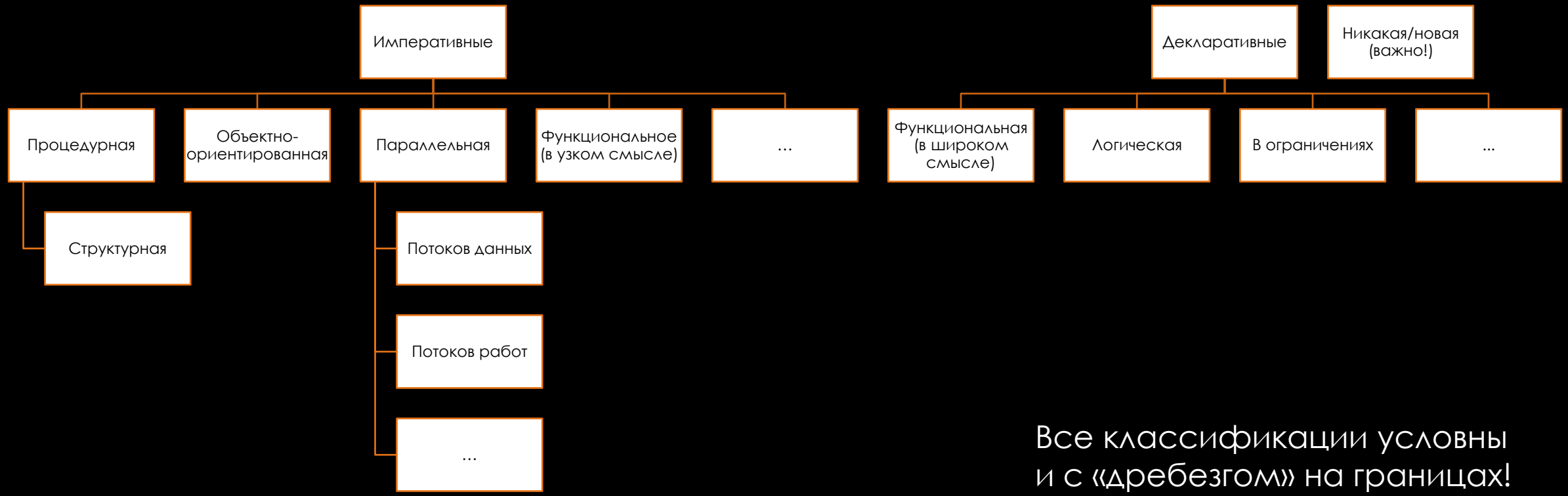
Парадигмы в C#

ИМПЕРАТИВНОСТЬ И ДЕКЛАРАТИВНОСТЬ

- **Императивность**
  - Что делаем - «чётко описанная последовательность элементарных действий»
- **Декларативность**
  - Что хотим получить – чёткое описание результата, достаточное для генерации чёткой последовательности действий

