Лекция 1-2

- ✔Платформа Microsoft.Net
- ✔ Структура программы
- ✔ Консольный ввод-вывод
- ✔ Основные элементы языка

Литература (учебники для студентов)

- Биллиг В.А. Основы программирования на C#. – М.:ИНТУИТ.РУ, 2006.
- Марченко А.Л. Основы программирования на C# 2.0 – М.:ИНТУИТ.РУ, 2006.
- Подбельский В.В. Язык C#. Базовый курс. – М.: Финансы и статистика, 2011.

Средства разработки Microsoft .NET

Visual Studio 2017 и выше (C#, C++, VB, F#)

Express Edition (бесплатная, доступна всем желающим)

http://www.microsoft.com/visualstudio/ru-ru/products/2010-editions/visual-csharp-express

Системы программирования

- Включают в себя:
- текстовый редактор, предназначенный для ввода и редактирования текста программы на языке программирования;
- компилятор
- средства отладки и запуска программ;
- библиотеки, содержащие многократно используемые элементы программ;
- справочная система и др.

Платформа программирования

- Включает в себя не только среду разработки для нескольких языков программирования Visual Studio. NET, но и другие встроенные средства: механизмы поддержки баз данных, электронной почты и коммерции и т.п.

Платформа программирования

- Компиляторы, входящие в состав платформы, переводят программу не в машинные коды, а в промежуточный язык *Microsoft Intermediate Language (MSIL или упрощенно IL), который* не содержит команд, зависящих от языка, операционной системы или типа компьютера.

Платформа программирования

- Программа на этом языке выполняется не самостоятельно, а под управлением системы, называемой общеязыковой средой выполнения (Common Language Runtime или CLR).

Платформа программирования

- При выполнении программы CLR вызывает JIT-компилятор («just in time» - «вовремя»), который переводит код с языка IL в машинные коды конкретного процессора, которые в свою очередь немедленно выполняются

