

Визуальное программи- рование

ЛЕКЦИЯ 1

Содержание лекции

01

Что такое .NET

03

NuGet

02

Project and Solution

04

Язык C#

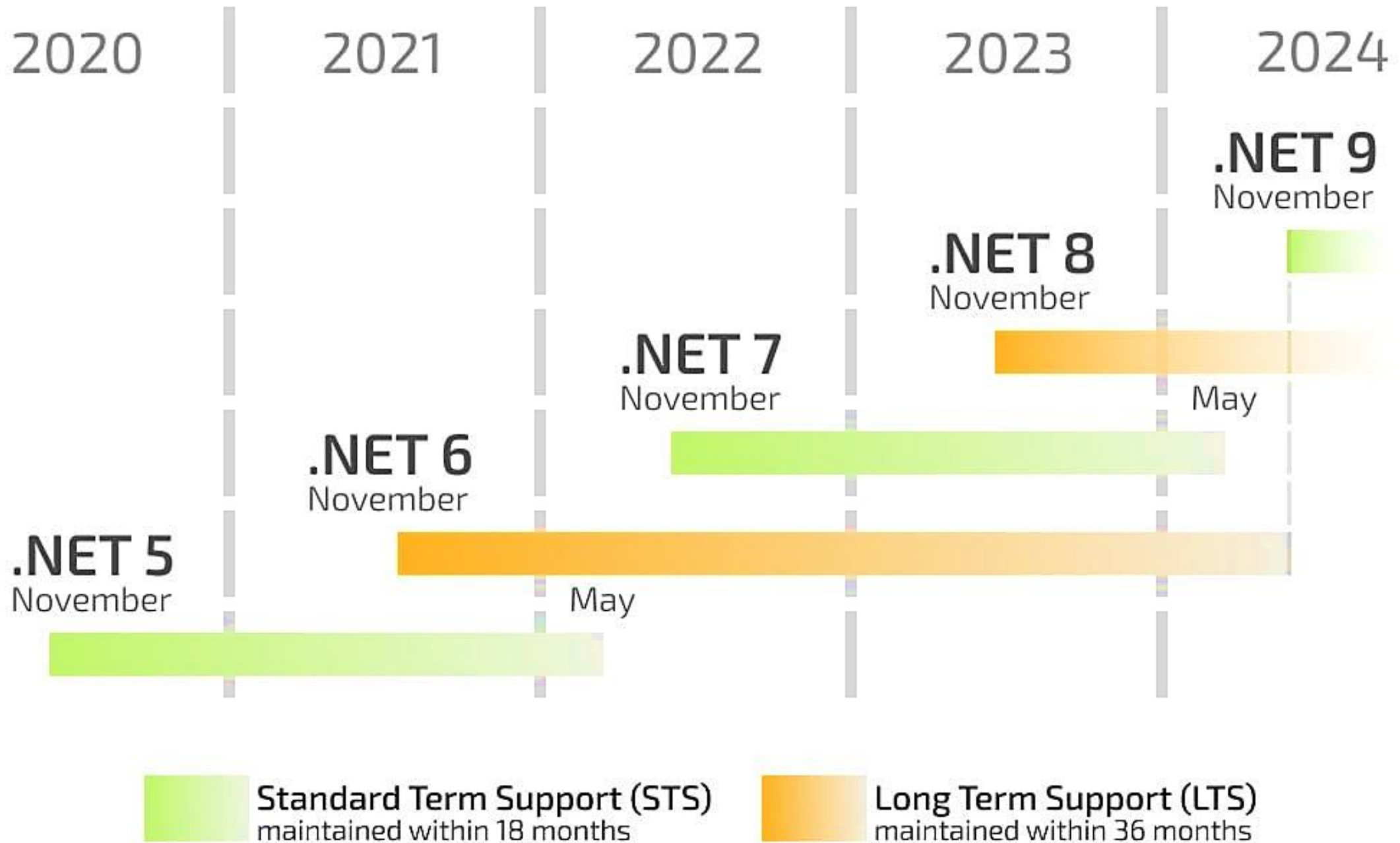
ЧТО ТАКОЕ .NET?

Что такое .NET

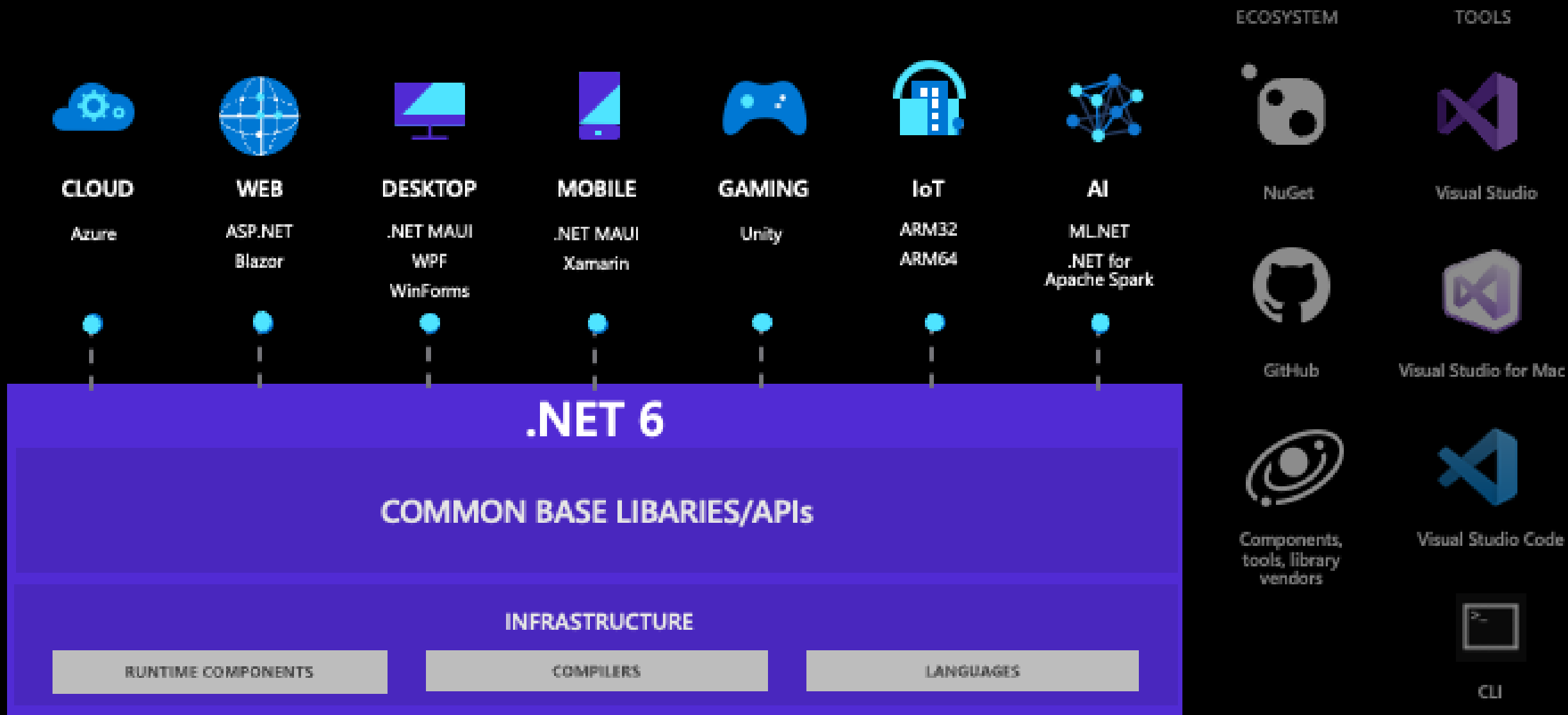
The logo consists of a solid purple rectangle. Inside the rectangle, the word "Microsoft" is written in a white, sans-serif font. Below it, ".NET" is written in a larger, bold, white, sans-serif font.