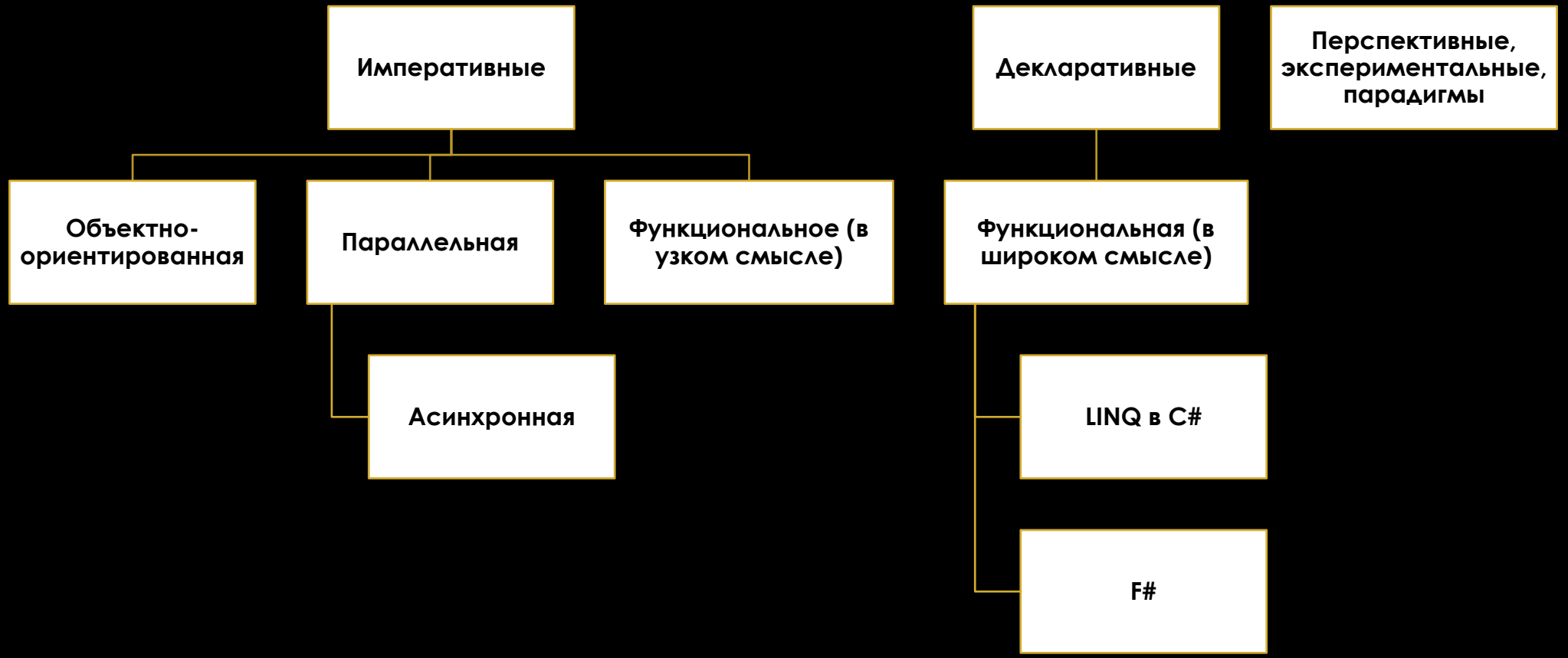
- **Императивное программирование** [imperative programming] описывает алгоритм в явном виде, позволяя задать последовательность изменений состояний исполнителя алгоритма во времени
- **Декларативное программирование** [declarative programming](от лат. Declaratio – объявление) – программирование, не требующее описания процесса получения результата, а требующее лишь описать сам результат, хотя бы и достаточно формально, чтобы компьютер смог выполнить процесс его получения

ПАРАДИГМЫ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ



Все классификации условны
и с «дребезгом» на границах!

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ C#



ВОЗМОЖНОСТИ РАЗРАБОТКИ НА C#

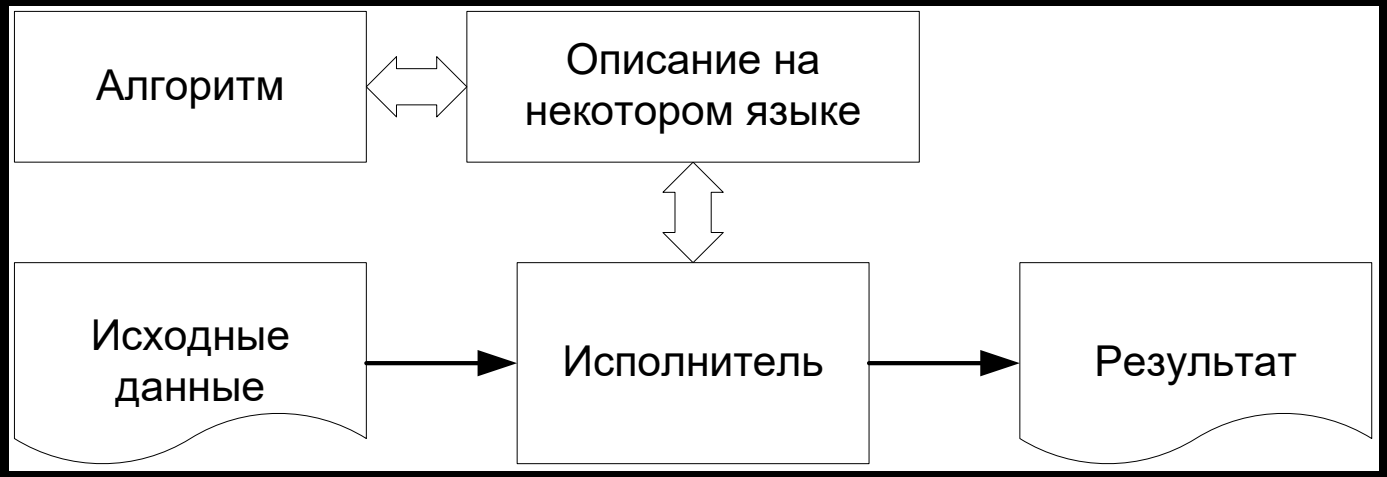
- Настольные приложения (Windows Forms/WPF);
- Мобильная разработка на Xamarin для iOS, Android;
- Web-приложения с использованием ASP.NET Core;
- Сервисы (WCF) и распределённые приложения;
- Компьютерные 2D/3D игры с использованием движка Unity.
- Механизм ADO .NET (ActiveX Data Objects .NET) для работы с базами данных.
- ADO.NET Entity Framework;
- LINQ (Language Integrated Query).

