Платформа .**Net Framework** – новый подход к созданию и выполнению прикладных приложений (программ). Это ОО- платформа, которая позволяет использовать имеющиеся и создавать новые типы данных. В .Net под термином "тип" понимаются: классы, структуры, перечисления и другие типы данных. Платформа .Net позволяет разрабатывать компоненты (называемые сборками), которые предоставляют доступ к описанным в них типам данных другим компонентам (написанным на других языках программирования).

Основными целями разработки платформы .Net являлось создание:

- нового формата выполняемых программных модулей – компонент (EXE и DLL), называемых сборками (assembly) или управляемыми модулями, основными особенностями которых является использование общего (независимого от исходного языка) промежуточного языка программирования и метаданных, описывающих все открытые типы данных, содержащиеся в них;
- специальной виртуальной машины (общеязыковой исполняющей среды, **Common Language Runtime, CLR**), которая управляет компиляцией в инструкции процессора и выполнение модулей, составленных на промежуточном языке; CLR начинает работать при каждом запуске управляемых модулей на выполнение;
- общей библиотеки классов .NET Framework (**Framework Class Library, FCL**), которые позволяют выполнить все базовую функциональность управляемых приложений (например, работа с коллекциями, файлами, сетями, графическим интерфейсом и т.п.);
- набора программных средств, помогающих разрабатывать управляемые модули (например, такит как компиляторы и отладчики);
- основным средством разработки (система программирования) является интегрированная среда разработки – Visual Studio, позволяющая автоматизировать разработку приложений на всех языках программирования, поддерживаемых платформой.

Совокупность средств, с помощью которых осуществляется написание программ, их модификация, преобразование в машинные коды, отладка и выполнение, называют *средствами разработки* (средства разработки входят в состав системы программирования).

Система программирования *- язык программирования и набор программных средств, поддерживающих разработку и исполнение программ, написанных на этом языке. Для выполнения программа должна быть загружена в среду исполнения. В случае использования языка высокого уровня загрузке программы может предшествовать ряд преобразований, целью которых является приведение программы к виду, необходимому для загрузки в среду исполнения.*

Для долговременного хранения программа размещается на внешнем запоминающем устройстве в виде файлов. Часть программы, которая хранится в одном файле, называется ***модулем*** *(в простом случае программа хранится в одном файле). Имена файлов назначает разработчик, а расширения файлов назначаются автоматически.*

Модуль, содержащий программу на языке высокого уровня, называется ***исходным модулем****. Текст исходного модуля состоит из отдельных предложений, называемых* ***операторами****.*

Модуль, содержащий программу в виде, готовом для загрузки в среду исполнения, называется ***исполняемым модулем****. Есть две основные схемы преобразования исходного модуля в исполняемый модуль:* ***трансляция*** *и* ***интерпретация****.*

Схема ***трансляции*** *используется для представления исполняемого модуля в виде машинных команд (исходный модуль должен быть предварительно переведен на язык машинных команд; перевод выполняется специальной программой –* ***транслятором****).*

Схема ***интерпретации*** *используется для непосредственного распознавания и выполнения операторов исходного модуля. Распознавание и выполнение операторов возлагается на специальную программу –* ***интерпретатор*** *(в этом случае понятия исходного и исполняемого модуля совпадают).*

Средства разработки могут использоваться автономно или объединятся в систему. В первом случае запуск каждого из средств инициируется разработчиком путем ввода команды операционной системы. Средства разработки, объединенные в систему на основе общего интерфейса и общей базы данных, образуют *среду программирования*.

