Разработка .NET приложений

Трансляция

QR Code

Подключиться в браузере по ссылке:
https://jazz.sber.ru

Код конференции: ***
Пароль: *****

Шаталов Юрий

Знакомство

- О себе
  - Шаталов Юрий
  - Закончил отделение второго высшего на ВМК
  - Разработчик в крупной российской IT компании
  - Microsoft Certified Professional Developer (MCPD)
  - Курс читается несколько лет.
    - Изменяется от года к году
    - Рассказ о современном положении дел
  - E-mail
- Предложение
  - Давайте общаться на ты и по именам

Знакомство

Заполните, пожалуйста, файлик

- О Вас
  - Представьтесь

    QR Code
  - Каково ваше отношение к IT?
    - работаете в IT, собираетесь, интересуетесь
  - Что знаете / слышали о

    Telegram
    - .NET, .NET Framework, .NET Core
    - C# / C++ / C / Java ?

    QR Code
  - Что ожидаете от курса ?

Программа курса

1 семестр

- Основы программирования на языке C# 12
  в среде .NET 8 (9) / .NET Framework 4.8.1

2 семестр

- Доступ к данным и манипуляция данными
  - ADO.NET, LINQ, Entity Framework / Entity Framework Core
- Backend. Разработка распределенных приложений / сервисов
  - ASP.NET Core, gRPC, Web API
- Разработка .NET приложений с пользовательским интерфейсом
  - .NET MAUI, Windows Presentation Foundation (WPF)
- Разработка Web-приложений на платформе .NET

Отчетность

- В конце каждого семестра
  - Экзамен
  - Зачет
- Для зачета необходимо сделать ВСЕ домашние задания
- Зачет и Экзамен независимые* и необязательные**

Литература

Основы C# и .NET

- Э. Троелсен, Ф. Джепикс. Язык программирования C# 9 и платформа .NET 5: основные принципы и практики программирования (2022)
- Марк Прайс. C# 10 и NET 6. Современная кроссплатформенная разработка (2023)
- Джозеф Албахари: C# 9.0. Справочник. Полное описание языка (2021)
- Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# (2013/2017/2023)
- Старайтесь читать литературу по .NET в оригинале (на английском языке)
- Help https://docs.microsoft.com/ru-ru/dotnet/
- Исходники .NET, .NET Core https://github.com/dotnet/core
- Исходники .NET Framework http://referencesource.microsoft.com/

Разное

- Софт
  - Visual Studio 2022 – https://visualstudio.microsoft.com/ru/vs/community/. Достаточно версии Community
  - Опционально – Resharper – https://www.jetbrains.com/ru-ru/resharper/ – помогает быстрей и качественней разрабатывать
  - Альтернатива

    Rider – https://www.jetbrains.com/ru-ru/rider/
- Сайт
  - http://msudotnet.ru/
- Почта
  - e-mail

Разновидности .NET

- .NET Framework
  - ~~Наиболее полная версия .NET~~
  - Кроссплатформенная для Windows
  - Входит в дистрибутив и обновления Windows
  - C# 7.3
  - Актуальная и последняя версия .NET Framework 4.8.1
- .NET Core
  - Платформа разработки с открытым кодом
  - Поддерживается Майкрософт и сообществом .NET на сайте GitHub
  - Кроссплатформенная: Windows, MacOS и Linux
  - Активно развивается
  - C# 8
  - Актуальная версия .NET Core 3.1
- .NET Standard
  - Поддерживается всем платформами .NET (.NET Framework, .NET Core)
  - Открытый стандарт
  - Актуальная версия .NET Standard 2.1 (2.1 уже не поддерживается .NET Framework)
- Xamarin
  - Позволяет единым образом создавать приложения для Android, iOS, Mac
- Mono
  - Урезанная версия NET для Unix и Mac
  - Open source
  - Не Майкрософт (Sponsored by Microsoft)
- .NET 5
  - Следующая версия .NET Core (.NET 5 = .NET Core vNext)
  - C# 9+
  - Платформа разработки с открытым кодом
  - Поддерживается Майкрософт и сообществом .NET на сайте GitHub
  - Заменяет .NET Framework, .NET Core, Mono

  - .NET 6 – Ноябрь 2021 – C# 10 – LTS (long time support)
  - .NET 7 – Ноябрь 2022 – C# 11
  - .NET 8 – Ноябрь 2023 – C# 12 – LTS (long time support)
  - .NET 9 – Ноябрь 2024 – C# 13

  - Релизы новых версий по ноябрям
  - LTS версия раз в 2 года

Сегодня

- Предыстория
- Понятие платформы .NET
- Первая программа на C#
- Основные типы

Предыстория.

