Первый взгляд на платформу .Net

Программист пишет программу, компьютер её выполняет. Программа создается на языке, понятном человеку, а компьютер умеет выполнять только программы в машинных кодах. Совокупность средств, с помощью которых программы пишут, корректируют, преобразуют в машинные коды, отлаживают и запускают, называют *средой разработки* или *оболочкой.*

Среда разработки обычно содержит:

- Текстовый редактор (для ввода и корректировки текста программы);

- компилятор (с помощью которого программа переводится в машинные коды);
- средства отладки и запуска программы;
- общие библиотеки;
- справочные материалы.

Под платформой понимается нечто большее, чем среда разработки для одного языка. Платформа .NET включает не только среду разработки для нескольких языков программирования, но и механизмы поддержки баз данных, электронной почты и коммерции.

Платформа позволяет успешно решать основные задачи при создании программ:

1) Переносимость – возможность выполнения на различных типах компьютеров;
2) Безопасность – невозможность несанкционированных действий;
3) Надежность – способность выполнять необходимые функции а предопределенных условиях;
4) Межъязыковое взаимодействие.

Для обеспечения переносимости компиляторы, входящие в состав платформы, переводят программу не в машинные коды, а в промежуточный язык MSIL или просто IL.

(Microsoft Intermediate Language), который не содержит команд, зависящих от языка, операционной системы и типа компьютера. Программа на этом языке выполняется не самостоятельно, а под управлением системы, которая называется общеязыковой средой выполнения CLR(Common Language Runtime). Среда CLR может быть реализована для любой операционной системы. При выполнении программы CLR вызывает так называемый JIT – компилятор, переводящий код с языка IL в машинные команды конкретного процессора. JIT означает «just in time», то есть компилируются только те части программы, которые требуются выполнить в данный момент.

Каждая часть программы компилируется один раз и сохраняется в кэше для дальнейшего использования.