Платформа Microsoft .NET

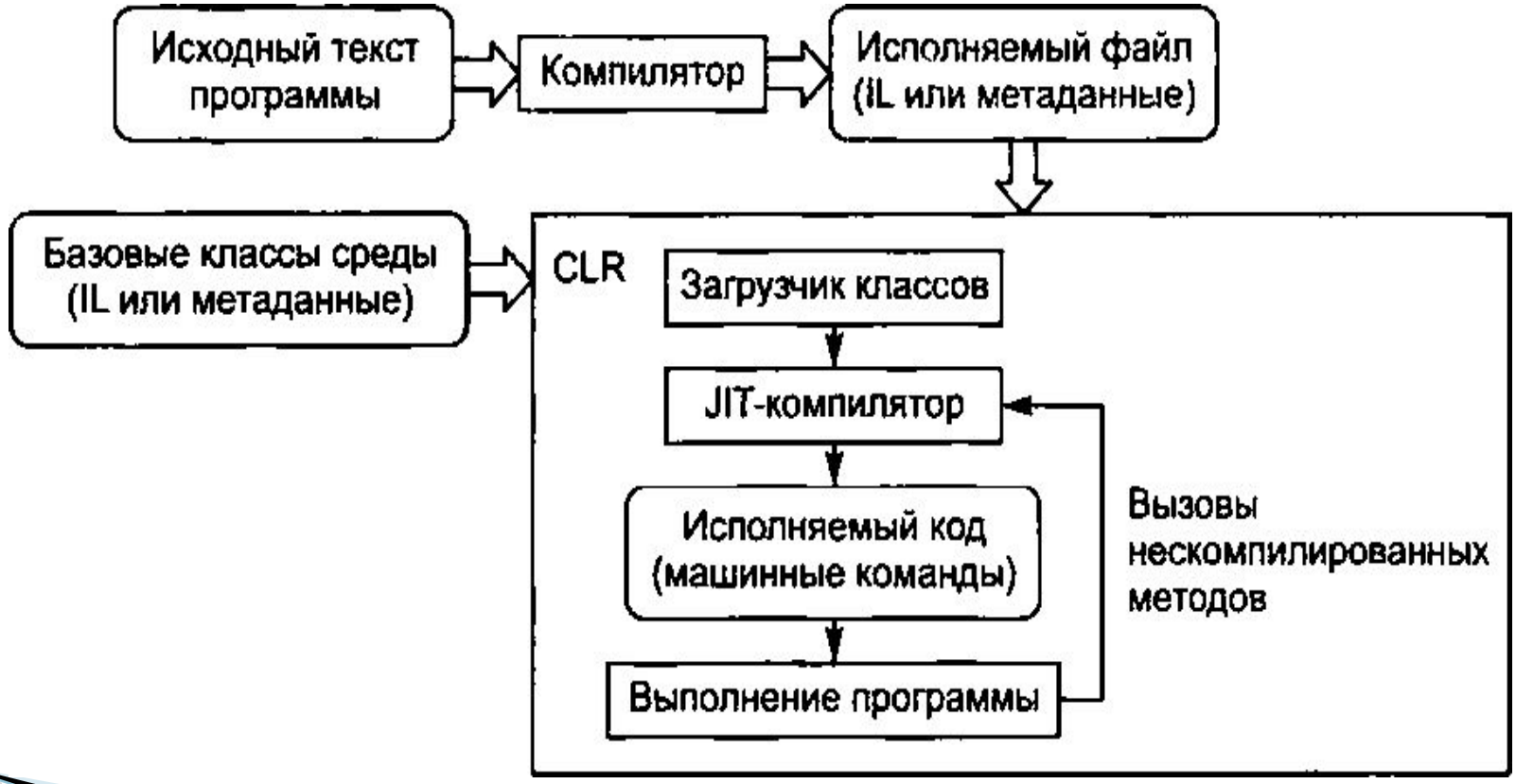
Достоинства

- объектно-ориентированная модель программирования
- автоматическое управление ресурсами
- упрощение развертывания приложений
- безопасность кода
- межъязыковое взаимодействие