- Предпосылки появления .NET Framework. Начало 2000-ых
- Язык C. Использование Win API
  - Процедурный стиль
  - Не объектный
  - Сложный API
- Язык C++. Использование различных оберток над Win API
  - Сложная работа с указателями
  - Ручное управление памятью
  - Переносимость, оптимизация
- Visual Basic 6
  - Нет классического наследования
  - Трудности с многопоточностью
- Java
  - Подразумевает использования только языка Java
- COM
  - Нет наследования
  - Не гарантии совместимость типов
  - Необходимость регистрации в реестре
  - Невозможность использования разных версий одного и того же модуля

Что дает .NET

- Кроссплатформенность
  - .NET Framework только разные версии Windows
  - Mono – Mac OS и частично клоны Unix http://www.mono-project.com
  - .NET Core (Core CLR). Open Source https://github.com/dotnet/coreclr
  - Xamarin – Android, iOS, Mac
  - .NET 5+ – Windows, Linux, macOS, iOS, Android, tvOS, watchOS, WebAssembly и другие
- Поддержка нескольких языков программирования
  - C#, VB.NET, JScript.NET, Managed C++ (C++/CLI), F#
  - https://en.wikipedia.org/wiki/List_of_CLI_languages
- Общая среда выполнения для различных языков программирования
  - Прозрачное межязыковое взаимодействие
- Библиотека базовых классов
  - Web, Mobile, Desktop, Services, Data Access, Cloud, …
- Упрощенная работа с памятью
- Простое развертывание
- Безопасность
- Взаимодействие со старым кодом (native, COM и т.д.)
- Side-by-side установка программ, использующих разные версии .NET

Версии .NET Framework

Версия .NET	Год выхода	По умолчанию в версиях Windows	Нововведения	Visual Studio	Версия C#
.NET Framework 1.0	2002			Visual Studio .NET	C# 1
.NET Framework 1.1	2003	Windows Server 2003		Visual Studio .NET 2003	C# 1.2
.NET Framework 2.0	2005		Обобщенные типы, анонимные методы, x64 и IA-64.	Visual Studio 2005	C# 2
.NET Framework 3.0	2006	Windows Vista, Windows Server 2008	WPF, WCF, WF	Visual Studio 2005 + расширения	C# 3
.NET Framework 3.5	2007	Windows 7, Windows Server 2008 R2	LINQ, ADO.NET Entity Framework, ASP.NET AJAX	Visual Studio 2008	
.NET Framework 4.0	2010		Dynamic, PLINQ, F#, Windows Azure Apps	Visual Studio 2010	C# 4
.NET Framework 4.5, 4.5.1, 4.5.2	2012, 2013, 2014	Windows 8 (8.1), Windows Server 2012 (R2)	Windows Store Apps (Windows 8, 8.1)	Visual Studio 2012, 2013	C# 5
.NET Framework 4.6, 4.6.1, 4.6.2	2015, 2016	Windows 10, Windows Server 2016	RyuJIT, UWP, .NET Native, поддержка экранов с высоким DPI, расширение криптографии	Visual Studio 2015	C# 6
.NET Framework 4.7, 4.7.1, 4.7.2	2017, 2018	Windows 10 RS2-RS4	High DPI for WinForms, TLS 1.2 support, DI ASP.NET	Visual Studio 2017 (с обновлениями)	C# 7
.NET Framework 4.8	2019	Windows 10 R6 (1903)	Обновленный JIT от .NET Core, минорные изменения	Visual Studio 2017	C# 7.3 (09.18)
.NET Framework 4.8.1	08.2022	Windows 10 Version 20H2	Расширение криптографии, улучшение поддержки разных DPI	Visual Studio 2022 (17.3)	**C# 7.3**

Версии .NET Core

Версия	Релиз	Поддержка	Окончание поддержки
.NET Core 1.0	Январь 2016	Поддержка окончена	Июнь 2019
.NET Core 1.1	Ноябрь 2016	Поддержка окончена	Июнь 2019
.NET Core 2.0	Август 2017	Поддержка окончена	октябрь 2018
.NET Core 2.1	Май 2018	**Длительная поддержка (3 года)**	21 августа 2021
.NET Core 2.2	Декабрь 2018	Поддержка окончена	декабрь 2019
.NET Core 3.0	Сентябрь 2019	Поддержка окончена	Март 2020
.NET Core 3.1	Декабрь 2019	**Длительная поддержка (3 года)**	3 декабря 2022
.NET 5	Ноябрь 2020	Поддержка окончена	
.NET 6	Ноябрь 2021	**Длительная поддержка (3 года)**	
.NET 7	Ноябрь 2022	Поддержка окончена	
.NET 8	**Ноябрь 2023**	**Длительная поддержка (3 года)**	
.NET 9	**Запланирован на ноябрь 2024**	**Короткая поддержка**	