Microsoft
.NET

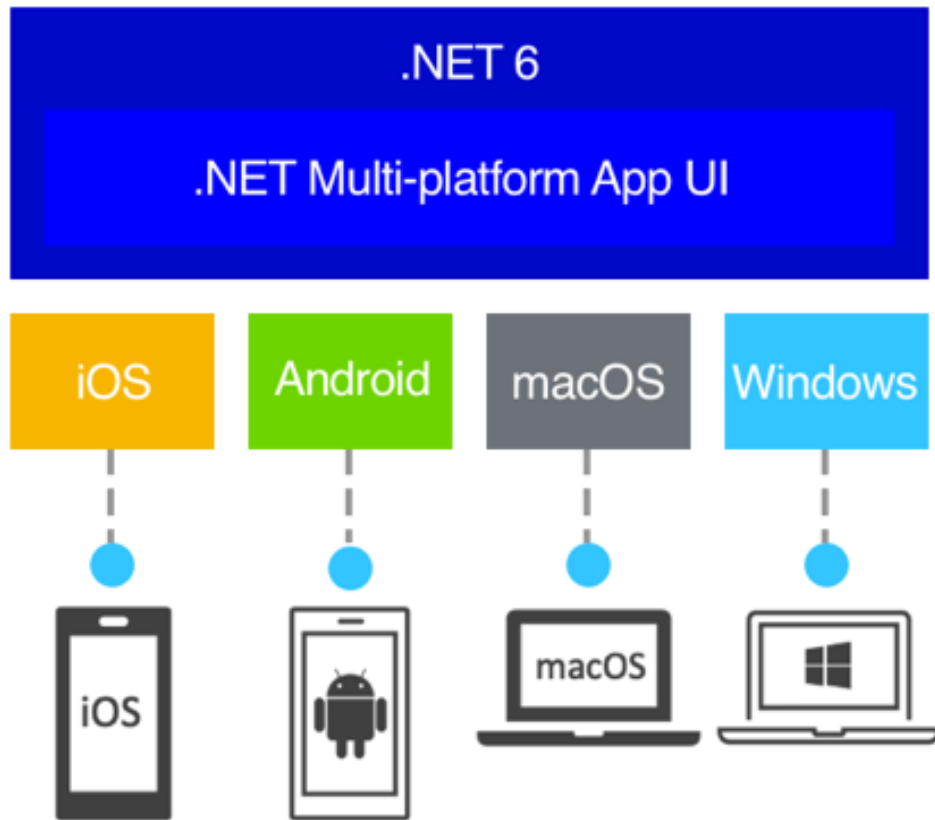
.NET - это бесплатная кроссплатформенная платформа для разработчиков с открытым исходным кодом для создания различных типов приложений.



.NET – A unified development platform



.NET MAUI



Это платформа для создания мобильных и десктопных приложений с помощью C# и XAML.

.NET MAUI объединяет интерфейсы API для Android, iOS, macOS и Windows в единый API, который позволяет разработчикам работать в любом месте, а также обеспечивает глубокий доступ к каждому аспекту каждой собственной платформы.

.NET Core



.NET Core - это кроссплатформенная реализация .NET для веб-сайтов, серверов и консольных приложений в Windows, Linux и macOS.

.NET Framework



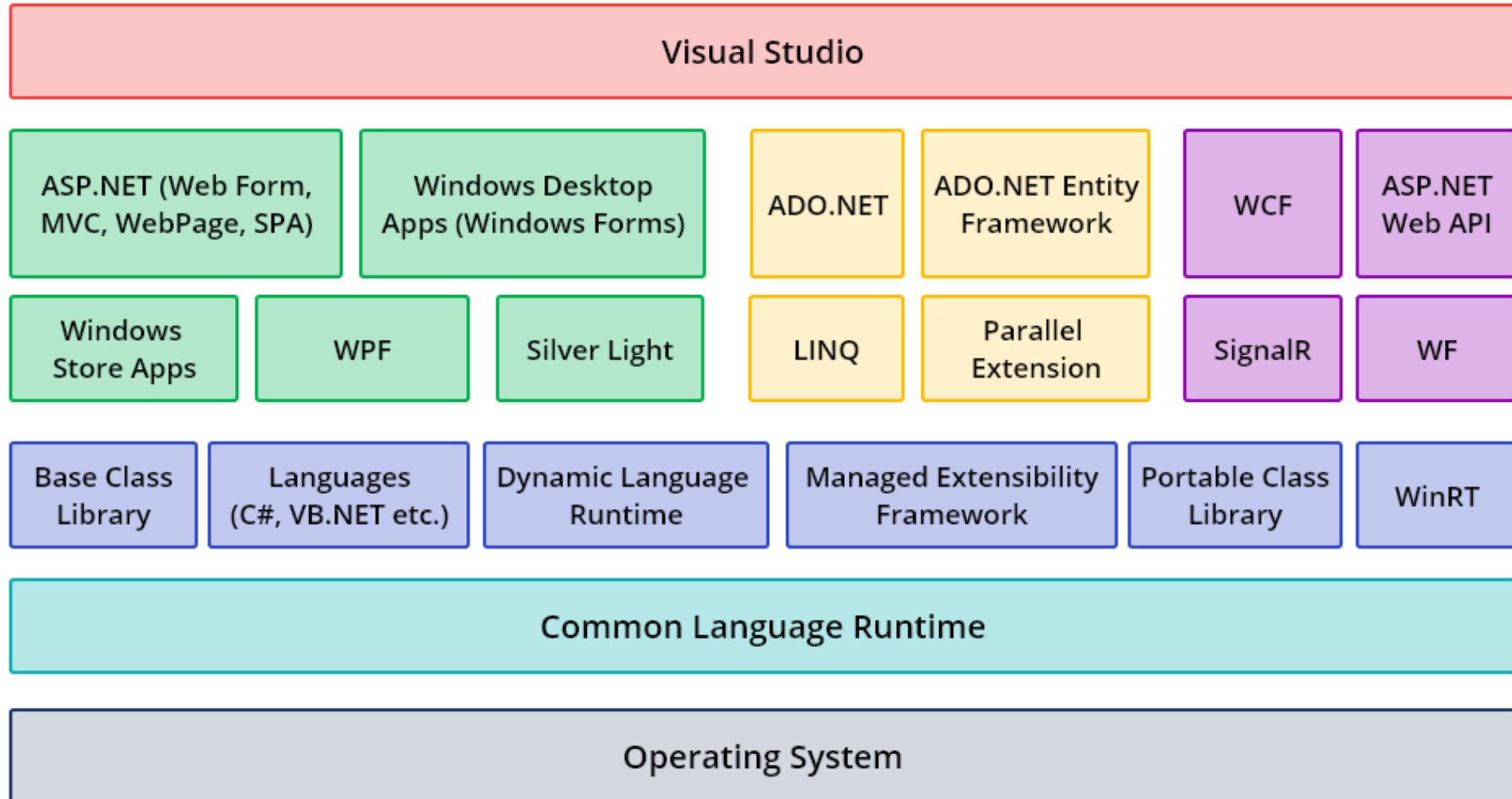
.NET Framework поддерживает веб-сайты, службы, десктопные приложения и многое другое в Windows.

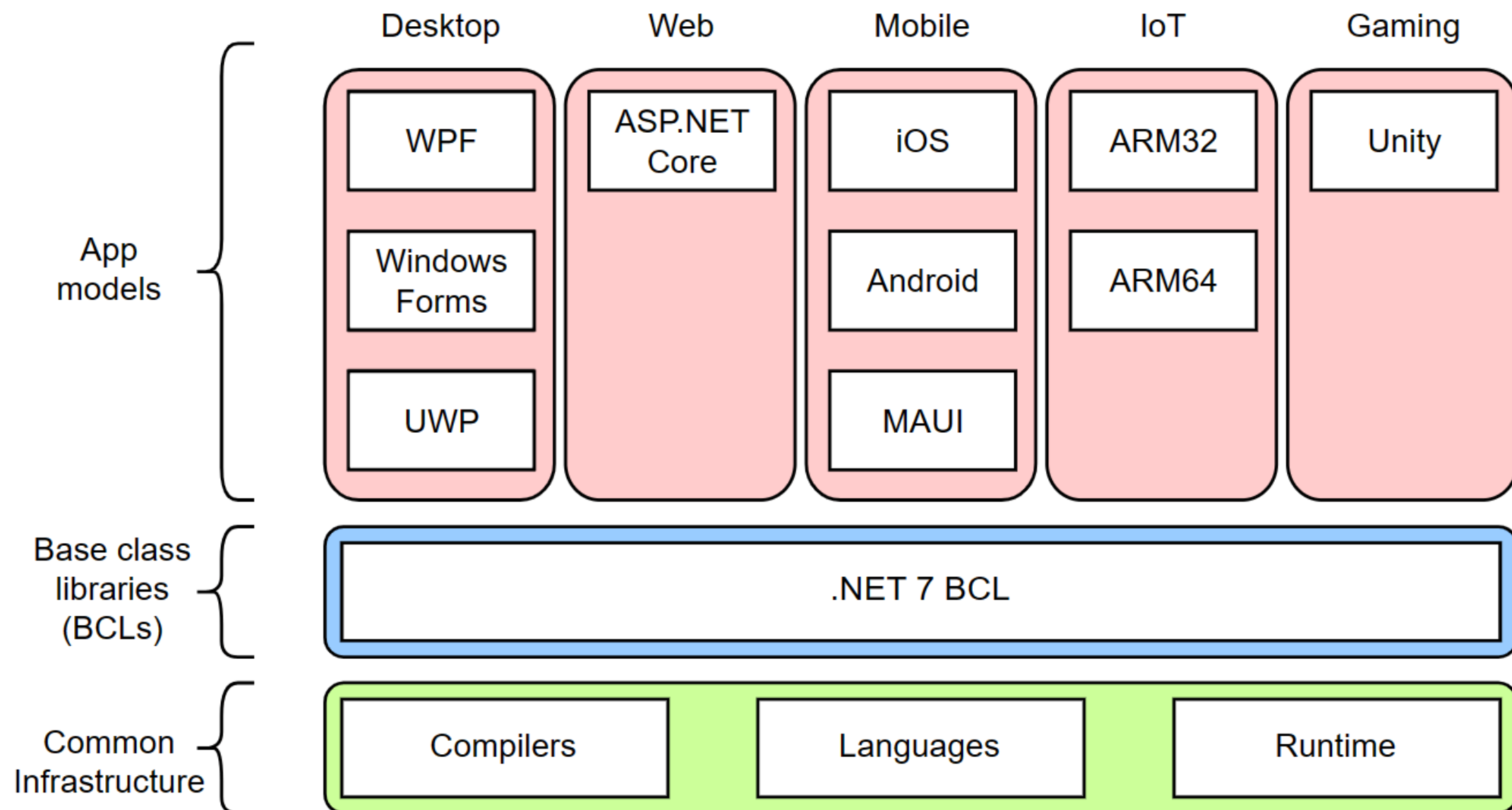
Xamarin/Mono



Xamarin / Mono - это реализация .NET для запуска приложений во всех основных мобильных операционных системах.