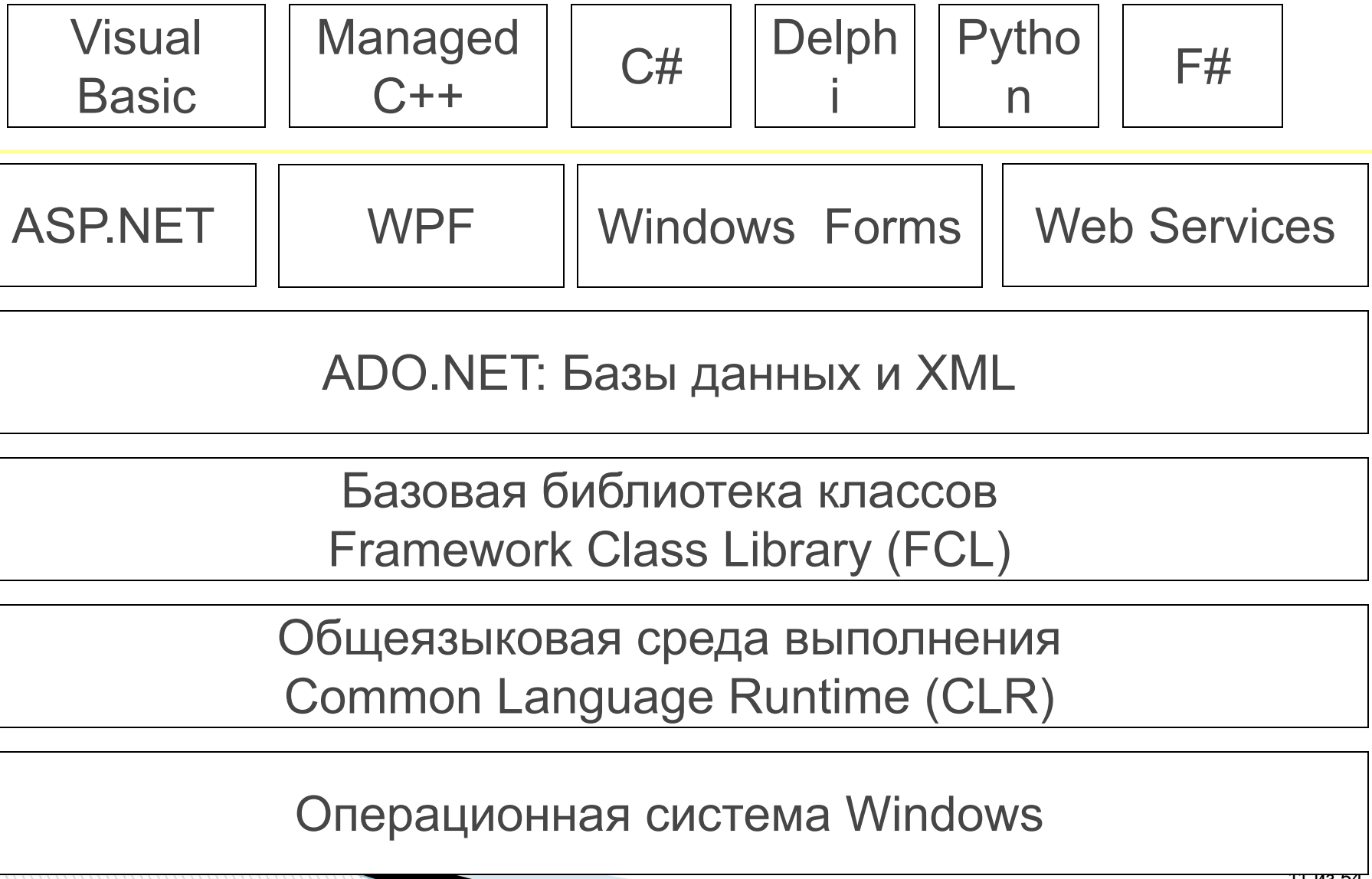
Недостатки

- привязка к ОС Windows (тем не менее есть версия для Linux: Mono)
- замедление при запуске и работе программы

Схема выполнения программы в .Net



Структура Microsoft .NET



Среда разработки Visual Studio 2017 Express

- Приложение (программа) в процессе разработки называется *проектом;*
- Среда Visual Studio.NET позволяет создавать проекты следующих типов:
- Windows-приложение (использует элементы интерфейса Windows такие как кнопки, формы, флажки и т.п.);
- консольное приложение осуществляет вывод на консоль, т.е. на экран и ввод с консоли, т.е. с клавиатуры.

Среда разработки Visual Studio 2017 Express

- Web-приложение (приложение, доступ к которому осуществляется через браузер, формирующее по запросу Web-страницу и отправляющее ее по сети клиенту);
- Web-сервис (компонент, методы которого могут вызываться через Internet);
- библиотека классов (объединяет классы, которые предназначены для использования в других приложениях).

Среда разработки Visual Studio 2017 Express

- Несколько проектов могут быть объединены в одно *решение (solution).*
- Решение создается автоматически при создании нового проекта.

Создание консольного приложения

- 1. В окне начальной страницы выбрать *Создать проект (New Project), после чего появится диалоговое окно;*
- 2. В списке типов приложений выбрать *Консольное приложение (Console Application);*

Обозреватель решений

- Это окно управления проектом.
- Если данного окна нет, то его подключение выполняется с помощью команды *Вид – Другие окна – Обозреватель решений (View - Solution Explorer).*

Обозреватель решений

- Окно Обозреватель решений содержит ресурсы:
- ссылки на библиотеку (System, System.Data, System.XML и т.п.);
- файл с исходным текстом класса *Program.cs;*
- информация о сборке *AssemblyInfo.cs*, являющейся результатом работы компилятора и содержит код на промежуточном языке и метаданные.