Версии .NET / .NET Core

.NET Schedule

- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

https://devblogs.microsoft.com/dotnet/introducing-net-5/

Сегодня

- Предыстория
- **Понятие .NET**
- Первая программа на C#
- Основные типы

Платформа .NET

- **Общеязыковая среда выполнения**
  Common Language Runtime – CLR

- Единая система типов
  Common Type Specification – CTS

- Общеязыковая спецификация
  Common Language Specification – CLS

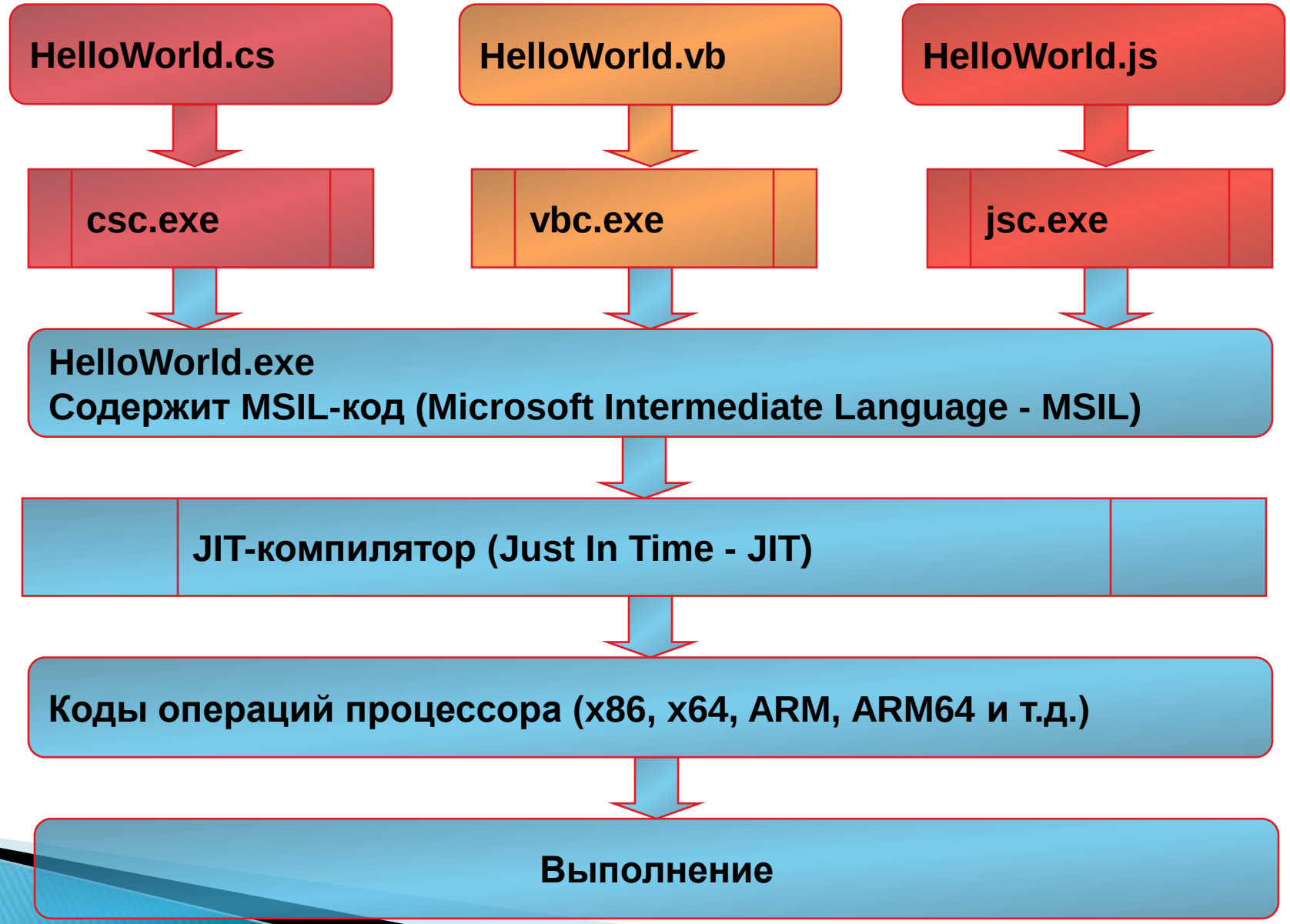
- Библиотека базовых классов
  Base Class Library – BCL

Общеязыковая среда выполнения

- **Common Language Runtime – CLR (Core CLR)**
- Виртуальная исполняющая среда
- Запускается при старте вашего приложения

- Отвечает за:
  - Загрузку сборок
  - Just In Time (Jit) компиляцию
  - Управление памятью
  - Управление безопасностью

MSIL–компиляция



JIT компиляция и CLR

- Программный код компилируется в промежуточный код (Intermediate Language – IL, MSIL, CIL)
- CLR – не интерпретатор. Компиляция происходит 1 раз. Повторно не компилируется, а используется уже откомпилированный код

- Более медленный старт и работа приложения

- Экономия памяти
  - Код на IL обычно занимает меньше места
  - Компилируется только тот код, который выполняется
- JIT компилятор получает высоко оптимизированный код (заточенный под конкретную аппаратную модель)
- CLR отслеживает частоту вызова и может производить оптимизацию налету

Общеязыковая среда выполнения

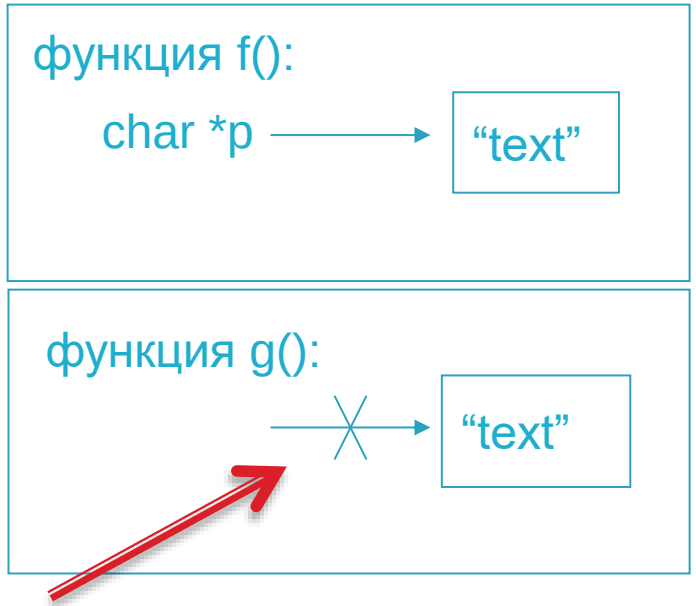
- Common Language Runtime – **CLR**
- Отвечает за:
  - Загрузку сборок
  - Just In Time компиляцию
  - **Управление памятью**
  - Управление безопасностью