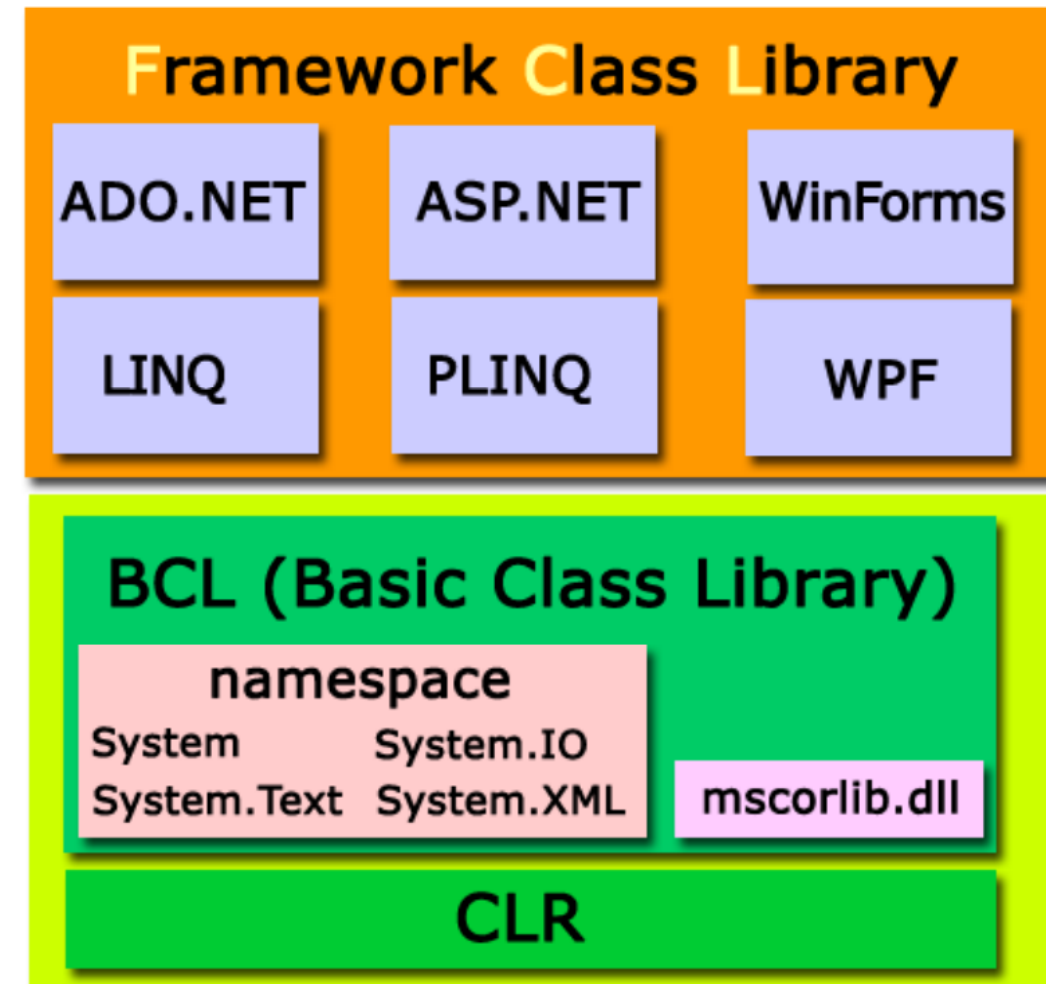
.NET FRAMEWORK ARCHITECTURE





CLR

- **Общезыковая среда выполнения** (Common Language Runtime) – среда выполнения, которая подходит для разных языков программирования.
- Основные возможности CLR (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация) доступны в любых языках программирования, использующих эту среду.

