Введение в платформу .Net

Этапы развития программирования

- 40-е годы. Машинный код
- 1950 год. Ассемблер (язык низкого уровня, т.е. учитывающий особенности процессора)
- С 1957-й. Джон Бэкус. первый язык высокого уровня Фортран
- 1958-й ALGOL 68 (подпрограммы)
- 1960-й LISP LISt Processing (обработка списков)
- 1959-й COBOL насыщен разнообразными возможностями поиска, сортировки и распределения
- К концу 1960-х. Появление структурного программирования
- В конце 1970-х и начале 1980-х - появление классов и ООП.

Причины создания .Net

- Объединение всех наиболее удачных наработок в рамках единой платформы и их унификация.
- Новая платформа должна была напрямую поддерживать объектно-ориентированность, безопасность типов, сборку мусора и структурную обработку исключений.
- Конкуренция с Java
- Поддержка разработки веб-ориентированных приложений.

Плюсы .Net

- ООП
- Надежная система кеширования
- интегрированная среда разработки (IDE). Это упрощает для разработчиков весь процесс разработки, тестирования и развертывания.
- Кросс-платформенная функция
- Автоматический мониторинг
- Великая библиотека
- Большое сообщество
- Гибкое развертывание и простое обслуживание

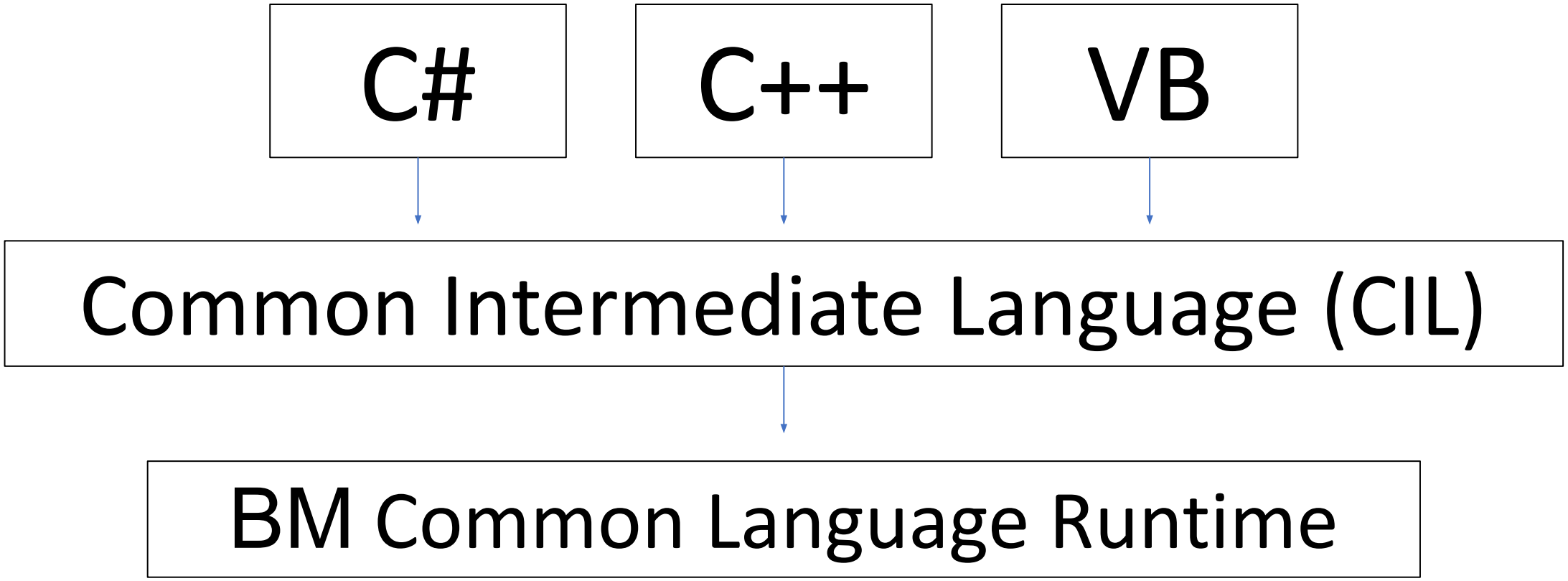
Минусы .Net

- **Ограниченная объектно-реляционная поддержка:** если в базу данных и дизайн вносятся какие-то изменения, она не поддерживает исходную версию и работает только в новых.
- **Привязка к поставщику:** разработчики должны соблюдать правила Microsoft.
- **Стоимость лицензирования:** когда размер проекта большой и сложный, лицензирование становится дорогостоящим.
- **Разрыв между выпуском и стабильностью** .