Управление памятью

- Автоматическая сборка мусора

```
// Утечка памяти в C
char *f(int a)
{
     char *p = (char)malloc(…);
     …
     return p;
}
…
void g(){ f(1); }
```

Утечка памяти



Сборщик мусора (Garbage Collector - GC) отслеживает ссылки на объекты. Он обнаружит, что на область памяти p больше нет ссылок и освободит эту область.

- CLR может перенести часто используемые объекты для оптимизации доступа к страницам памяти

Общеязыковая среда выполнения

- Common Language Runtime – **CLR**
- Отвечает за:
  - Загрузку сборок
  - Just In Time компиляцию
  - Управление памятью
  - Управление безопасностью

Сборки

- Независимая единица кода
- Файл с расширением dll или exe *
- Состоит из:
  - **Манифеста**
    - Содержит информацию о текущей версии сборки, культуре, перечень ссылок на все внешние сборки, необходимые для работы сборки
  - **Метаданных типов**
    - Описание всех типов внутри сборки, их публичных членов
  - **Промежуточного кода (IL)**
  - **Ресурсов**
- Благодаря самоописанию, значительно упрощает повторное использование (не нужна сложная COM инфраструктура, регистрация в реестре, не нужны заголовочные файлы)
- Сборка может содержать цифровую подпись

Утилиты ILDASM, dotPeek, .NET Reflector

* Сборка может состоять из нескольких модулей (netmodule). Не поддерживается Visual Studio

Структура .NET сборок

Демонстрация

Общеязыковая среда выполнения

- CLR (Common Language Runtime)
  - Загрузка сборок
  - Just In Time компиляция
  - Управление памятью
  - Управление безопасностью

Платформа .NET

- Общеязыковая среда выполнения

- Единая система типов
  Common Type Specification – CTS
- Общеязыковая спецификация
  Common Language Specification – CLS

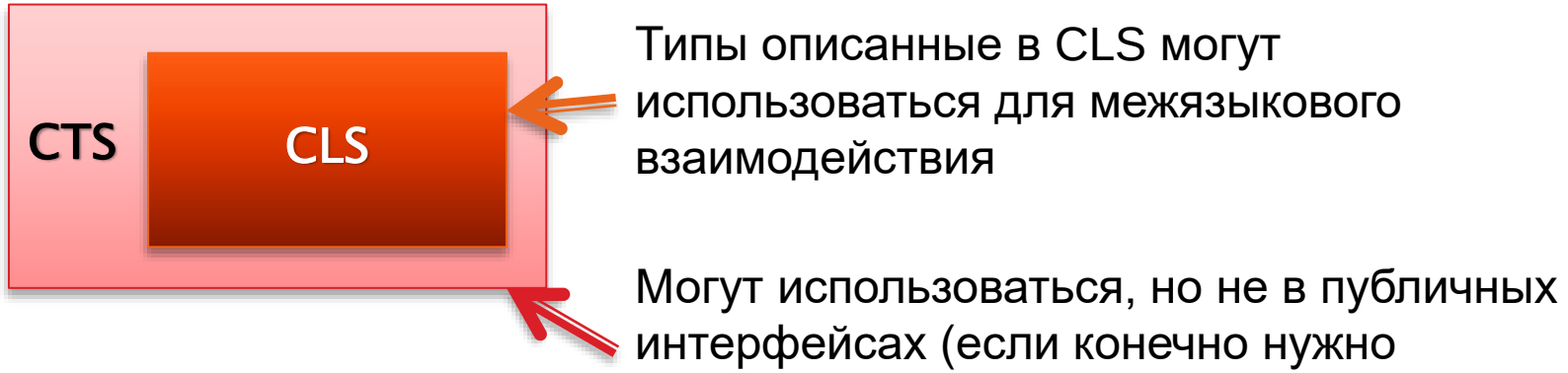
- Библиотека базовых классов
  Base Class Library – BCL

Межязыковое взаимодействие

Демонстрация

Единая система типов (CTS)

- Типы одинаковые на всех языках
- Поскольку в силу особенности языков не все языки могут поддерживать все типы (CTS) выделено подмножество типов. Это подмножество типов описано в Общеязыковой спецификации (CLS). Все типы в CLS обязаны поддерживаться всеми .NET языками.



Платформа .NET

- Общеязыковая среда выполнения
- Единая система типов
- Общеязыковая спецификация

- Библиотека базовых классов
  Base Class Library – BCL
  - Для .NET Framework – Global Assembly Cache – GAC

Библиотека базовых классов (BCL)

- Для .NET Framework
  - Расположена в Global Assembly Cache – GAC
  - c:\Windows\Microsoft.NET\assembly
  - Ранние версии – c:\Windows\assembly;
  - Может использоваться всем программами
  - Позволяет сохранять и использовать разные версии одной и той же сборки

- Для .NET Core
  - Расположена c:\Program Files\dotnet, c:\Program Files (x86)\dotnet
  - Может использоваться всем программами
  - Позволяет сохранять и использовать разные версии одной и той же сборки

  - Или поставляется вместе с приложением

  - mscorelib.dll – основная сборка. Используется во всех программах. Содержит пространство имен System.