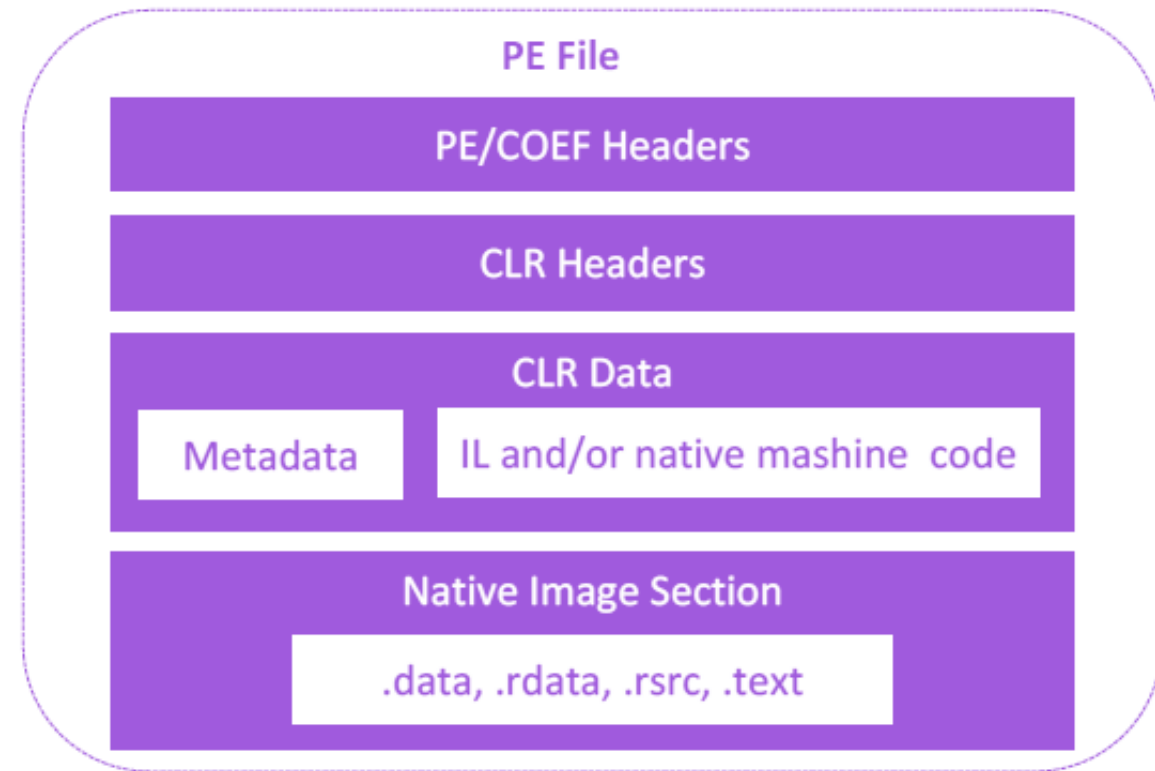


CLR

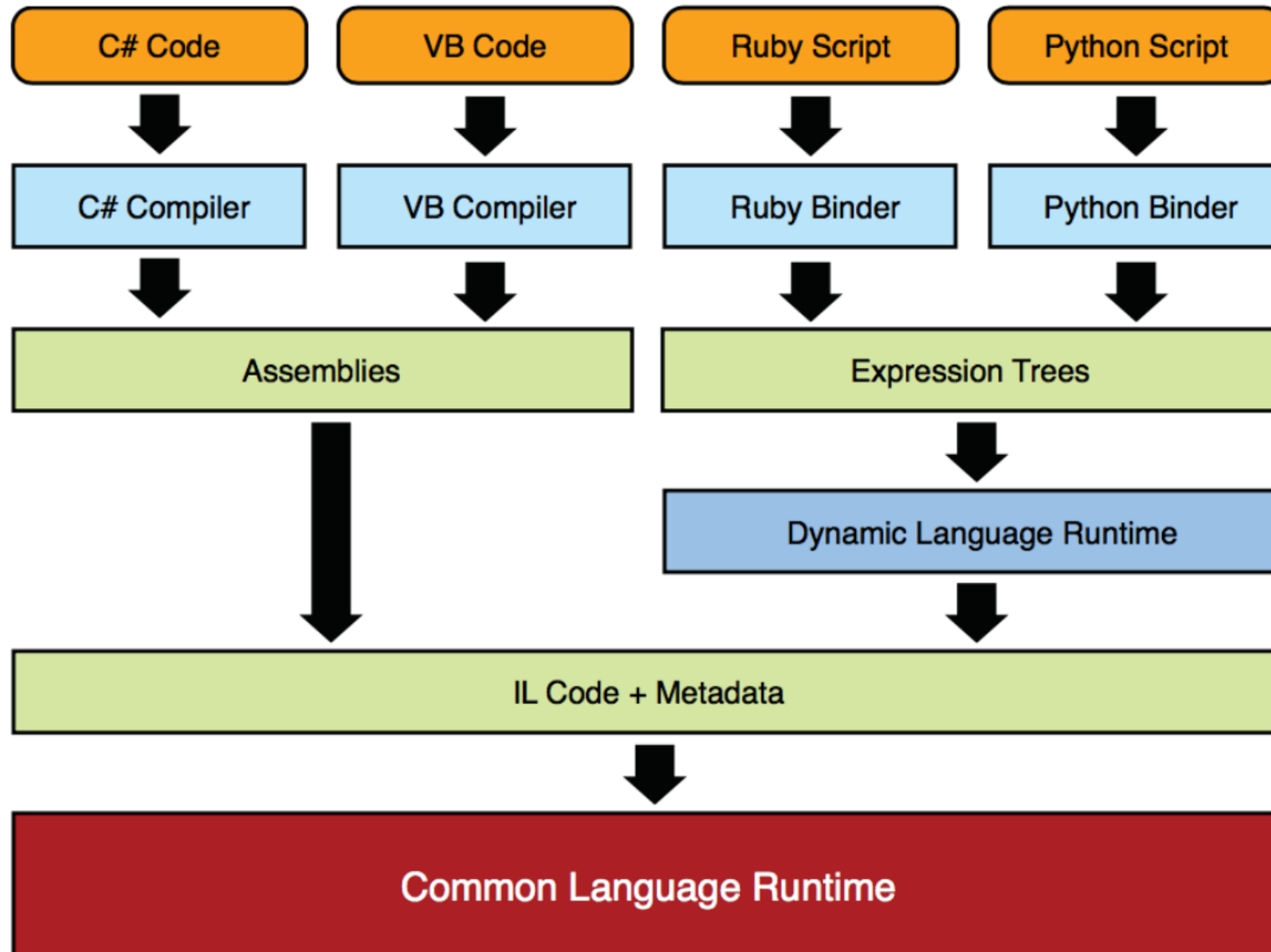
- Для создания, развертывания и поиска компонентов CLR имеет собственный набор концепций и технологий, принципиально отличающихся от тех, которые используются в COM, Java или Win32.
- Программы, написанные для выполнения CLR, находятся в управляемых модулях - байтовых потоках, хранящихся в виде файлов в локальной файловой системе или на web-сервере.
- Чтобы загрузчик операционной системы распознал исполняемый файл как действительный, он должен иметь структуру, определенную в формате файла PE/COFF (Portable Executable, Common Object File Format).

CLR

- Модуль CLR содержит **код, метаданные и ресурсы**. **Код** обычно хранится в формате **CIL**, хотя он также может быть сохранен в виде машинных инструкций, специфичных для процессора.
- **Метаданные** модуля описывают типы, определенные в модуле, включая имена, отношения наследования, сигнатуры методов и информацию о зависимостях.
- **Ресурсы** модуля состоят из статических данных только для чтения, таких как строки, растровые изображения и другие аспекты программы, которые не сохраняются как исполняемый код.



CLR



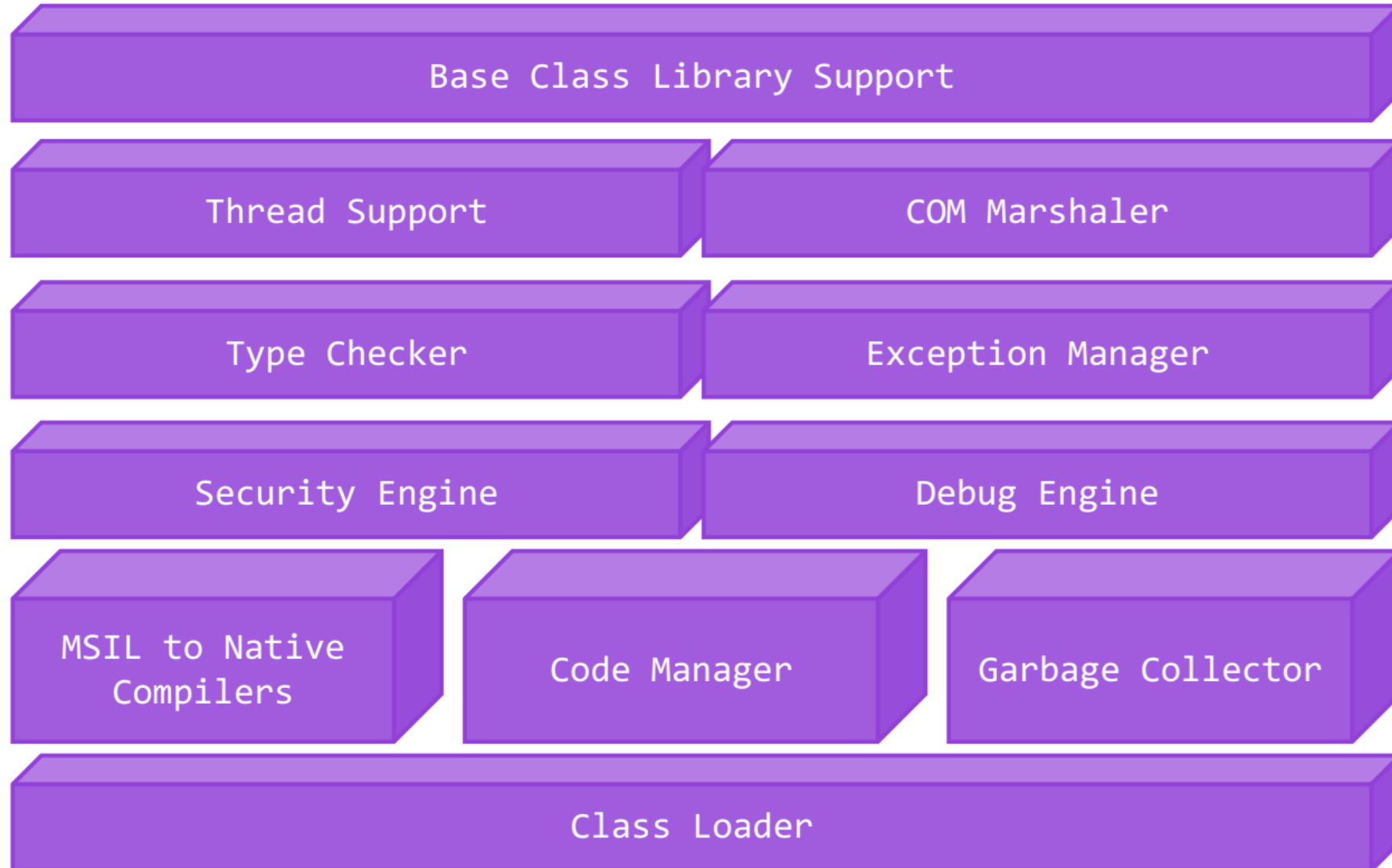
Метаданные

- Метаданные устраняют необходимость в заголовочных и библиотечных файлах при компиляции
- В процессе верификации кода CLR использует метаданные, чтобы убедиться, что код совершает только «безопасные» операции
- Метаданные позволяют сериализовать поля объекта в блок памяти на удаленной машине и затем десериализовать, восстановив объект и его состояние на этой машине
- Метаданные позволяют сборщику мусора отслеживать жизненный цикл объектов

IL

- Common Intermediate Language (сокращённо CIL) — «высокоуровневый ассемблер» виртуальной машины .NET. Промежуточный язык, разработанный фирмой Microsoft для платформы .NET Framework.
- IL является стековым языком — все его инструкции заносят операнды в стек вычислений (evaluation stack) и извлекают результаты из стека.
- IL не содержит инструкций для работы с регистрами (абстрагирует разработчика от конкретного процессора), что упрощает создание новых языков и компиляторов, генерирующих код для CLR.