Файлы проекта

- файл проекта с расширением *.csproj;*
- файл решения с расширением *.sln;*
- файл с кодом класса с расширением *.cs.*

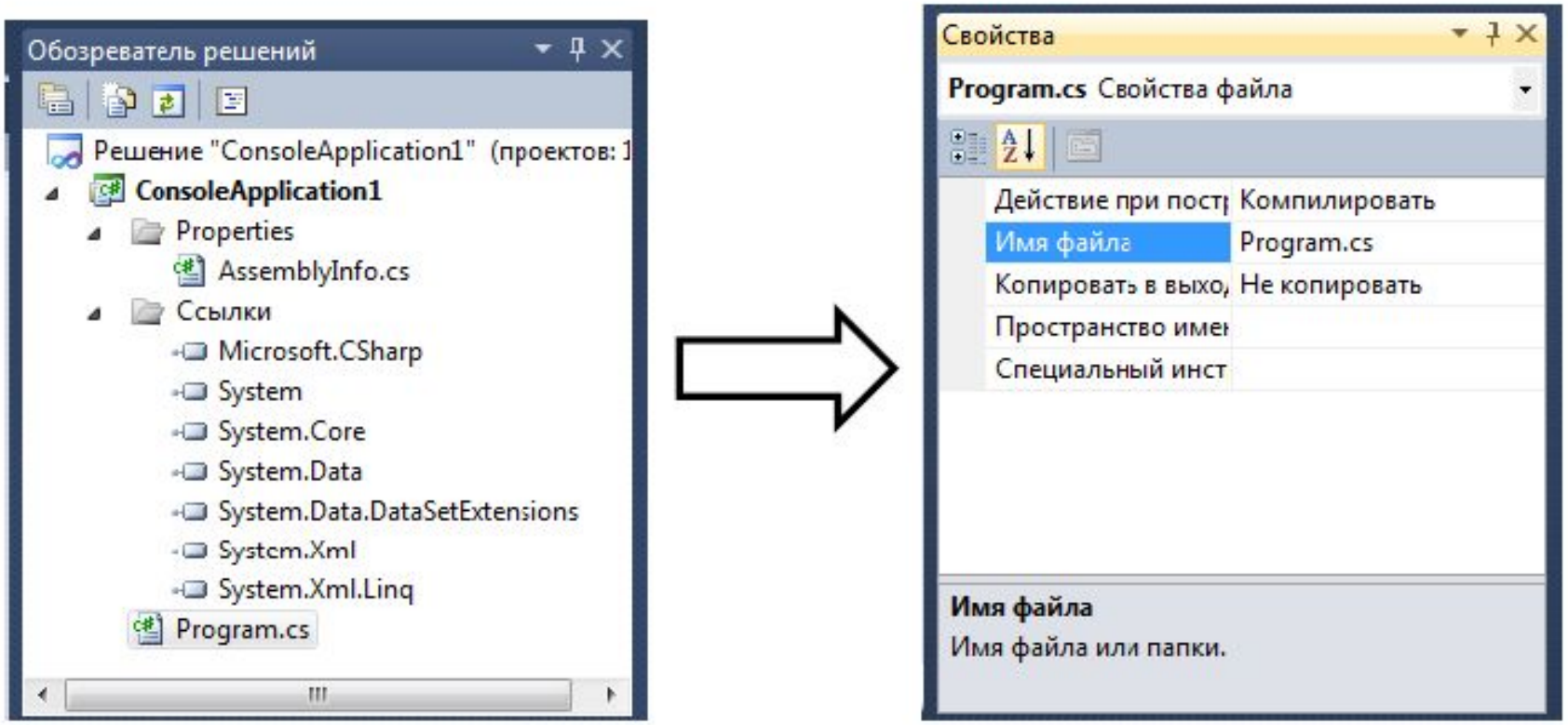
Окно свойств

- В данном окне отображаются характеристики выделенного элемента в окне *Обозреватель решений. Например, для того, чтобы изменить имя файла* Program.cs, необходимо:
- 1) выделить имя файла в окне *Обозреватель решений;*
- 2) изменить значение поля поле *Имя файла в окне Свойства;*
- 3) нажать Enter.