Платформа .NET

- Общеязыковая среда выполнения
  Common Language Runtime – CLR
  - MSIL JIT-компилятор
  - Сборщик мусора
- Единая система типов
  Common Type Specification – CTS
- Общеязыковая спецификация
  Common Language Specification – CLS
- Библиотека базовых классов
  Base Class Library – BCL

Сегодня

- Предыстория
- Понятие платформы .NET
- **Первая программа на C#**
- Основные типы

Hello, World!

```
using System;

namespace Hello
{
  class HelloWorld
  {
    /// <summary> Entry point </summary>
    static void Main(string[] args)
    {
      Console.WriteLine("Hello, C# World!");

    }  // end of Main()

  }  // end of HelloWorld

}  // namespace
```

Первая программа на C#

Демонстрация

Замечания

- Пространство имен
  - объединяет группу семантически связанных между собой типов
  - Позволяет отделять типы с одинаковыми названиями
- Варианты метода Main
  - `static void Main(){…}`
  - `static int Main(){… return 0; }`
  - `static void Main(string[] args){…}`
  - `static int Main(string[] args){… return 0; }`
- ***using*** позволяет сократить полное название типа (System.Console). Как бы объединяет пространства имен или тип с текущим
- .NET использует Unicode.
  - Название типов можно заводить и на русском языке (но не рекомендуется)
- Языки для .NET чувствительны к регистру
  - Main() и main() разные методы
- Вывод на консоль: *System.Console.WriteLine("текст")*
- Чтение данных с консоли: *string s = System.Console.ReadLine()*

Замечания для .NET 5 (C# 9). Инструкции верхнего уровня

```
using System;
namespace HelloWorld
{
  class Program
  {
     static void Main(string[] args)
     {
          Console.WriteLine("Hello World!");
     }
   }
}
```

```
using System;
Console.WriteLine("Hello World!");
```

```
System.Console.WriteLine("Hello World!");
```

Удобны для обучения, экспериментов с кодом и очень простых программ

- Только один файл в приложении может использовать инструкции верхнего уровня
- Не может быть объявления точки входа (Main)
- Можно обращаться к массиву строк с именем args
- Как бы один файл содержит инструкции, которые обычно находятся в методе Main
- Нет ограничений на использование .NET функциональности (BCL и своего кода)

Сегодня

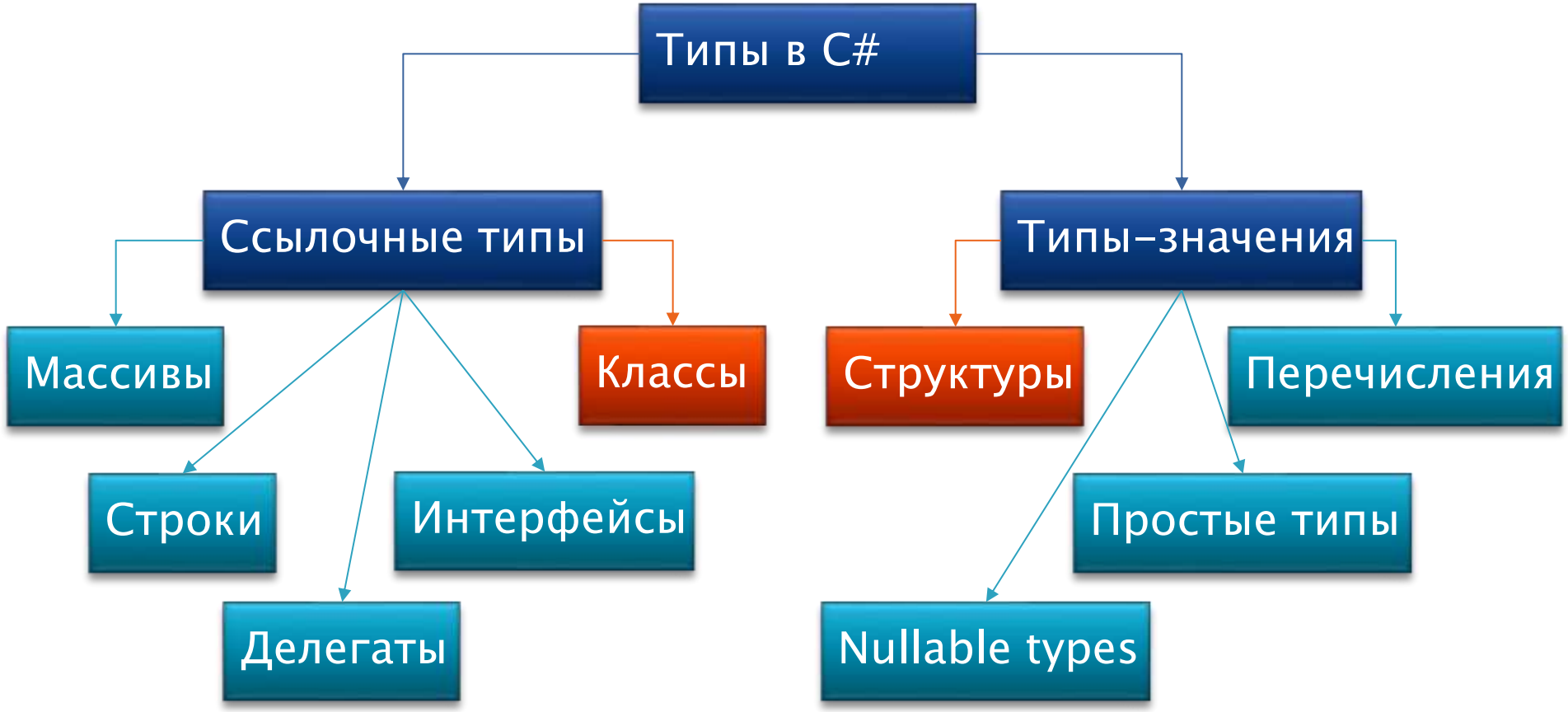
- Предыстория
- Понятие платформы .NET
- Первая программа на C#
- **Основные типы**