CLR



Class loader

Загрузчик классов

- находит, загружает .NET-классы в память
- готовит их для исполнения
- кэширует информацию о классе, чтобы класс не пришлось загружать снова в процессе работы
- определяет, сколько требуется выделить памяти для экземпляра класса
- вставляет заглушку, вроде пролога функции, в каждый метод загруженного класса, предназначенную для того, чтобы отмечать состояние JIT-компиляции и для перехода между управляемым и неуправляемым кодом
- если загруженный класс ссылается на другие классы, загрузчик попытается загрузить эти классы, если классы, указанные в ссылках, уже были загружены, загрузчику ничего делать не надо
- использует соответствующие метаданные для инициализации статических переменных и создания экземпляра загруженного класса

Type Checker

Верификатор

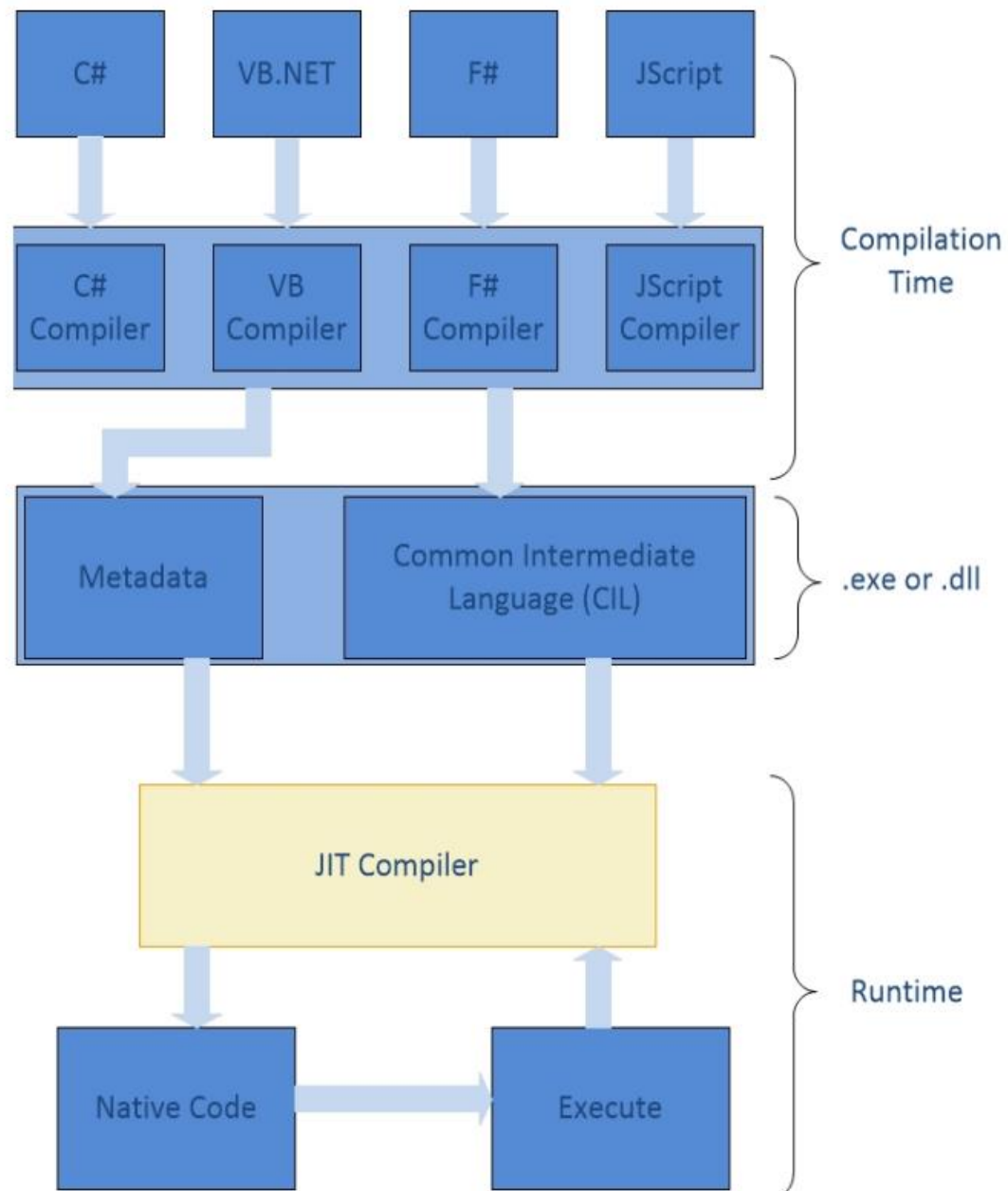
- проверяет являются ли метаданные корректными и действительными
- проверяет безопасен ли IL-код в отношении типов, т. е. корректно ли используются сигнатуры типов

MSIL to Native Compilers

JIT -компиляторы

- компилирует метод и преобразует его в управляемый машинный код
- генерирует управляемые данные, необходимые диспетчеру кода для поиска и разворачивания стековых фреймов

JIT-компиляция



Сервисы CLR для поддержки и управления исполнением

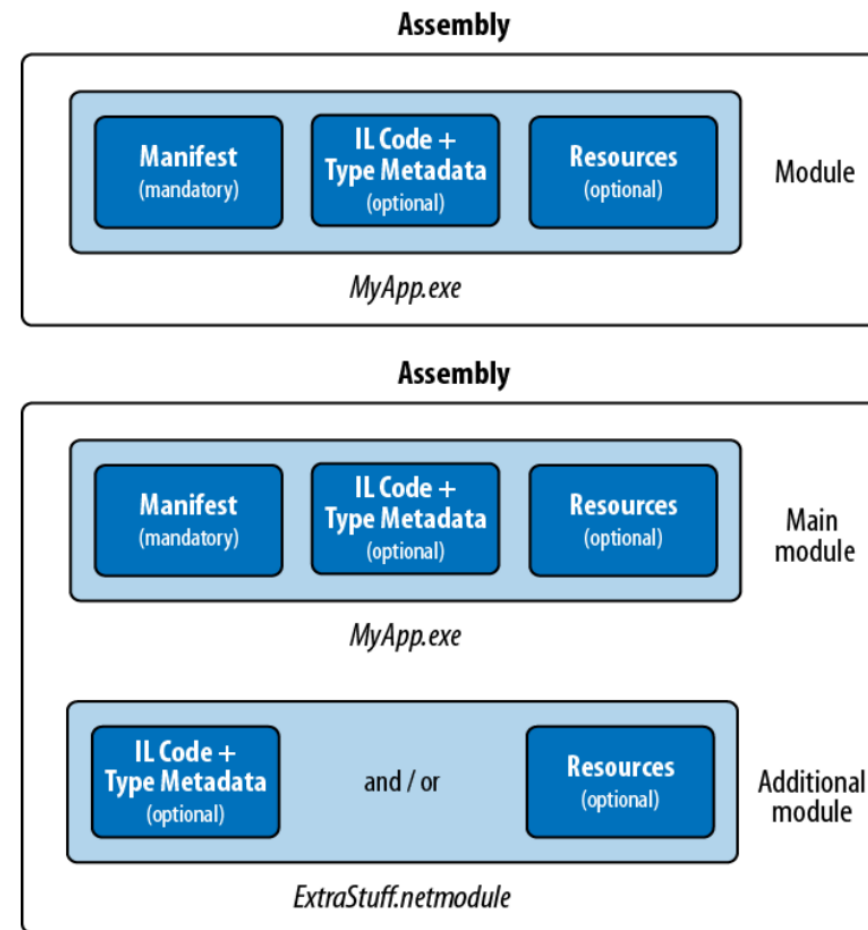
- **Code Manager** (диспетчер кода) использует управляемые данные для управления исполнением кода, в том числе перемещения по стеку
- **Garbage Collector** (сборка мусора) поддержка автоматического управления временем жизни всех объектов среды .NET
- **Exception Manager** (Обработка исключений) поддерживает стандартный механизм обработки исключений, работающий во всех языках, позволяя всем программам использовать общий механизм обработки ошибок. Механизм обработки исключений в CLR интегрирован со структурной обработкой исключений (Windows Structured Exception Handling, SEH)
- **Security Engine** (поддержка безопасности) осуществляет различные проверки безопасности, чтобы гарантировать, что код безопасен для исполнения и не нарушает никаких требований безопасности