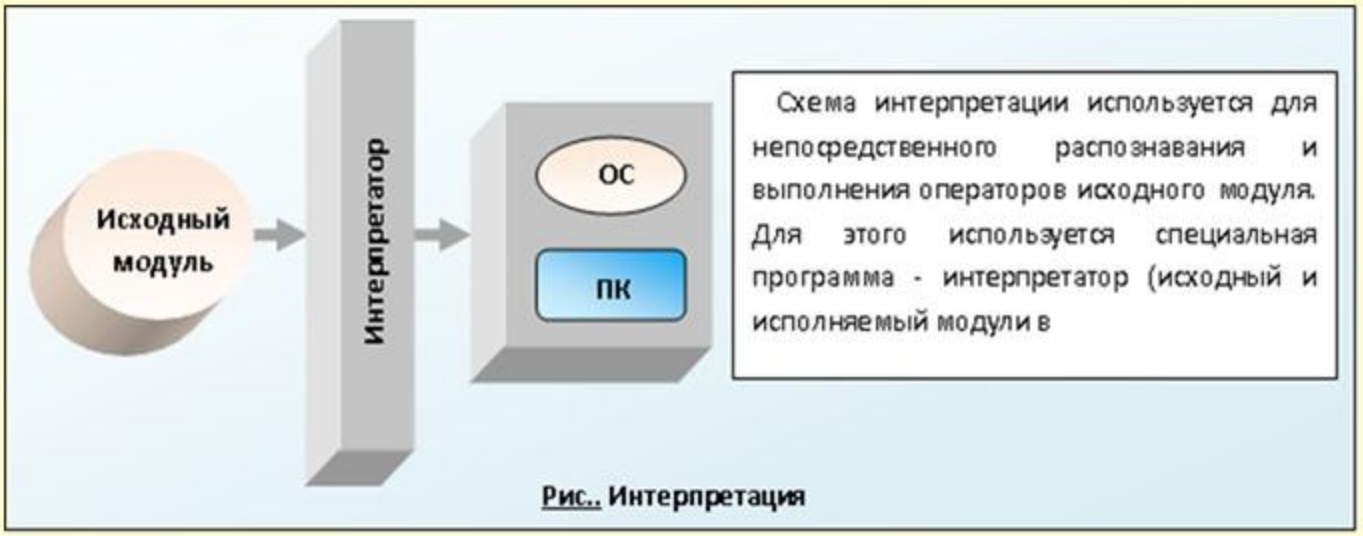

Рис.. Интерпретация

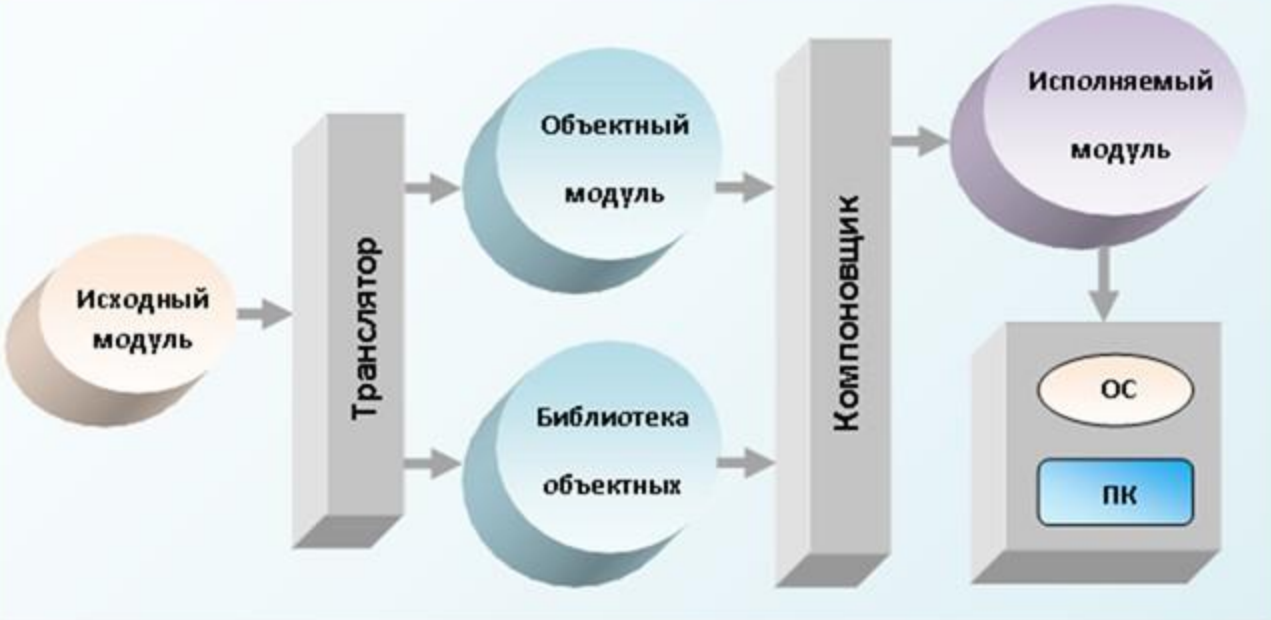

Схема трансляции используется для представления исполняемого модуля в виде машинных команд (исходный модуль предварительно должен быть переведен на язык машинных команд). Перевод выполняется специальной программой - **транслятором**. При разработке программ обычно используются ранее созданные подпрограммы, которые хранятся в библиотеке стандартных подпрограмм в виде, пригодном для загрузки в среду исполнения. Подключение стандартных подпрограмм может выполняться в ходе выполнения программы (динамически компонуемые библиотеки) или предварительно до загрузки исполняемого кода в среду исполнения (статически компонуемые библиотеки). В последнем случае модуль, полученный транслятором, называют объектным модулем. Операции по добавлению подпрограмм в библиотеку и удаления подпрограмм из библиотеки выполняются специальными программами.

Подключение стандартных подпрограмм возлагается на специальную программу - **компоновщик** (редактор связей). Транслятор и компоновщик являются составными частями системы программирования.

Рис. Трансляция

Платформа ***.Net Framework*** предназначена для разработки и выполнения:
-автономное консольное приложение (текстовый интерфейс);
-автономное Windows-приложение (графический интерфейс);
-автономное WPF-приложение (графический интерфейс);
-библиотека классов (для использования в других приложениях);
-Web-приложение (доступ к нему выполняется через браузер) которое по запросу формирует Web-страницу и отправляет ее клиенту по сети;
-Web-сервис – компонент, методы которого могут вызываться через Интернет;
ADO.Net приложения для работы с БД и др.