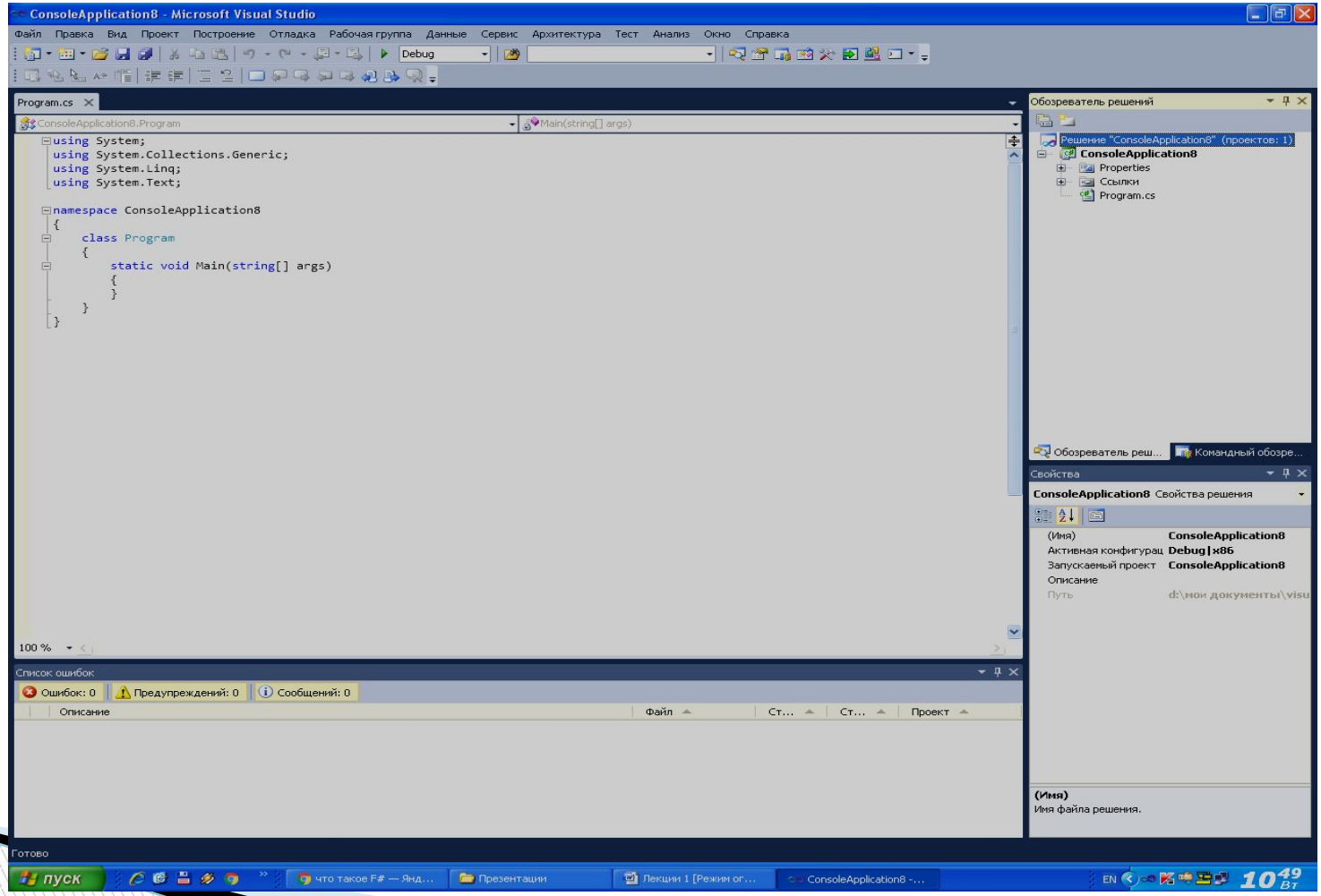
Окно свойств

- Если окно свойств отсутствует в окне среды, то нужно выполнить команду *Вид – Другие окна – Окно свойств.*