Сервисы CLR для поддержки и управления исполнением

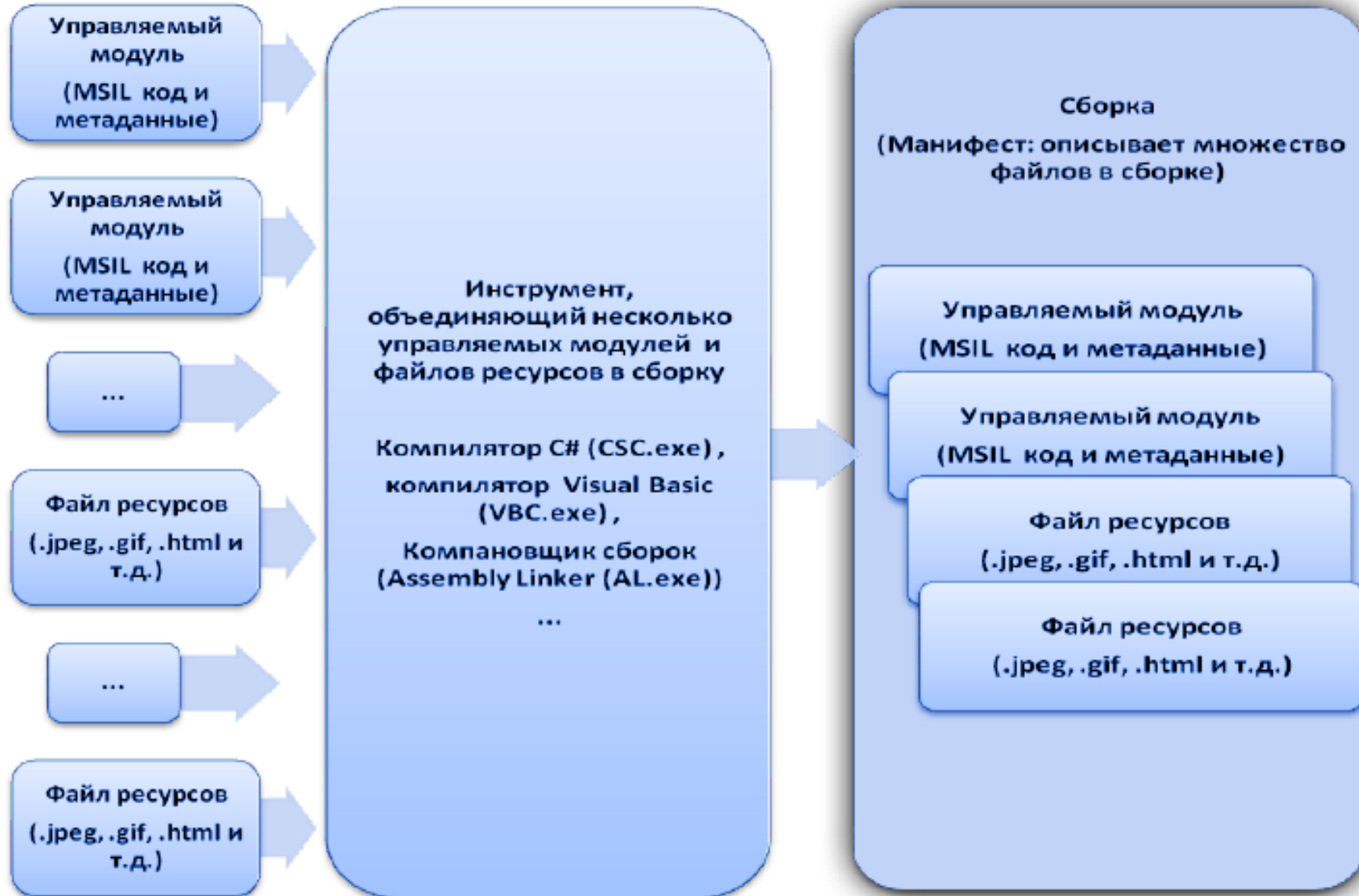
- **Debug Engine** (поддержка отладки) предоставляет богатую поддержку отладки и профилирования, содержит поддержку управления исполнением программы, точек останова, исключений, управляющей логики и т. п.
- **COM Marshaler** (поддержка взаимодействия кода) поддерживает взаимодействие между управляемым (CLR) и неуправляемым (без CLR) «мирами». Средство COM Interop играет роль моста, соединяющего COM и CLR, обеспечивая COM-объекту возможность использовать .NET-объект, и наоборот. Средство Platform Invoke (P/Invoke) позволяет вызывать функции Windows API.
- **Thread Support** (поддержка потоков) – CLR обладает собственной абстракцией, концептуально аналогичной потоку операционной системы

Assembly

- **Сборка** - логическая группировка одного или нескольких управляемых модулей и файлов ресурсов.
- Это самая маленькая единица с точки зрения повторного использования, безопасности и управления версиями.
- Номер сборки включает: Major version number, Minor version number, Build number, Revision number



Assembly



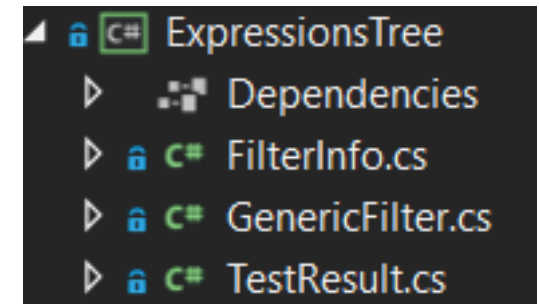
Assembly

- В сборке определены повторно используемые типы
- Сборка помечена номером версии
- Со сборкой может быть связана информация безопасности
- Сборка определяет границы типов
- Сборки являются самоописываемыми
- Сборки поддаются конфигурированию

PROJECT & SOLUTION

Project

- Проект содержит все файлы, которые будут скомпилированы в исполняемую программу, библиотеку или веб-сайт.
- Проект также содержит параметры компилятора и другие файлы конфигурации, которые могут потребоваться разным службам или компонентам, с которыми взаимодействует программа.

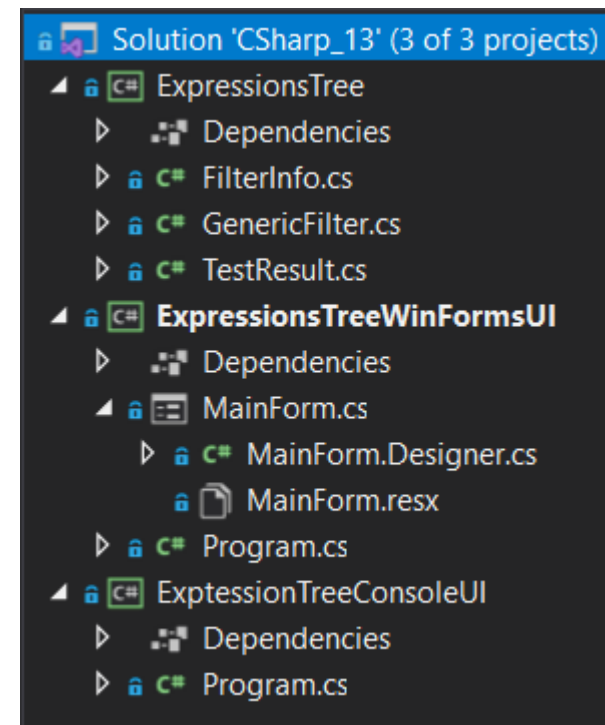


Project

- Visual Studio использует **MSBuild** для создания каждого проекта в решении, и каждый проект содержит файл проекта MSBuild.
- Файл проекта — это **XML-документ**, содержащий все сведения и инструкции, необходимые MSBuild для сборки проекта (содержимое, требования к платформе, сведения о версиях, веб-сервер или параметры сервера базы данных и выполняемые задачи).

Solution

Это контейнер для одного или нескольких связанных проектов вместе с информацией о сборке, параметрами окна Visual Studio и любыми прочими файлами, которые не относятся к какому-либо конкретному проекту.



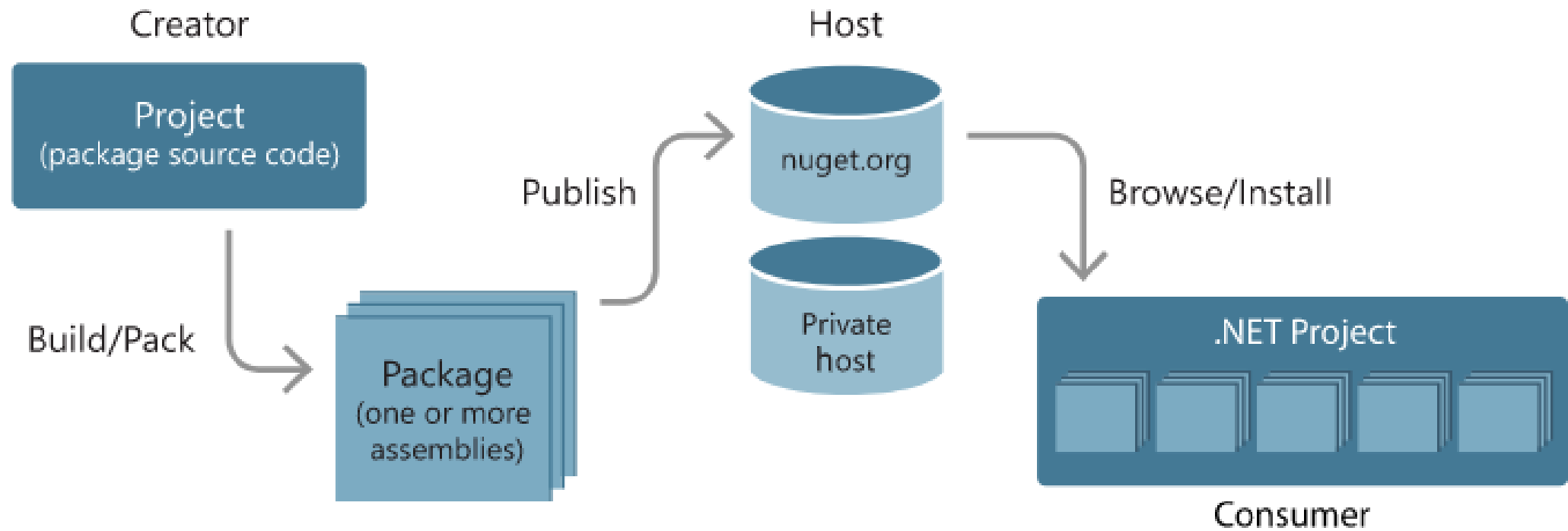
NUGET

NuGet

NuGet — система управления пакетами для платформ разработки Microsoft, в первую очередь библиотек .NET Framework.



NuGet





★ **Newtonsoft.Json** автор: James Newton-King

13.0.1

Json.NET is a popular high-performance JSON framework for .NET



★ **EntityFramework** автор: Microsoft

6.4.4

Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years of feature development and stabilization.



★ **log4net** автор: The Apache Software Foundation

2.0.14

log4net is a tool to help the programmer output log statements to a variety of output targets.

