# Язык программирования Си-Шарп (C#)

# *Общие сведения о C#*

- **C#** (произносится си шарп) — разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы **Microsoft.NET**.

- Андерс Хейлсберг (Anders Hejlsberg; род. в декабре 1960, Копенгаген) — датский инженер-программист.
- В 1980 году написал свой первый компилятор языка Паскаль и продал его фирме **Borland**.

- Эта версия легла в основу **Turbo/Borland Pascal**, который развивался до 1995 года. До 1996 года Хйлсберг был главным проектировщиком фирмы **Borland**, где создал новое поколение компиляторов Паскаля — язык **Delphi**, компилятор которого работал уже под операционной системой Windows.
- В 1996 году он перешёл в **Microsoft**, где работал над такими проектами, как **J++** (версия Java) и **Microsoft Foundation Classes**. Позже возглавил группу по созданию и проектированию языка **C#**.

- ***C#*** относится к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к ***C++*** и ***Java***.

- Переняв многое от своих предшественников — языков ***C++***, ***Java***, ***Delphi***, ***Модула*** и ***Smalltalk*** — **C#**, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем

- Авторы ***C#*** стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков (вроде Java) с богатством возможностей и мощью ***C++***.
- По словам Андерса Хейлсберга, ***C#*** позаимствовал большинство своих синтаксических конструкций из ***C++.*** Некоторые синтаксические конструкции ***C#*** унаследованы и от ***Visual Basic***.

# История создания C#

- Проект **C#** был начат в декабре 1998 и получил кодовое название **COOL** (C-style Object Oriented Language).

- Версия 1.0 появилась вместе с платформой .NET в июне 2000 года, тогда же появилась и первая общедоступная бета-версия;

- **C# 1.0** окончательно вышел вместе с **Microsoft Visual Studio.NET** в феврале 2002 года.

| | | |
|---|---|---|
| C# 1.0 | Февраль 2002 | Managed code |
| C# 2.0 | Сентябрь 2005 | Iterators/Generics/Anonymous |
| C# 3.0 | Август 2007 | LINQ/Lambda |
| C# 4.0 | Апрель 2010 | Dynamic/PLIQ |
| C# 5.0 | Август 2012 | TAP (Task-based asynchronous pattern) |

# Структура программы в C#

Рассмотрим структуру простейшей программы на классическом примере вывода сообщения «Всем привет!»:

```
using System;
class FirstProject
{
    public static void Main(string[] args)
    {
      // вывод сообщения на экран
      System.Console.WriteLine("Всем привет!");
    }
}
```

Любая программа на языке C# - это набор классов, которые взаимодействуют друг с другом.

В одном из классов программы должна находиться, так называя «точка входа» - статический метод **Main**. Наличие или отсутствие этого метода определяет тип получаемого результата компиляции – сборки.

Если метод присутствует – получаем исполняемую программу EXE, в противном случае – библиотеку DLL. Классы могут быть вложены друг в друга. Но точка входа должна быть только в одном.

Класс определяется с помощью ключевого слова class, после которого идет имя класса. **Тело класса заключается в фигурные скобки.**

Язык программирования C# чувствителен к регистру символов, поэтому метод Main должен начинаться с большой буквы. Кроме того, этот метод должен быть определен как static, что позволит вызвать метод без создания объекта класса.

Заголовок можно упростить, удалив аргументы, которые, как правило, не задаются. Они имеют смысл, когда проект вызывается из командной строки, позволяя с помощью параметров задать нужную стратегию выполнения проекта.

## Правила синтаксиса.

При написании программы придерживаются синтаксических правил таких как:

{ }  операторные скобки объединяют несколько операторов в один блок,

;  конец оператора,

,  разделитель при перечислении констант, переменных,

( )  содержат параметры функций или операторов.

**Комментарии.**

C# представляет несколько механизмов комментирования кода:

- построчное //
- многострочное /* */
- комментарии, которые позволяют автоматически генерировать документацию в XML – формате. Эти комментарии начинаются с символов ///, за которыми следуют специальные тэги.