Окно свойств



Структура программы



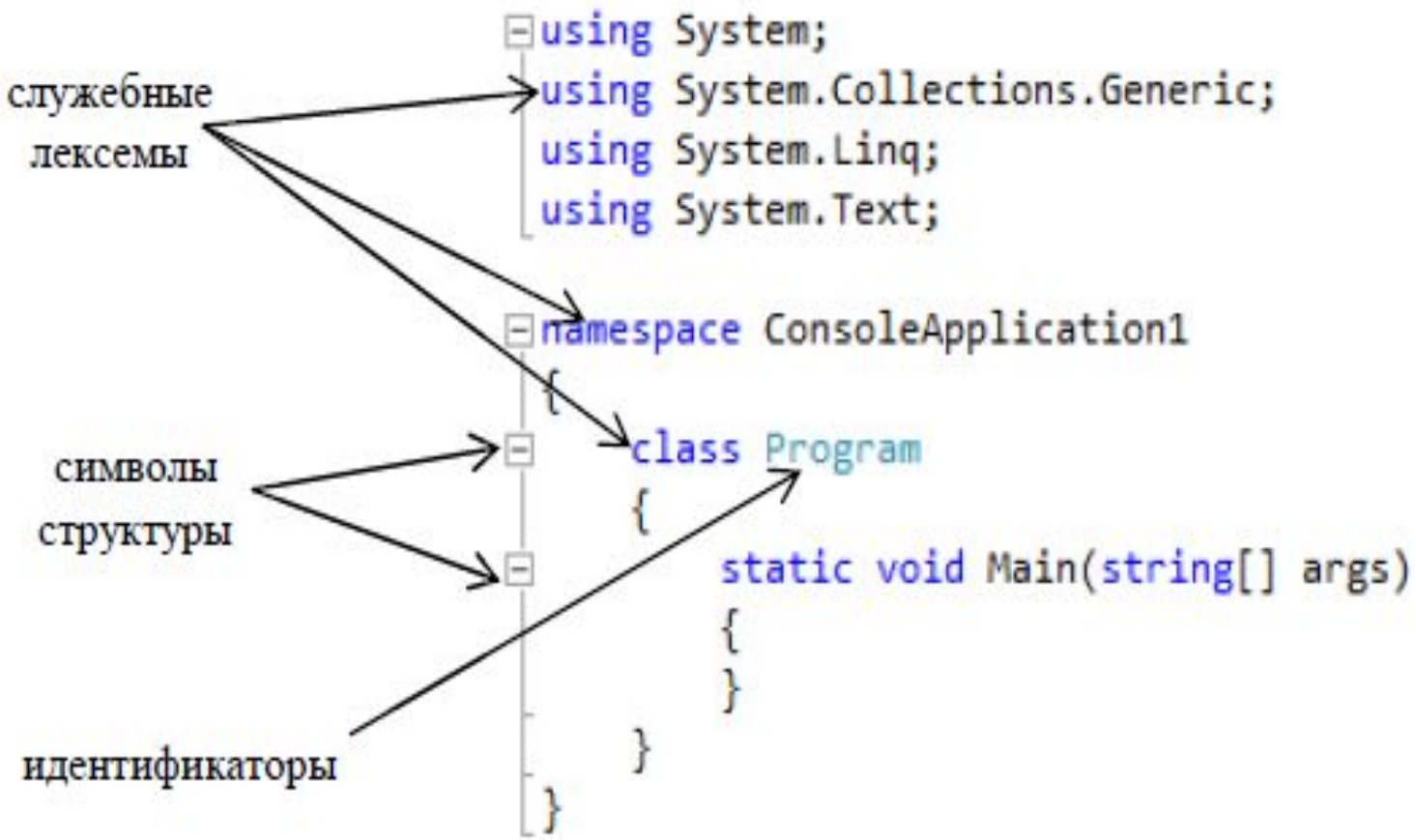
Структура программы

- Структура программы содержит следующие элементы:
- служебные (ключевые, зарезервированные) слова (лексемы), отображающиеся синим цветом, например, **using, namespace, class, static, void и др.;**

Структура программы

- комментарии различного типа, отображающиеся серым или темно-зеленым цветом;
- идентификаторы, отображающиеся голубым цветом, например, *Program;*
- остальные элементы, отображающиеся черным цветом.