In case of problems with an application, it is helpful to enable logging so that the problem



★ **RestSharp** автор: John Sheehan, Andrew Young, Alexey Zimarev and RestSharp community

107.2.1

Simple REST and HTTP API Client



★ **NUnit** автор: Charlie Poole, Rob Prouse

3.13.2

NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.

Все пакеты лицензируются их владельцами. NuGet не несет ответственности за пакеты сторонних производителей и не предоставляет лицензии на такие пакеты.

ЯЗЫК C#

Синтаксис

С# имеет С-подобный синтаксис.

```
class Program
{
    Ссылка: 0
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

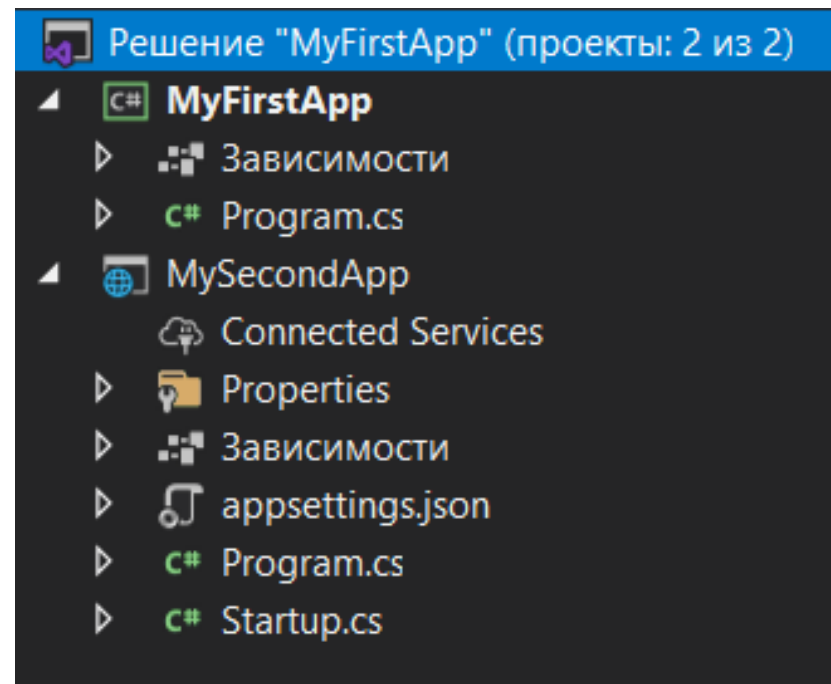
Naming convention

- Pascal Case для имен классов
- Pascal Case для имен методов
- Camel Case для переменных и параметров методов

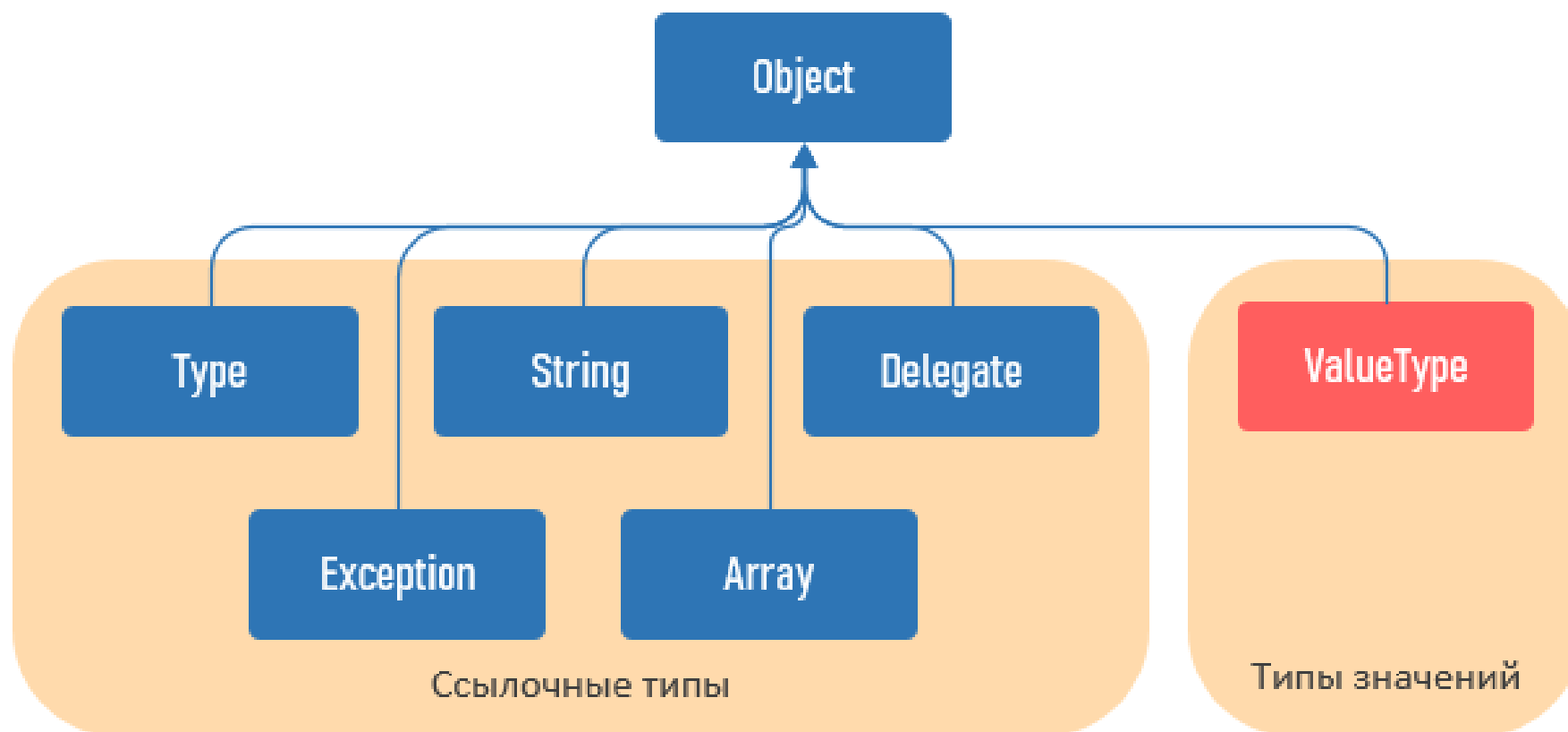
```
namespace MyFirstApp
{
    Ссылка: 0
    class MyFirstClass
    {
        int myFirstVariable;
        Ссылка: 0
        void MyFirstMethod(string myFirstParameter)
        {
            Console.WriteLine(myFirstParameter);
        }
    }
}
```

Solution

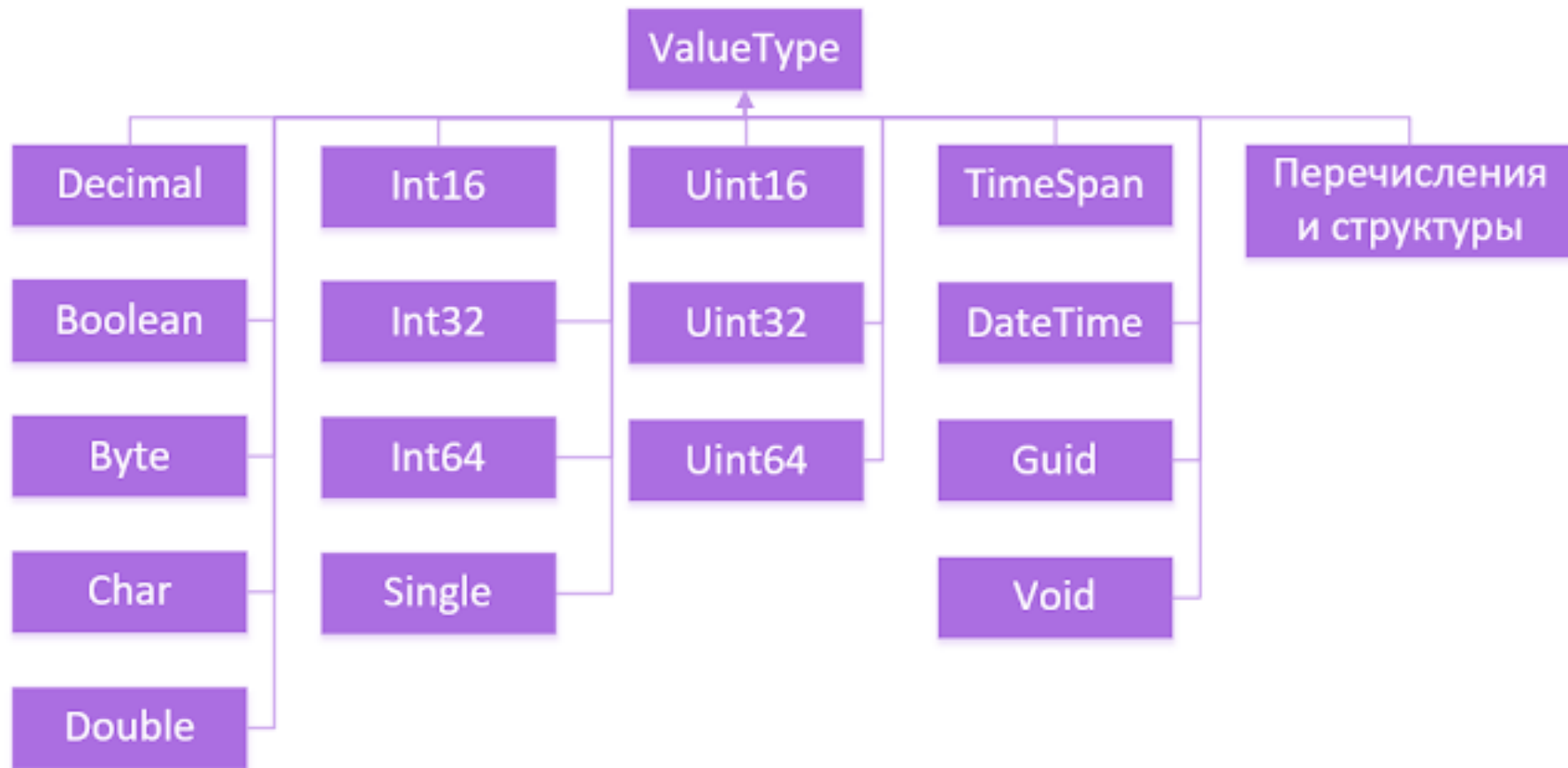
Одно решение может содержать несколько разнотипных проектов.



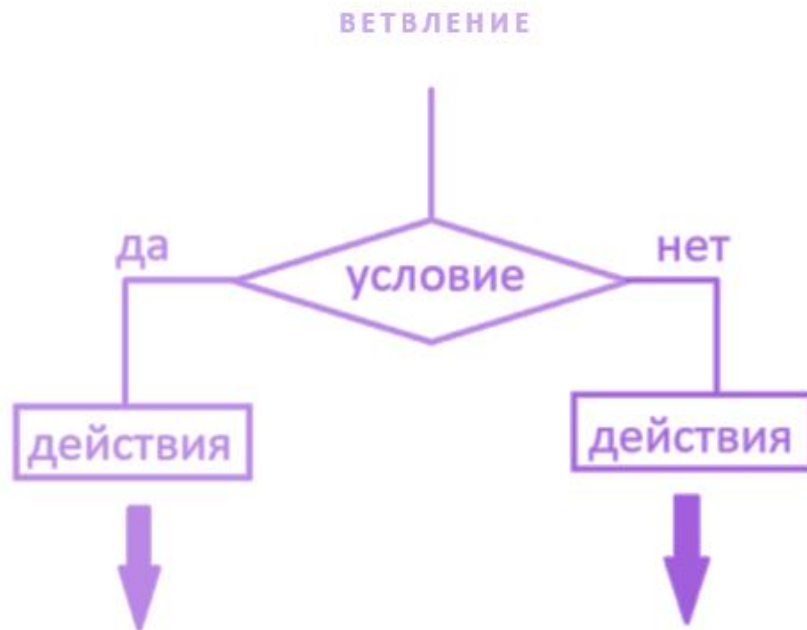
Типы данных



Значимые типы



Ветвления