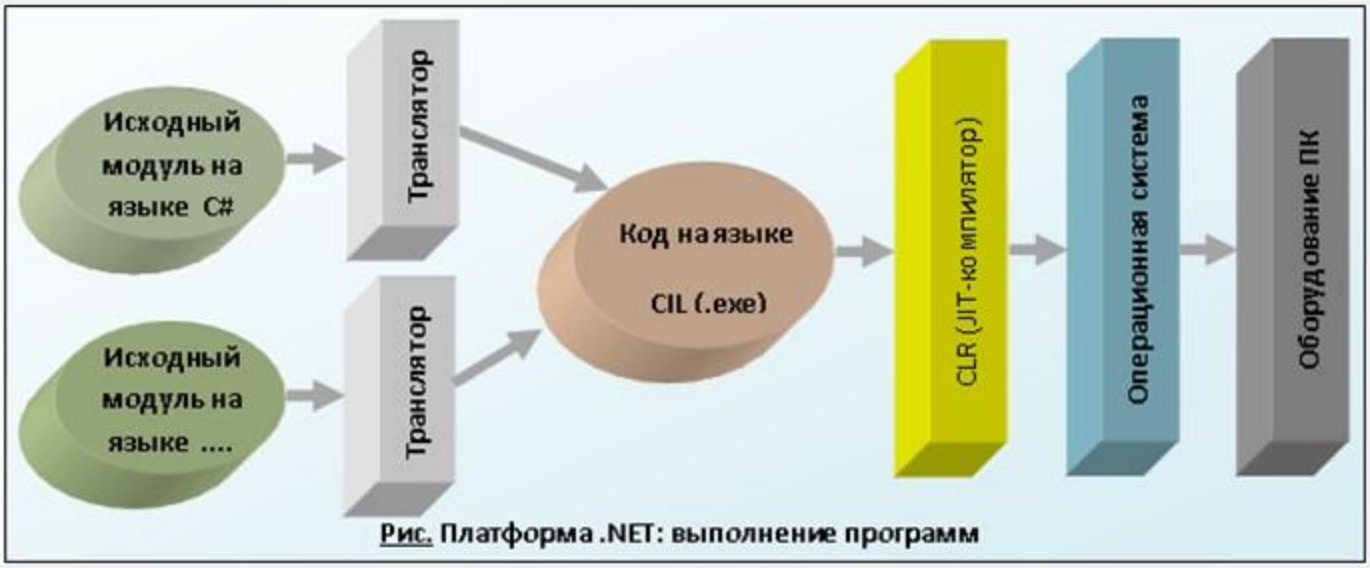

Рис. Платформа .NET: выполнение программ

Приложение выполняется в режиме **управляемого** или **небезопасного** кода.

- управляемый код: исходный код должен быть переведен на специально разработанный для платформы промежуточный язык CIL. Для исполнения кода на промежуточном языке приложения используется программная компонента платформы - общеязыковая среда исполнения CLR.
- небезопасный код: исходный код должен быть переведен на язык машинных команд. Машинный код исполняется непосредственно под управлением ОС.

Платформа .NET Framework является надстройкой над операционной системой (в качестве которой может выступать любая версия Windows, Unix и другие ОС) и состоит из ряда компонентов.

Платформа .NET Framework включает в себя:

- ✓ Четыре официальных языка: **C#**, VB.NET, Managed C++ и JScript.NET.
- ✓ Общеязыковую объектно-ориентированную среду выполнения CLR, совместно используемую этими языками для создания приложений.
- ✓ Ряд связанных между собой библиотек классов под общим именем FCL.

Платформа .NET Framework содержит две компоненты:

- Статическая компонента - базовая библиотека классов, содержащая обширный набор готовых к использованию программных компонент на промежуточном языке. Базовая библиотека классов является общей для всех языков программирования, поддерживаемых в платформе.
- Динамическая компонента – общеязыковая среда выполнения (CLR).

Взаимосвязи компонентов платформы .NET представлены на рис.