БАЗОВЫЕ ПОНЯТИЯ

- ✓ Алгоритм и его исполнитель, модель вычислений
- ✓ Программа и исходный код
- ✓ Язык программирования

АЛГОРИТМ

- **Алгоритм** [*algorithm*] – чётко описанная последовательность элементарных действий исполнителя над исходными данными для достижения предварительно сформулированного результата
- В программировании:
  - **Исполнитель** [*executor, actor*] – средства вычислительной техники (компьютер)
  - **Данные** [*data*] – любая информация, представленная в форме, пригодной для хранения, передачи и обработки средствами вычислительной техники



О ПРОБЛЕМЕ ФОРМАЛИЗАЦИИ АЛГОРИТМА

- **Вопрос «что такое алгоритм?» очень глубок:**
  - Что мы хотим от формального определения алгоритма
  - Сравнение различных исполнителей алгоритмов и их классификация
  - Модели вычислений и аксиоматический подход
  - Формализация входа и выхода алгоритма
  - Связь формализации алгоритма с теорией групп и теорией категорий
  - Конечность алгоритма и алгоритмическая разрешимость
  - Массовость алгоритмов
  - Понятие корректности алгоритма
  - Параллельные, распределённые, квантовые алгоритмы
- *Yuri Gurevich – Introduction to Algorithms and Computational Complexity, 2 of 3:* ***What Is An Algorithm****?* (http://channel9.msdn.com/Shows/Going+Deep/C9-Lectures-Yuri-Gurevich-Introduction-to-Algorithms-and-Computational-Complexity-2-of-n)

О МОДЕЛИ ВЫЧИСЛЕНИЙ – ОПРЕДЕЛЕНИЕ

- **Модель вычислений** [*model of computation*] – формальное абстрактное описание исполнителя алгоритма
  - Математическая модель, позволяющая исследовать, делать предсказания и проводить верификацию работы исполнителя
  - Один из основных объектов изучения теоретической информатики [*theoretical computer science*]

- *Wikipedia Category: Models of computation* (http://en.wikipedia.org/wiki/Category:Models_of_computation)
  - Оцените разнообразие!

ПРИМЕРЫ МОДЕЛЕЙ ВЫЧИСЛЕНИЙ

- Машина Тьюринга
- Алгорифмы Маркова
- Продукции Поста
- Конечный автомат с магазинной памятью
- **Равнодоступная адресная машина** [*Random-Access Machine (Memory Design) – RAM*]
- …
- Модель взаимодействующих последовательных процессов
- **Параллельная равнодоступная адресная машина** [*Parallel Random Access Machines – PRAM*]
- …
- Вероятностные модели в ассортименте
- Квантовый компьютер в нескольких вариантах

ПРОГРАММА

- **Программа** [*program*] – формальная запись алгоритма и обрабатываемых им *данных* на некотором *формальном языке* для некоторого класса *исполнителей*
  - Глубокое понимание программирования как создания программы требует понимания *теории формальных языков и теории трансляции*
  - Но «на пальцах» можно объяснить это через рассмотрение различных представлений «**исходного кода**» на языках программирования различного уровня – от Ассемблера до «СиПлюсПлюса» (*CPP*) и «ЯваСкрипта» (*JS*)
- **Исходный код** [*source code*] – текст компьютерной программы на языке программирования, который может быть *прочтён человеком*