Языки .Net

- C#,
- C++
- F#
- Visual Basic
- Python
- JavaScript
- И многие другие

Архитектура .Net Framework



Common Language Specification

документ, в котором говорится, как компьютерные программы могут быть превращены в код SIL

Common Type System

Формальная спецификация, определяющая, как какой-либо тип (класс, интерфейс, структура, встроенный тип данных) должен быть определён для его правильного выполнения средой .NET. Тип может быть описан как определение набора допустимых значений (например, «все целые от 0 до 10») и допустимых операций над этими значениями (например, сложение и вычитание).

Base Class Library

Base Class Library, или так называемая .NET FCL (англ. *Framework Class Library*), сокращённо *BCL* — стандартная библиотека классов для всех языков, поддерживающих платформу .NET.

Метаданные

Метаданные — это данные в двоичном формате с описанием программы, хранящиеся либо в переносимом исполняемом (PE) файле среды CLR, либо в памяти. В метаданных описываются все типы и члены, определенные или используемые в модуле или сборке. (**Portable Executable** (**PE)**, «переносимый исполняемый»)

Манифест

Любая сборка содержит коллекцию данных с описанием того, как ее элементы связаны друг с другом. Эти метаданные содержатся в манифесте сборки.

Сборка

Сборка представляет собой коллекцию типов и ресурсов, собранных для совместной работы. Сборки создаются в виде исполняемого файла (*EXE*) или файла библиотеки динамической компоновки (*DLL*) и являются стандартными блоками приложений .NET.

Плюсы C#

- Язык активно развивается. Регулярно выходят новые версии C#, которые добавляют новые синтаксические конструкции в язык, а также увеличивают его быстродействие и надежность.
- объектно-ориентированный подход
- обилие синтаксического сахара (тернарный оператор if, +=, ? для проверки на null и пр.)
- наличие большого количества библиотек
- строгую типизацию, которая позволяет защититься от дурака
- кроссплатформенность

Минусы C#

- C# очень легко дизассемблируется. Это означает, что с большой долей вероятности твой код будет получен и изучен конкурентами.
- .NET использует концепцию JIT-компиляции. Это означает, что программа будет скомпилирована в машинные коды по мере необходимости прямо во время работы приложения. При первом запуске возможны весьма серьезные тормоза.
- C# не является повсеместно распространенным языком.

От себя

+

Многие языки программирования роботов на биржу используют C#.

-

Язык менее гибкий, чем, к примеру, связка php и js.

Рефлектор
Дотфускатор

Рефлектор требуется для декомпиляции файлов. Самый известный рефлектор Red Gate Reflector.

Защитник кода .NET от декомпиляции с помощью так называемой обфускации. Можно выделить встроенный в .NET дотфускатор Dotfuscator Community. Другие дотфускаторы C# Source Code Obfuscator, Eazfuscator, AppFuscator, Obfuscar

Простейшая программа на C#

У вас на экране ☺️

Типы данных

- Целочиленные
- Дробные (с плавающей запятой)
- Символьный и строковые
- Булев (логический)
- Типы связанные со временем
- Object
- var

Числовые типы данных

Целочисленные

С плавающей запятой

- sbyte
- byte
- short
- ushort
- int
- uint
- long
- ulong
- Int16
- Int32

Строковые и символьные типы

- char (‘c’)
- Char
- string (“строка”)
- String
- StringBuilder (“строка”)

Другие типы данных

Логический (булев) тип

- bool
- Boolean

Дата и время

- DateTime
- TimeSpan (внутридневной тип)

Простейшие ввод и вывод на консоль

- Console.Write(“Привет ”+name)
- Console.WriteLine(“Привет {1}, {0}”,surname, name)
- Console.WriteLine($"Имя: {name} Возраст: {age}");
- Str=ReadLine();

Работа со знаками действий

- = (присвоение)
- +, +=
- -, -=
- *, *=
- / (для дробных типов деление, для целочисленных деление нацело), /=
- % (для целочисленных типов остаток от деления), %=.
- == сравнение на равенство, != сравнение на неравенство, >, <, <=, <=.

Циклы и условный оператор

```
for(int i=0; i<5; i++)
{
    if(i%2==0)
        continue;
    Console.Write(i+” ”);
}
```

//continue – переход к следующей итерации

```
Int32 i=2;
while(true)
```

Массивы

- int[] ar = new int[] {6, 10, 2};
- int[] ar0 = new int[3]{10, 5, 1};
- int[] ar0;

  ar0= new int[] {6, 12, 20};
- int[] ar0 = new int[3];

  ar[0]=1;

  ar[1]=4;

  ar[2]=10;