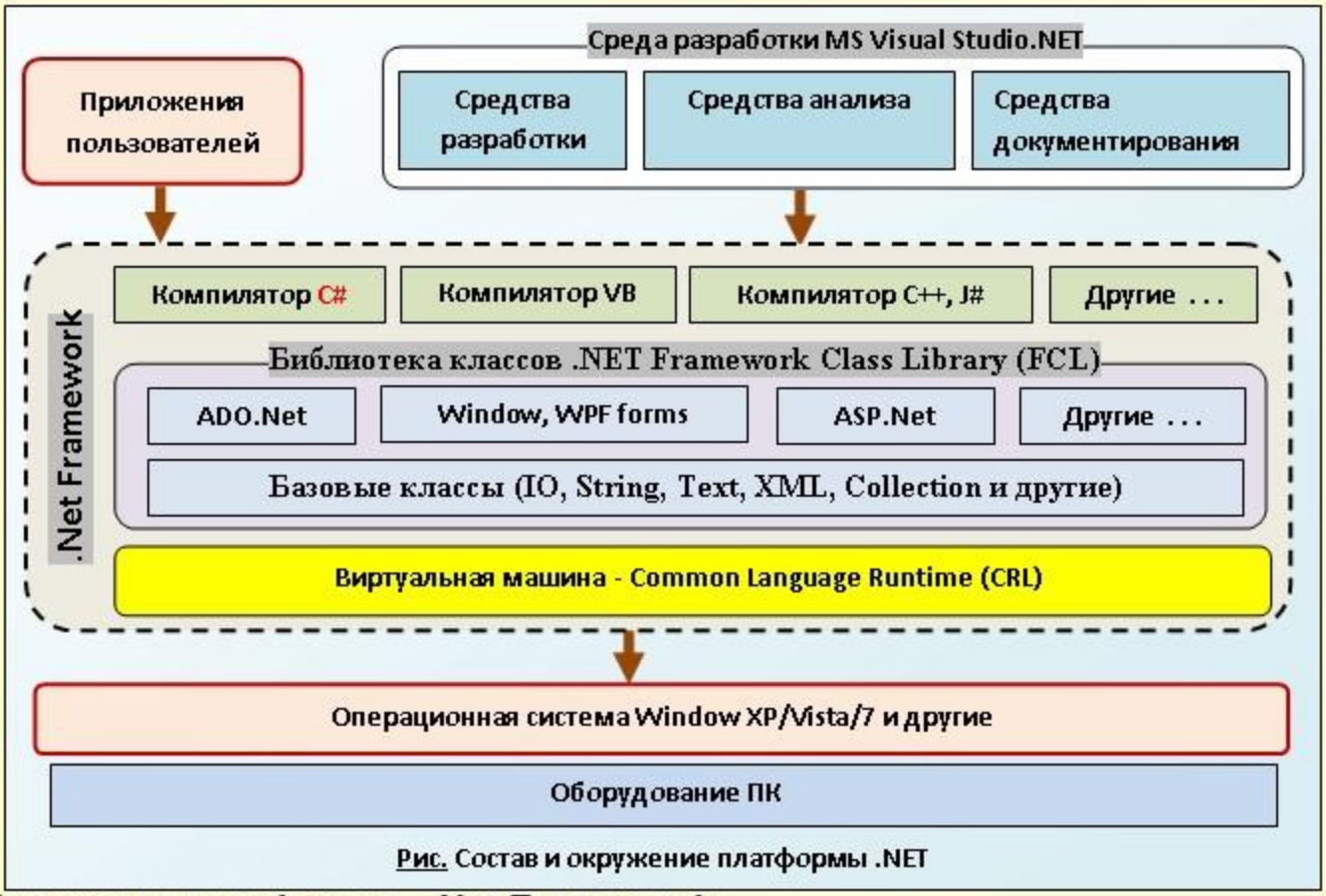

Рис. Состав и окружение платформы .NET

Термины платформы - .Net Framework

CLI *(Common Language Infrastructure -общеязыковая инфраструктура) - это набор стандартов для связи компонентов платформы в единую систему.*

CLI *- система архитектуры, соглашений и правил построения* ***.Net***

Основные компоненты CLI:

CLR – Common Language Runtime – Общеязыковая исполняющая среда
CLS – Common Language Specification – Общеязыковая спецификация
BCL – Base Class Library – Базовая библиотека классов
CTS – Common Type System – Общая система типов данных
CIL – Common Intermediate Language– Общий промежуточный язык

CLI и C# приняты в качестве международного стандарта ECMA International *(European Computer Manufacturers Association).*

Наиболее важными частями CLI являются CTS и CLS.

CTS *(Common Type System) – определяет характеристики типов данных, которые можно использовать в управляемом коде. Типы данных в каждом NET-языке соответствуют типам определенных CTS. Базовым типом данных для всех типов данных – является тип* ***object*** *(объект).*

CLS *(Common Language Specification) - описывает правила, свойства и поведение NET-совместимых языков программирования. CLS – определяет конструкции построения классов, передачу параметров, типы данных.*

Для работы с ОС Windows используется библиотека классов .NET Framework (содержит более 10 000 различных системных типов данных: классов, структур, интерфейсов, перечислений и делегатов). Библиотека FCL объектно-ориентированная и используется всеми языками, которые работают с платформой .Net.