ЯЗЫК ПРОГРАММИРОВАНИЯ

- **Язык программирования** (алгоритмический язык) [*programming language*] – формальная знаковая система, служащая для формального описания и реализации на компьютере алгоритмов обработки данных
- **Машинный язык** (машинный код) [*machine language/code*] – запись команд процессора непосредственно в том виде, в котором они поступают на ему для выполнения
  - ☑ Для современных цифровых компьютеров общего назначения осуществляется в **двоичном (бинарном) коде**
- **Язык ассемблера** [*assembler language*] – это символьная форма машинного языка с рядом возможностей, характерных для языка высокого уровня
  - С помощью языка ассемблера, как и с помощью машинного языка, программист получает доступ ко всем ресурсам компьютера

ЯЗЫК ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

- **Язык программирования высокого уровня** (ЯПВУ) [*high-level language*] – 1. Язык программирования, понятия и структура которого удобны для восприятия человеком (согласно ГОСТ 19781-90)
  2. Аппаратно-независимый язык программирования

- Все ЯПВУ являются **проблемно-ориентированными** (в той или иной мере), то есть более подходящими для создания программ для определённой предметной области
  - Те из языков, которые не имеют явной привязки к предметной области, принято называть **универсальными**

ЛЕКСЕМЫ C#

- **Алфавит** [character set] языка программирования – набор символов, используемый для построения текста программы

символы латинского алфавита	подчёркивание _
цифры	специальные символы

АЛФАВИТ C#

- **Лексема** [lexeme] – элементарная синтаксическая единица текста программы, то есть минимальная значимая последовательность символов алфавита. Описание лексем обычно даётся отдельно от синтаксического описания

идентификаторы

ключевые слова

знаки операций

литералы

разделители

КЛЮЧЕВЫЕ (СЛУЖЕБНЫЕ) СЛОВА C#

Ключевое слово [reserved word] – лексема языка программирования, имеющая предопределенный смысл для транслятора.
Ключевое слово никогда нельзя использовать в качестве идентификатора

if public int class interface

Полный перечень ключевых слов C#

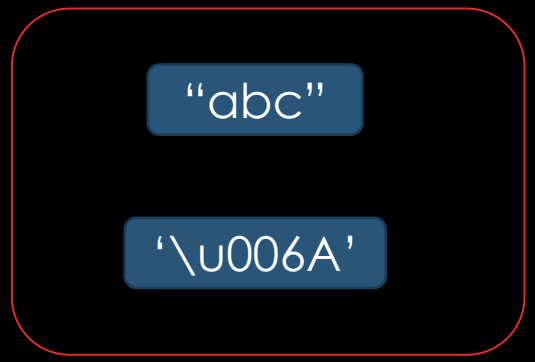
C# Keywords [https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/]

Метка [label] – необязательный специальный идентификатор оператора программы, служащий для указания места безусловной передачи управления оператором безусловного перехода goto (синтаксис может меняться)

goto Имя метки

ЕЩЁ НЕМНОГО ТЕРМИНОВ

- Литерал [literal] используется для записи непосредственного значения, например, чисел (числовые литералы – нумералы) и текстовых строк (символьные литералы)



Литералы C#

Нумералы C#

Мы будем следовать переводу документации Microsoft для C#, где не разделены литералы и нумералы

ПЕРВАЯ ПРОГРАММА НА C#

Структура программы

СТРУКТУРА ПРОСТОЙ ПРОГРАММЫ

```
// using-директивы
using System;

class Program // Класс приложения
{
    static void Main() // Метод Main - точка входа
    {
        // Код решения задачи
    } // Main()
} // class Program
```

C# - объектно-ориентированный язык! Любая программа явно (или неявно, начиная с C# 9.0) содержит хотя бы 1 тип данных – в данном случае, class Program

Руководство по языку C#. Основы. Общая структура программы [http://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/program-structure/?source=recommendations]

УПРОЩЕНИЕ НАПИСАНИЯ ПРОГРАММ – C# 9.0

```
using System;

// Работает с C# 9.0! Класс Program
// генерируется неявно.
Console.WriteLine("Hello world!");
```

Начиная с C# 9.0, написание коротких программ было значительно упрощено за счёт top-level statements

В данном случае мы не отходим от объектно-ориентированной парадигмы! Класс программы, метод Main и (опционально) параметр args генерируются неявно.