Строгая типизация в C#

- Каждая переменная и экземпляр объекта в системе относится к четко определенному типу !!!
- Все типы происходят от одного корневого предка – типа **object** *

** – точнее приводятся к типу object*

Типы в C#



* - условная схема, поскольку все ссылочные типы (кроме интерфейсов) – классы, все типы значения - структуры

Ссылочные типы и типы значения

Признак	Ссылочные типы	Типы значения
Переменная	Ссылка на объект	Сам объект
Память	Куча	Стек или куча
Присваивание	Копирование ссылки	Копирование объекта
Значение по умолчанию	null	0, 0.0, '\0', false

Ссылки и значения

Демонстрация

Самые важные типы

- **int** – 32-битное целое (System.Int32)
- **bool** – логический тип (System.Boolean). Значения только **true** и **false**
- **float**, **double** – вещественные типы (System.Single и System.Double)
- **char** – символьный тип Unicode
- **string** – строка текста (Unicode)
- **DateTime** – дата и время

Простые целые типы

Тип (C#)	Полное название типа	Диапазон	Описание	Размер (бит)
sbyte	System.Sbyte	-128 до 127	Знаковое целое	8
byte	System.Byte	0 до 255	Без знаковое целое	8
short	System.Int16	-32 768 до 32 767	Знаковое целое	16
ushort	System.UInt16	0 до 65 535	Без знаковое целое	16
int	System.Int32	-2 147 483 648 до 2 147 483 647	Знаковое целое	32
uint	System.UInt32	0 до 4 294 967 295	Без знаковое целое	32
long	System.Int64	$-9 * 10^{19}$ до $9 * 10^{19}07$	Знаковое целое	64
ulong	System.UInt64	0 до $18 * 10^{19}$	Без знаковое целое	64
char	System.Char	U+0000 до U+ffff	Символ в Unicode	16
nint C# 9	System.IntPtr	Зависит от архитектуры	Знаковое целое	Зависит от архитектуры
nuint .NET 5	System.UIntPtr	Зависит от архитектуры	Без знаковое целое	Зависит от архитектуры

* Все типы – типы значения

Вещественные типы

Тип (C#)	Полное название типа	Диапазон	Точность	Размер (бит)
float	System.Single	$\pm 1.5 * 10^{-45}$ до $\pm 3.4 * 10^{38}$	7 знаков	32
double	System.Double	$\pm 5.0 * 10^{-324}$ до $\pm 1.7 * 10^{-308}$	15-16 знаков	64
decimal **	System.Decimal	$(-7.9 \times 10^{28}$ до $7.9 \times 10^{28}) / (10^{0}$ до $^{28})$	28-29 знаков	128

** Не имеет аппаратной поддержки
Всегда проверяет диапазон

* Все типы – типы значения

Важные типы

- **bool** – логический тип
  - System.Boolean
  - Значения только **true** и **false**
  - Тип значение
- **string** – строка текста (Unicode)
  - System.String
  - неограниченной длины
  - Ссылочный тип
- **DateTime** – дата и время
  - Структура (тип-значение)
  - От 1 января 1 года до 31 декабря 9999 года
  - Точность 100 нс
  - Работает с временными зонами