```
if (condition)
{
    int summ = first + second;
}
else
{
    int diff = first - second;
}
```

Ветвления

```
if (condition)
{
    int summ = first + second;
}
```


```
bool condition = true;

bool subCondition = true;

// основное тело программы

if (condition)
{
    // какие-то сложные действия
}
else if (subCondition)
{
    // какие-то сложные действия
}
else
{
    // какие-то сложные действия
}

// основное тело программы
```



Ветвления

ВЕТВЛЕНИЕ SWITCH

```
string operation = '+';
```

```
string operation = '-';
```

```
string operation = "жаба";
```

```
switch (operation)
{
    case "+":
        // действия для случаев когда operation == "+"
        result = first + second;
        break;
    case "-":
        // действия для случаев когда operation == "-"
        result = first - second;
        break;
    default:
        // действия для случаев когда operation != "+"
        result = 0;
        break;
}
```

Массивы

```
int[] array1 = new int[10];  
var array2 = new int[10];  
var array3 = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

ЦИКЛЫ

```
var i = 0;
do
{
    array1[i] = i * 10;
    i++;
}
while (i < 10);

while (i < 10)
{
    array2[i] = i * 10;
    i++;
}
```


```
for (int j = 0; j < 10; j++)
{
    array3[j] = j * 10;
}

foreach (var element in array3)
{
    Console.WriteLine(element);
}
```

ЦИКЛЫ


```
count = count - 1;
```

```
while (condition)
{
    count--;
    result = first + 10;
}
```



```
count = count + 1;
```

```
while (condition)
{
    count++;
    result = first + 10;
}
```



Пространство имен

Ключевое слово `namespace` используется для объявления области действия, которая содержит набор связанных объектов.

Пространство имен можно использовать для организации элементов кода и для создания глобально уникальных типов.

```
namespace MyCompany.Proj1
{
    class MyClass
    {
    }
}
```

Пространство имен

Можно сделать вложенные пространства имен.

```
namespace SomeNameSpace
{
    public class MyClass
    {
        static void Main()
        {
            Nested.NestedNameSpaceClass.SayHello();
        }
    }

    // a nested namespace
    namespace Nested
    {
        public class NestedNameSpaceClass
        {
            public static void SayHello()
            {
                Console.WriteLine("Hello");
            }
        }
    }
}
```

C# 1.0
Управляемый
код

C# 2.0

- Обобщения
- Смешанные типы
- Анонимные методы
- Итераторы
- Null-типы

C# 3.0

- Неявно типизируемые локальные переменные
- Инициализаторы объектов и коллекций
- Автоматическая реализация свойств
- Анонимные типы
- Методы расширения
- Запросы
- Лямбда-выражения
- Деревья выражений

C# 4.0

- Динамическое связывание
- Именованные и дополнительные аргументы
- Обобщенная ковариантность и контрвариантность

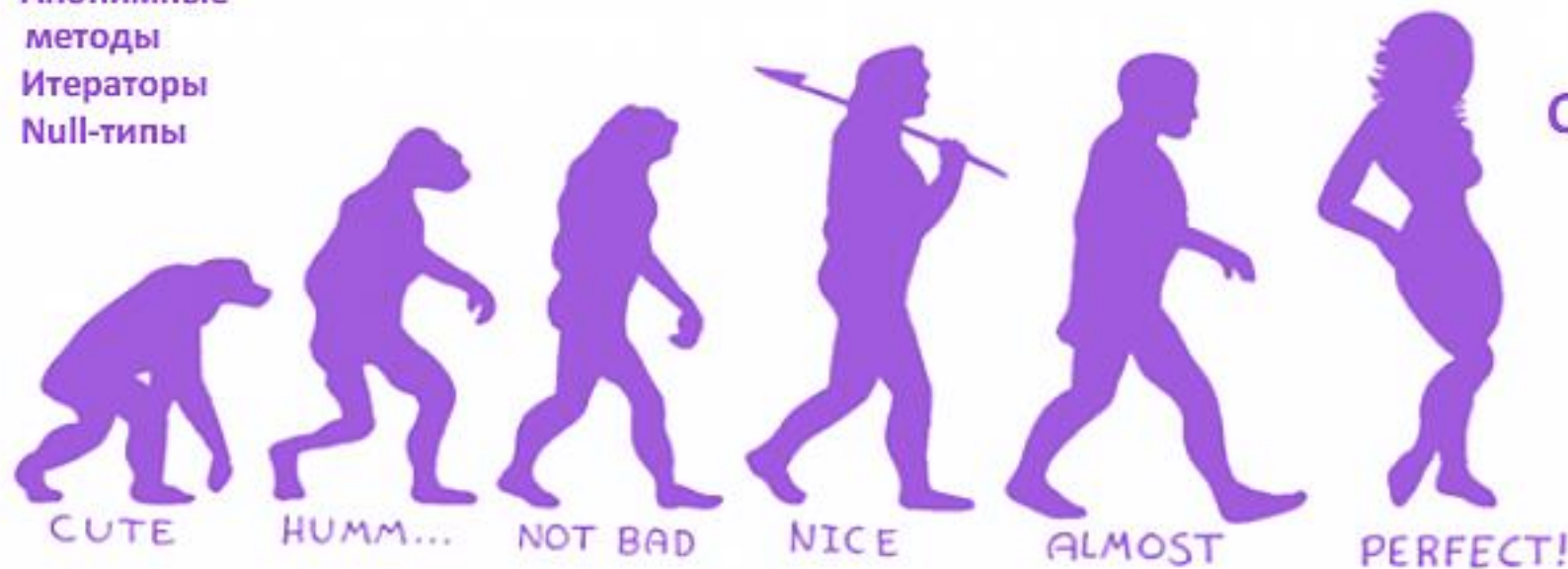
C# 5.0

Асинхронные
функции

C# 6.0

- Getter-only auto-properties
- Auto-property initializers
- Expression-bodied members
- Null-conditional operators
- Using static members
- Index initializers
- String interpolation
- nameof operator
- Await in catch/finally
- Exception filters
- Extension Add in collection initializers
- Improved overload resolution

C# 7.*!



ВОПРОСЫ ПО ЛЕКЦИИ