РОЛЬ ДЕКЛАРАЦИИ USING

```
System.Console.WriteLine("Для нажмите выхода ENTER.");
System.Console.ReadLine();
```

```
using System;

Console.WriteLine("Для нажмите выхода ENTER.");
Console.ReadLine();
```

МЕТОД MAIN()

```
static int Main()
{
    Операторы
}
```

```
static int Main(string[] args)
{
    Операторы
}
```

```
static void Main()
{
    Операторы
}
```

- **Main()** обязательно должен быть помечен модификатором **static**
- Результатом работы **Main()** может быть только тип: **void** , **int**, **Task** или **Task<int>** (последние два варианта допустимы при работе с асинхронными сценариями)
- Можно указать **string[ ]** – единственный допустимый параметр для **Main()**, т. е. набор аргументов, получаемых из командной строки

БЛОК, ОПЕРАТОР И ОПЕРАЦИЯ В C#

```
{ // Блок – набор операторов, выполняемых последовательно.
    int group = 5; // оператор – выражение из нескольких инструкций.
    System.Console.WriteLine("Номер группы: " + group);
}
```

```
int num1 = 10;
int num2 = 20;
int num3 = 30;
// Несколько операций «+» в рамках одного оператора.
System.Console.WriteLine(Сумма: " + (num1 + num2 + num3));
```

Попробуйте убрать круглые скобки

ОФОРМЛЕНИЕ КОДА

Комментирование

Отступы

ОФОРМЛЕНИЕ КОДА

- Отступы
- Комментарии

Комментарий – часть исходного кода, которая не анализируется транслятором

Виды комментариев

- Указания инструментальным средствам контроля версий, автоматического документирования и проч.
- Маркеры, помечающие места в исходном коде, на которые в дальнейшем надо обратить внимание
- Повторение кода 👎
- Объяснение кода
- Резюме кода
- Описание целей кода

ОФОРМЛЕНИЕ КОДА C#

```
using System;

class Program {
    static void Main(string[] args) {
        int inputData = 144; // присваивание значения 144 в inputData

        // не забыть добавить сюда ввод с клавиатуры

        double sqrtFromInputData = Math.Sqrt(inputData); // корень из
                                                         // inputData
    }
}
```

Комментарии могут быть полезными и бессмысленными

Как исправить?

Код примера (https://repl.it/@Maksimenkova/lec0102)

КОММЕНТАРИИ В КОДЕ C#

Однострочные:

```
// Это однострочный комментарий
```

Многострочные с двумя ограничителями:

```
int b; /* начало
Комментарий… мемуары…
конец */
```

Документирующие:

```
/// <summary>
/// Этот текст будет содержимым
/// элемента XML-документа
/// </summary>
```

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- Материалы лекций по C# Подбельского В.В., Дударева В.А
- Материалы лекции *Неотрефлексированный сдвиг парадигмы: от поколений языков программирования высокого уровня к метапрограммированию и высокопродуктивным DSL* Незнанова А.А.
- Себеста Р. Основные концепции языков программирования. – М. : Вильямс, 2001. – 672 с.
- Макконнелл С. Совершенный код. Практическое руководство по разработке программного обеспечения. – Спб. : Питер, 2005. – 896 с.
- Хант Э., Томас Д. Программист-прагматик. – Лори, 2004. – 270 с.
- Фаулер М. Рефакторинг. Улучшение существующего кода. – М.: Символ-Плюс, 2005. – 432 с.
- *Savage J.E. Models of Computation: Exploring the Power of Computing. – Addison-Wesley, 1998. – 672 p.*
  - http://cs.brown.edu/people/jsavage/book/pdfs/ModelsOfComputation.pdf

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- *Dharani R. Web API Design: Crafting Interfaces that Developers Love. 2017* (http://www.amazon.com/Web-API-Design-Interfaces-Developers/dp/1973436248)
- *Martin Fowler articles tagged by: API design* (http://martinfowler.com/tags/API%20design.html)