Для простоты использования FCL, все ее содержание структурировано в виде иерархически организованных групп типов. Каждая группа типов называется ***пространством имен***. Всего в FCL около 100 таких пространств. В каждом из них содержатся классы и другие типы, имеющие некоторое общее назначение. Например, большая часть API Windows для управления окнами содержится в пространстве имен System.Windows.Forms (здесь находятся все классы, представляющие окна, диалоги, меню и другие элементы, обычно применяемые в приложениях с графическим интерфейсом пользователя). Отдельное пространство – System.Collections – содержит классы коллекций и словарей, в пространстве имен System.IO – классы для работы с данными на внешних устройствах.

Пространство имен	Содержимое
System	базовые типы данных и вспомогательные классы
System. Collections	коллекции, словари, массивы переменной размерности и другие контейнеры
System.Data и др.	классы ADO.NET для доступа к данным
System.Drawing	классы для рисования в окне (GDI+)
System.IO	классы файлового и потокового ввода – вывода
System.Net	классы для работы с сетевыми протоколами (http и др.)
System.Reflection и др.	классы для чтения и записи метаданных
System.Runtime. Remoting и др.	классы для распределенных приложений
System. ServiceProcess	классы для создания служб Windows
System.Tbreading	классы для создания и управления потоками
System.Web	классы для поддержки протокола http
System. WebServices	классы для разработки Web-сервисов
System.Web.Services.Protocols	классы для разработки клиентов Web-сервисов
System.Web.UI	основные классы, используемые ASP.NET
System.Web. UI.WebControls	серверные элементы управления ASP.NET
System.Windows. Forms	классы для реализации графического интерфейса пользователя
System.Xml и др.	классы для чтения и вывода данных в формате XML

Компиляция и выполнение программы в среде CLR Раньше почти все компиляторы генерировали код для конкретных процессорных систем. Все CLR-совместимые компиляторы генерируют CIL код - **управляемый модуль** (т.к. CLR управляет жизненным циклом и выполнением кода программы).

***Составные части управляемого модуля*:**

Заголовок PE32 или PE32+: Файл с заголовком в формате PE32 может выполняться в 32- или 64-разрядной ОС, а с заголовком PE32+ только в 64 разрядной ОС. Заголовок показывает тип файла: GUI, GUI или DLL, он также имеет временную метку, показывающую, когда файл был собран. Для модулей, содержащих только IL-код, основной объем информации в PE-заголовке игнорируется, Для модулей, содержащих процессорный код, этот заголовок содержит сведения о процессорном коде.

Заголовок CLR: Содержит информацию, которая превращает этот модуль в управляемый. Заголовок включает нужную версию CLR, некоторые флаги, метку метаданных, точки входа в управляемый модуль (метод **Main**), месторасположение и размер метаданных модуля, ресурсов и т.д.

Метаданные - это набор таблиц данных, описывающих то, что определено в модуле. Есть два основных вида таблиц: описывающие типы и члены, определенные в исходном коде, и описывающие типы и члены, на которые имеются ссылки в исходном коде. *Метаданные:*

- устраняют необходимость в заголовочных и библиотечных файлах при компиляции, так как все сведения о типах и членах, на которые есть ссылки, содержатся в файле с IL-кодом, в котором они реализованы. Компиляторы могут читать метаданные прямо из управляемых модулей.
- при компиляции IL-кода в машинный код CLR выполняет верификацию (проверку «безопасности» выполнения кода) используя метаданные, например, нужное ли число параметров передается методу, корректны ли их типы, правильно ли используется возвращаемое значение и т.д.
- позволяют сборщику мусора отслеживать жизненный цикл объектов и т.д.

CIL-код: управляемый код, создаваемый при компиляции исходного кода. Во время исполнения CLR компилирует IL-код в команды процессора.