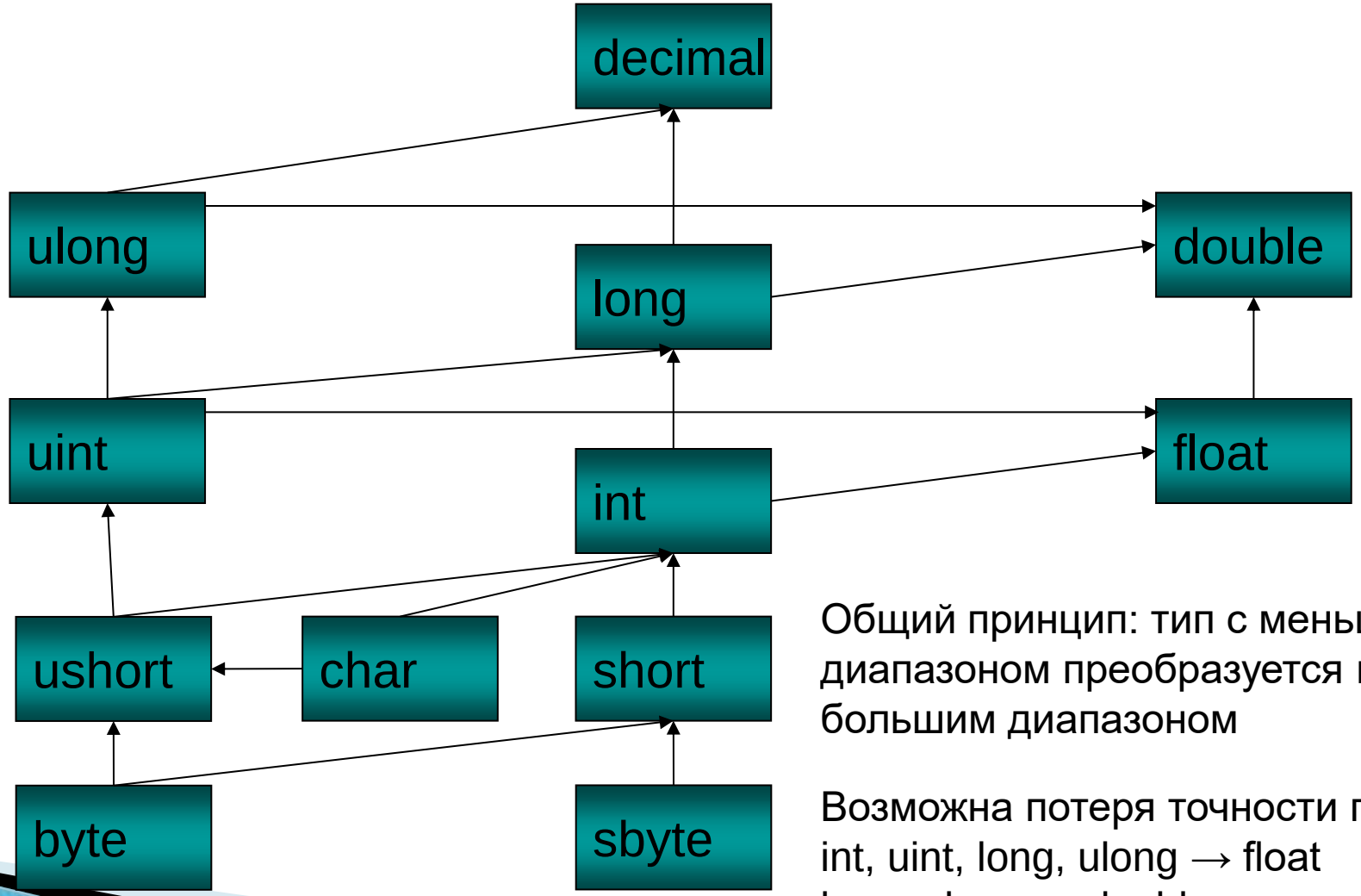
Типы данных по умолчанию

- Если значение целое и оно помещается в int – то подразумевается int
  - 5 – тип int Пример: int i= 45;
- Если значение вещественное – то подразумевается double
  - 5.6 – тип double Пример: double d= 12.277;
- Для обозначения конкретных типов служат “суффиксы”
  - 5l – long Пример: long l = 5l;
  - 5.4f – float Пример: float f = 5f;
  - 4m – decimal Пример: decimal d = 0m;
- Шестнадцатеричное число 0xЧИСЛО
  - 0x009F Пример: int i= 0x1234FFFF;
- Восьмеричное число 0ЧИСЛО
  - 06732 Пример: int i= 05777;
- Двоичное число 0bЧИСЛО. _ – разделитель разрядов
  - 0b1010, 0b1101_10101 Пример: int i= 0b0111_0000_1111_0101;

Вычисление типа выражения

- Тип выражения определяется в порядке приоритета:
  - Если в выражении присутствует decimal, то результат операции – decimal
  - Если присутствует вещественное число, то результат операции – double
  - ulong, если присутствует тип ulong
  - long, если присутствует тип long
  - Результат операции с целыми числами – int

Неявное приведение типов



Например:
short s = 5;
long l = s;

Общий принцип: тип с меньшим диапазоном преобразуется в тип с большим диапазоном

Возможна потеря точности при:
int, uint, long, ulong → float
long, ulong → double.

Явное приведение типов

- Синтаксис:
  - `(type)expression`
- Пример:

```
double d = 5.5;
int i = (int)d;
```

- Применяется при преобразованиях типов с возможной потерей точности
- При «зашкаливании» результат определяется контекстом

Контекст проверки вычисления

- 2 контекста
  - **checked** – проверяет на переполнение
  - **unchecked** – не проверяет

- Устанавливаются
  - Глобально (опции проекта)
  - Локально (блоки **checked** и **unchecked**)
  - Не распространяется на вызовы функций

- По умолчания проверка выключена.
  - Однако, если значение выражения может быть вычислено в процессе компиляции, то употребляется контекст checked
  - byte h = (byte) (255 + 100); // вызовет ошибку в процессе компиляции