Код на C# представляет собой последовательность операторов, каждый из которых оканчивается точкой с запятой. Поскольку пустое пространство игнорируется, то можно располагать по несколько операторов в одной строке

C# — язык, обладающий блочной структурой; другими словами, каждый оператор является частью некоторого блока кода. Эти блоки, для обозначения начала и конца которых используются фигурные скобки { и }, могут содержать произвольное количество операторов или не содержать их вовсе. Фигурные скобки не сопровождаются точками с запятой.

# Пространство имен .NET Framework

- Программа на языке C# выполняется в среде .NET Framework
- **.NET Framework** состоит прежде всего из огромной библиотеки программ, к которой можно обращаться из различных языков программирования с помощью различных технологий объектно-ориентированного программирования (**FCL** - Framework Class Library)
- Число классов библиотеки FCL велико (более 4 тысяч). Поэтому понадобился способ их структуризации. Логически классы с близкой функциональностью объединяются в группы, называемые пространством имен (**Namespace)**.

**Так как классы группируются по пространствам имен**, то это означает, что в общем случае имя класса может иметь сложную структуру — состоять из последовательности имен, разделенных между собой точками. Последнее имя в этой последовательности собственно и является именем класса.

Классы, имена которых различаются лишь последними членами (собственно именами классов) последовательностей, считаются принадлежащими одному пространству имен.

Основным пространством имен библиотеки FCL является пространство **System**, содержащее как классы, так и другие вложенные пространства имен.

- В пространство **System** вложен целый ряд других пространств имен.
  - Например, в пространстве **System.Collections** находятся классы и интерфейсы, поддерживающие работу с коллекциями объектов - списками, очередями, словарями.
  - В пространство **System.Collections**, в свою очередь, вложено пространство имен **Specialized**, содержащие классы со специализацией, например, коллекции, элементами которых являются только строки.
  - Пространство **System.Windows.Forms** содержит классы, используемые при создании Windows-приложений. Класс **Form** из этого пространства задает форму - окно, заполняемое элементами управления, графикой, обеспечивающее интерактивное взаимодействие с пользователем.

- Часть библиотеки .NET Framework посвящена описанию некоторых базисных типов. Тип — это способ представления данных; определение наиболее фундаментальных из них (например 32-разрядного целого со знаком) облегчает совместное использование языков программирования с помощью .NET Framework. Все вместе это называется CTS (Common Type System — единая система типов)

- Помещение типа в пространство имен присваивает этому типу длинное имя, состоящее из всех пространств, как серии их имен, разделенных точками (.) и заканчивающееся именем класса.
- Если не использовать оператор using, для корректного обращения к функциям необходимо писать полный путь класса, что является достаточно емкой работой.

Средством "навигации" по пространствам имен, а точнее, средством, которое позволяет сокращать имена классов, является оператор

**using <ИмяПространстваИмен>;**

В приложении может объявляться собственное пространство имен, а также могут использоваться ранее объявленные пространства.

Оператор **using** сам по себе не обеспечивает доступа к именам, находящимся в других пространствах имен.

До тех пор, пока код из пространства имен не будет каким-либо способом привязан к нашему проекту (например, описан в исходном файле проекта или описан в каком-либо коде), привязанному к этому проекту, мы не получим доступа к содержащимся в нем именам.

Более того, если код, в котором содержится некое пространство имен, привязан к нашему проекту, то мы обладаем доступом к содержащимся в нем именам независимо от использования оператора `using`.

Оператор `using` всего лишь упрощает обращение к этим именам и позволяет сократить сильно удлиняющийся в противном случае код, делая его более понятным.

## Создание и запуск проекта

При создании нового проекта автоматически создается достаточно сложная вложенная структура – **решение,** содержащее проект, содержащий пространство имен, содержащее класс, содержащий точку входа. Для простых решений такая структурированность представляется избыточной, но для сложных – она осмысленна и полезна.

Проект можно скомпилировать, если компиляция прошла без ошибок, то в результате будет построена **сборка** в соответствующей папке **Debug** нашего проекта. **Сборка (assembly)** - это логическая единица, содержащая скомпилированный код для **.NET Framework**.

Приложение можно запустить на выполнение нажатием соответствующих клавиш (например, **CTRL+F5**). Приложение будет выполнено под управлением CLR. В результате выполнения появится консольное окно с предложением нажать любую клавишу для закрытия окна.