По умолчанию CLR-совместимые компиляторы генерируют *управляемый* код, безопасность выполнения которого проверяется средой CLR. Однако можно разрабатывать неуправляемый или «небезопасный» код, который может работать непосредственно с адресами памяти и управлять байтами в этих адресах. Эта возможность, обычно полезна при взаимодействии с неуправляемым кодом или при необходимости добиться максимальной производительности при выполнении критически важных алгоритмов. Однако использовать неуправляемый код довольно рискованно, т.к. он способен разрушить существующие структуры данных.

Преимущества платформы .Net для разработки программ:

- *Объектно-ориентированное программирование* - .Net Fraemwork и C# полностью базируются на объектно-ориентированных принципах;
- *Дизайн* - библиотека классов организована с очень понятным интерфейсом;
- *Независимость о языка* - языки C#, J#, C++ обладают возможность взаимодействия, так как компилируются в общий язык – CIL;
- *Динамические Web - страницы* - в .Net включена интегрированная поддержка Web - страниц с применением новой технологии - ASP.NET;
- *Доступ к данным* - компоненты ADO.NET предоставляют эффективный доступ к базам данным. ADO.NET - основная модель доступа к данным(базам данных) для приложений, основанных на Microsoft .NET. Представляет собой самостоятельную технологию. Компоненты ADO.NET входят в поставку оболочки .NET Framework;
- *Встроена поддержка XML.* XML - расширяемый язык разметки, представляющий собой свод общих правил. XML – текстовый формат, предназначенный для хранения структурированных данных и обмена информацией между программами;
- *Разделение кода* - .Net заменил способ разделения кода между приложениями, введя концепцию сборки, заменившая .DLL;
- *Безопасность приложений* - каждая сборка содержит информацию о безопасности и определяет какая категория пользователей может работать с классами и процессами;
- *Поддержка Web -служб* - процесс разработки web - служб стал наиболее доступен и разрабатывается как обычное программное обеспечение;
- *C#* - новый объектно - ориентированныя язык, предназначенный для применения с .NET;
- *Строгая иерархичность* организации пространств для типов, классов и имен сущностей программы позволяет стандартизировать и унифицировать реализацию;
- *Новый* подход к интеграции компонент приложений в среде вычислений Internet (или так называемые веб-сервисы) дает возможность ускоренного создания приложений для широкого круга пользователей;.

Разработка приложений для платформы Framework.Net может выполняться на любом языке, который формирует код на общем промежуточном языке CIL и поддерживает взаимодействие с библиотекой FCL.

Язык C# был специально разработан для платформы Framework.Net и в полной мере учитывает все ее возможности – как FCL, так и CLR.

Создателем языка C# является сотрудник Microsoft - Anders Hejlsberg (который ранее был ведущим разработчиком среды программирования Delphi. C# создавался - как язык компонентного, объектно-ориентированного программирования, и в этом одно из главных его достоинств, направленное на возможность повторного использования созданных компонентов.

Основные достоинства языка:

- C# - полностью объектно-ориентированный язык (встроенные в язык типы являются объектами классов);
- C# - мощный объектный язык с возможностями наследования и универсализации (создания обобщенных классов);
- C# - наследник языка C++. Сохранив основные черты великого родителя, C# стал проще и надежнее. Простота и надежность, связаны с тем, что C# хотя и допускает, но не поощряет опасные свойства C++ (указатели, адресация, адресная арифметика);
- мощная библиотека .Net Framework поддерживает удобство построения различных типов приложений на C#;
- унифицированная система типизации (соответствует идеологии Microsoft . Net Framework);
- *свойства* как средство *инкапсуляции данных*;
- *обработка событий* (с расширениями в части *обработки исключений)*;
- *делегаты* (delegate – развитие *указателя* на функцию в языках C и C++);
- *индексаторы* (indexer – операторы индекса для обращения к элементам контейнера, массива);
- *перегруженные* операторы;
- оператор *foreach* для обработки элементов классов-коллекций;
- механизмы *boxing* и *unboxing* для преобразования типов;
- *атрибуты* (средство оперирования метаданными).