Checked и unchecked

Демонстрация

Тип Перечисление – Enum

- Служит для кодирования возможных значений или магических чисел
- enum WeekDays {
  - Monday,
  - Thursday,
  - …
- }
- enum OneMoreEnum {
  - Monday = 2,
  - Thursday,
  - Среда = 4,
  - …}

```
enum Имя [:базовый целочисленный тип]
{
    Имя1 [=значение1]
    [, … ИмяN [=значениеN]]
}
```

- Объявление переменной и использование:
  - OneMoreEnum my = OneMoreEnum. Среда ;

- По умолчанию “наследуются” от int, но могут “наследоваться” от другого простого целочисленного типа
- Если не указано значение, то для первого по умолчанию – 0, для каждого последующего – предыдущее +1
- Возможно приведение типов: int I = (int)my; int j = (int)OneMoreEnum. Среда;

Массивы

- Содержат элементы только одного типа
- Длина
  - Задаётся при выделении
  - Изменить потом нельзя
- Бывают
  - Одномерные
  - Многомерные
- Допускается вложенность
- Нумерация элементов всегда с 0

- * В C# – всегда с 0, но .NET позволяет задавать массивы и с не нулевой нижней границей индекса

Одномерные массивы

- Объявление:
  - ***type_name*[] *var_name*** [= *init_expr*];
- Создание массива (обязательное задание длины массива)
  - `int[] arr1; // объявление переменной`
  - `arr1 = new int[5]; // создание массива из 5 элементов, выделение памяти`
  - `byte[] digits = new byte[10];`
- Индексация:
  - `int j = arr1[2];   // получение значения из массива`
  - `arr1[23] = 345;    // установка значения в массиве`
  - `int last = arr1[^1]; // Индекс от конца: оператор ^`
  - Компилятор добавляет квазистатическую проверку выхода из диапазона
  - Нумерация начинается с 0 (в C#)
- Получить длину массива
  - int len = arr1.Length;

Объявление и инициализация

- Одновременное объявление и инициализация:
  - `int[] arr1 = new int[3] { 10, 20, 30 };`
  - `int[] arr2 = new int[] { 10, 20, 30 }; // длинна вычисляется автоматически`
  - `int[] arr3 = new[] { 10, 20, 30 }; // тип массива определиться по левой части`
  - `int[] arr4 = { 10, 20, 30 };`
  - `int a = 17;`
  - `int[] arr5 = { 10, a, 30 };`

- C# 12 (.NET 8)
  - `int[] arr6 = [ 10, 20, 30 ];`
  - `int[] arr7 = [40, 50, 60];`
  - `int[] single = [.. arr6, .. arr7]; // 10, 20, 30, 40, 50, 60`

Многомерные массивы

- Объявление:
  - type_name[,…,] var_name [= init_expr]
- Объявление и создание массива
  - `int[,] matrix = new int[4, 7];`
  - `int[,,] matrix3d = new int[4, 2, 3];`
- Обращение к элементам массива
  - `int element = matrix[3, 5];`
  - `matrix[2, 1] = 34;`
- Получение длинны массива
  - Размер k-того измерения (k – 0,1,2,3…)

    `int dimension1 = matrix.GetLength(k);`
- Общее количество элементов в массиве

  `int dimension = matrix.Length;`

Инициализация массива при создании

- Объявление и создание массива
  - `int[,] matrix = new int[4, 7];`
  - `int[,,] matrix3d = new int[4, 2, 3];`

- Инициализация массива при создании

```
string[,] array2d = new string[3, 2] {
        { "one", "two" },
        { "three", "four" },
        { "five", "six" } };
```

- Можно создать массив без указания размера. Размер вычисляется по правой части выражения

```
int[,] matrix2d = { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
int[,] matrix2x2;
matrix2x2 = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
```

Массивы массивов (Jagged array)

- Объявление:
  - type_name[ ][ ]…[ ] var_name [= init_expr]

- Объявление и создание массива

```
int[][] matrix = new int[3][];
matrix[0] = new int[3];
matrix[1] = new int[7];
matrix[2] = new int[] { 11, 22 };
```

- Инициализация всего массива при создании. Размеры вычисляются автоматически

```
int[][] jaggedArray =
{
    new int[] {1,3,5,7,9},
    new int[] {0,2,4,6},
    new int[] {11,22}
};
```

- Обращение к элементам массива

```
int element = matrix[3][5];
matrix[2][1] = 34;
```

- Получение длинны массива (количество элементов (массивов) в массиве)

```
int dimension = matrix.Length;
```

Диапазоны

- Оператор ..

```
int[] numbers = new[] { 0, 10, 20, 30, 40, 50 };

int[] subset1 = numbers[1..4];  // 10, 20, 30 – диапазон с 1 по 4 элемент (правая граница исключается)
int[] subset2 = numbers[3..];   // 30, 40, 50 (с 3 элемента и до конца)
int[] subset3 = numbers[..2];   // 0, 10 (с начала и до 2 элемента)

int[] subset4 = numbers[..^2];  // 0, 10, 20, 30 (с начала и без 2 последних элементов)
int[] subset5 = numbers[^3..];  // 30, 40, 50 (последние 3 элемента)
int[] subset6 = numbers[^3..^1];// 30, 40 (отсчёт от конца массива)

int[] subset7 = numbers[1..^1]; // 10, 20, 30, 40
```

Массивы

Демонстрация

Сегодня

- Предыстория
- Понятие платформы .NET
- Первая программа на C#
- Основные типы