Структура программы



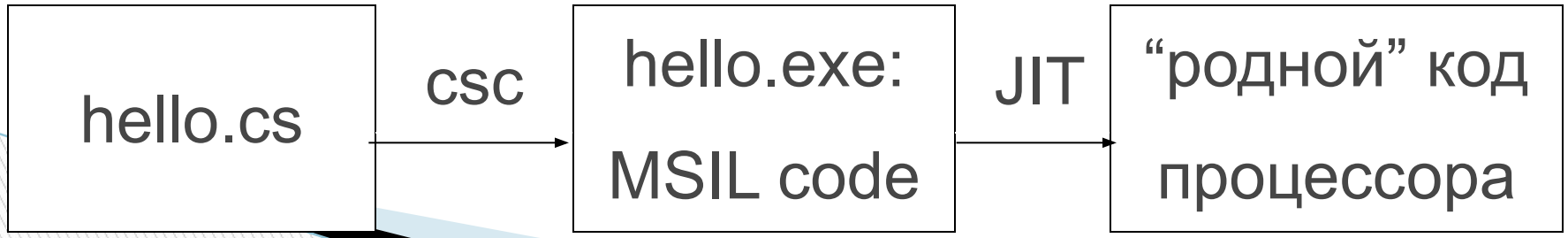
Структура программы на C#

- ✔ Любая программа - набор типов (классов)
- ✔ Все функции являются методами (элементами) классов. Формат вызова: имя_класса.имя_метода
- ✔ Все глобальные переменные должны быть статическими элементами внутри некоторых классов
- ✔ Точкой входа по умолчанию является статический метод Main

Первая программа на C#

```
using System;

public class Hello {
    public static void Main(String[] args)
    {
        Console.WriteLine("Hello, World!");
    }
}
```



Вторая программа на C# - a+b

```
using System;

public class Aplusb {
    public static void Main(String[] args)
    {
      int a,b;
   a = Convert.ToInt32(Console.ReadLine());
   b = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("{0} + {1} = {2}",a,b,a+b);
    }
}
```

Директива Using

Полное имя типа образуется из имени пространства имен и имени типа.
Пространство имен – это механизм, посредством которого поддерживается независимость используемых в каждой программе имен и исключается возможность их случайного взаимного влияния

Директива Using

✔ Полное имя типа образуется из имени пространства имен и имени типа.

```
public class Hello {
    public static void Main(String[] args)
    {
        System.Console.WriteLine("Hello, World!");
    }
}
```

✔ Директива using позволяет использовать сокращенные имена типов

```
using System;

public class Hello {
    public static void Main(String[] args)
    {
        Console.WriteLine("Hello, World!");
    }
}
```

Консольный вывод

✔ Для организации консольного вывода предназначены статические методы класса System.Console

```
Console.WriteLine("Hello, World!"); //перевод строки
Console.Write("Hello, World!");
```

✔ Методы Write и WriteLine могут иметь несколько параметров, при этом первый параметр – форматная строка

```
Console.WriteLine("{0}, {1}{2}","Hello","World","!");
Console.WriteLine (" {0} + {1} = {2} ", 3, 5, 8);
```

```
На экране:
Hello, World!
3 + 5 = 8
```

Консольный вывод

```
int i = 3;


Console.WriteLine( "i = " + i );

Console.WriteLine ( “i=” + i.ToString());
```

На экране:

i=3;

Форматирование при выводе

✔ Общий вид строки форматирования

```
{N,M:F<R>}
```

Количество выводимых разрядов

Формат вывода

Ширина поля

Номер параметра (начинаются с нуля)

✔ Форматы вывода

```
C - форматирование числа как денежной суммы
D - Целое число
E - Вещественное число в виде 1e+3
F - Вещественное число в виде 123.456
G - Вещественное число в наиболее компактном формате
N - Вещественное число в виде 123,456,789.5
X - Целое число в шестнадцатеричном виде
```

Форматирование при выводе

```
static void Main(string[] args)
        {
            double a = 35.983974;
            Console.WriteLine("{0,5:F2}", a);
            Console.ReadLine();
        }
```

На экране : 35,98

Консольный ввод

...

✔ Ввод строки

```
string s;
s = Console.ReadLine();
s = Console.Read();
```

✔ Преобразование строки в число: 2 способа

```
int a = Convert.ToInt32(s);
int b = Int32.Parse(s);
int a = Convert.ToInt32(Console.ReadLine());
```

✔ Преобразование числа в строку: 2 способа

```
string s1 = Convert.ToString(a);
string s2 = a.ToString();
```

Компоненты языка

- Алфавит - это фиксированный для данного языка набор основных символов, т.е. "букв алфавита", которые используются для написания текстов на языке. Любые другие символы - недопустимы.
- Синтаксис - система правил, определяющих допустимые конструкции из букв алфавита. Для каждой последовательности символов синтаксис позволяет ответить на вопрос, является ли она текстом на данном языке или нет.

Компоненты языка

- Семантика - система правил истолкования отдельных языковых конструкций, позволяющих (при заданных исходных данных) однозначно воспроизвести процесс обработки данных по заданной программе.
- Лексема – минимальная единица языка, имеющая собственный смысл (семантику)

Алфавит языка C#

- Буквы – произвольные буквенные Unicode, знак подчеркивания. Прописные и строчные буквы различаются в именах.
- Цифры (0…9)
- Специальные знаки “ { } , | [ ] ( ) + - * / % \ ; ‘ : ? < = > ! & ~ ^ . # @
- Разделители (пробел, табуляция, перевод строки)

Комментарии

Могут быть в любом месте, где есть разделитель

Виды комментариев:

- однострочные // комментарий
- многострочные /* длинные */

Имена

Имя (идентификатор) служит для обозначения каких-либо объектов.

Следует отметить, что в качестве имени нельзя использовать служебные слова.

Имя – это последовательность букв и цифр, начинающаяся с буквы. Пробелы в идентификаторах не допускаются. Различаются строчные и прописные буквы.

- Void служебное слово -
- Void1 +
- Program +
- My_program +
- My#Program -
- 1file -
- File_1 +
- File1 +
- Max и max это разные идентификаторы

Константы

- Целые:
  - десятичные 123, 0, 98
  - восьмеричные 01, 015
  - шестнадцатиричные 0xA1, 0X00FF
- Вещественные 5.8, .2e-3
- Символьные ‘A’, ‘\xF’, ‘\n’, ‘\a’, ‘\\’
- Строковые “привет”, “1 \n 2”, @”C:\\TEMP”

- $28,567=0,28567* 10^2$
- МАНТИССА 2 – ПОРЯДОК
- ‘А’ 1 байт в ОП
- “А” 2 байт
- Строковые константы завершаются спец символом-признаком конца строки

Управляющая (escape-последовательность)

- *Это* определенный символ, предваряемый обратной косой чертой. Используется для представления:
- кодов, не имеющих графического изображения (например, \n — переход в новую строку);

Управляющая (escape-последовательность)

- символов, имеющих специальное значение в строковых и символьных литералах, например, апострофа.

Примеры

- \a Звуковой сигнал
- \f Перевод страницы
- \n Перевод строки
- \’ Апостроф
- \” Кавычка
- \0 Нуль-символ
- \\ Обратный слэш

- если внутри строки требуется использовать кавычку, ее предваряют косой чертой, по которой компилятор отличает ее от кавычки, ограничивающей строку:
- "Издательский дом \"Питер\""

Дословные литералы

- Используются для строковых литералов с управляющими символами
- "C: \\app\\bi n\\debug\\a .exe"
- @"C:\app\bin\debug\a.exe"