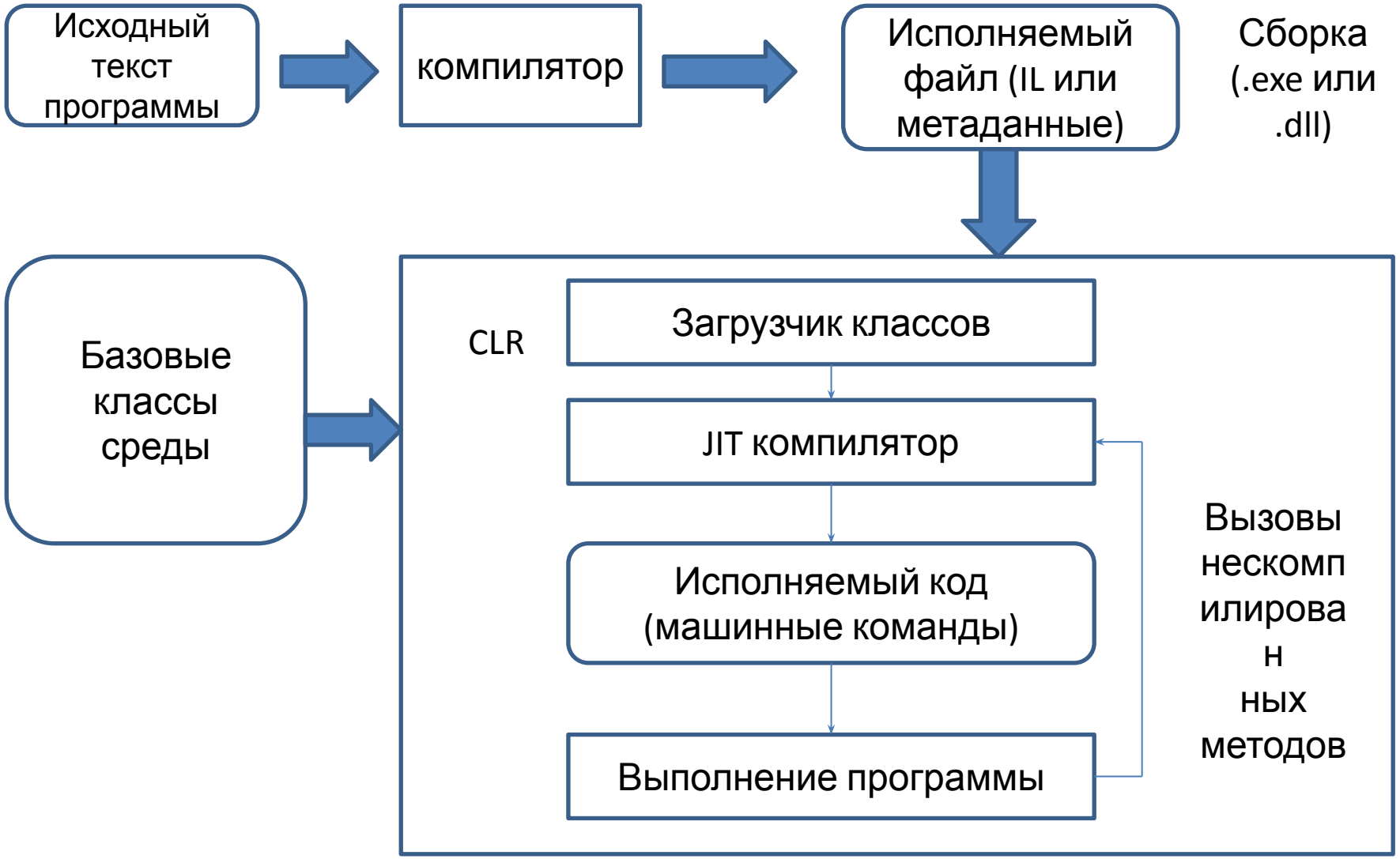


Схема выполнения программы в .NET

Компилятор в качестве результата своего выполнения создает так называемую сборку – файл с расширением exe или dll, который содержит код на языке IL. Метаданные представляют собой сведения об объектах, используемых в программе, а также сведения о самой сборке. Они позволяют организовать межъязыковое взаимодействие, обеспечивают безопасность и облегчают развертывание приложений.

Д.З. Структура платформы .NET.

Объектно-ориентированное программирование.

Основными понятиями ООП являются полиморфизм, инкапсуляция, наследование и класс.

Класс – это обобщенное понятие, определяющее характеристики и поведение некоторого множества конкретных объектов этого класса, называемых *экземплярами класса.* Обычно класс содержит данные, задающие свойства объектов класса, и функции, определяющие их поведение. Например, класс «точка на плоскости». Его данные-координата X, координата Y, его функции-вывод на экран, определение квадранта плоскости.

Либо класс Кнопка. Его данные - высота, ширина, цвет и т.д. Его функции – Нажатие на кнопку.

После описания класса, создается объект данного класса. Например, точка А, точка В, кнопка 1 и т.д.

Свойство объекта скрывать внутренние структуры данных называется *инкапсуляцией.*

Позволяет изменить реализацию объекта без модификации основной части программы. Осуществляется в основном при помощи ключей доступа public, protected, private, static.

Наследование является мощнейшим инструментом ООП и применяется для следующих целей:

- Исключение из программы повторяющихся фрагментов кода;
- Упрощение модификации программы;
- Можно использовать объекты, исходный код которых недоступен для изменений.

Класс, на основании которого строится новый класс, называется родительским или базовым, а унаследованный - потомком или производным классом.

Под полиморфизмом понимается возможность во время выполнения программы с помощью одного и того же имени выполнять разные действия или обращаться к объектам разного типа. Например, класс точка может иметь координаты X и Y как целого, так и вещественного типа. И можно сделать так, чтобы класс по-разному обрабатывал объекты классов.

Создание проекта.

Среда Visual Studio.NET работает на платформе Windows и ориентирована на создание Windows- и веб-приложений, однако разработчики предусмотрели работу и с консольными приложениями. Для создания проекта следует после запуска среды в главном меню выбрать команду File/New/Project. В левой части открывшегося диалогового окна нужно выбрать пункт Visual C# Projects, в правой – пункт Console Application. В поле Name можно ввести имя проекта, а в поле Location – место его сохранения.

Заготовка консольной программы.

```
Using System;
namespace ConsoleApplication1
{
class Class1
{
static void Main(string[] args)
{
///Add code to start application here
}
}
}
```

Директива using System разрешает использовать имена стандартных классов из пространства имен System. Ключевое слово namespace создает для проекта собственное пространство имен, по умолчанию ConsoleApplication. Строки, начинающиеся с 2х или 3х косых черт, являются комментариями. В новой программе создается один класс, поэтому его имя Class1. В фигурных скобках список элементов класса (его данные и методы). В нашем случае только один метод Main. Каждое приложение должно содержать 1 метод с именем Main- с него начинается выполнение программы.

Среда заботливо поместила внутрь метода комментарий //Add code…..

Удаляем спокойно комментарии, и напишем, например,

Console.WriteLine(“Hello World!!!!”);

Здесь Console – это имя стандартного класса из пространства имен System. Его метод WriteLine выводит на экран заданный в кавычках текст.

Запустить программу – нажать клавишу F5(или выбрать в меню команду Debug/Start) либо Ctrl+F5 (Debug/Start Without Debugging).

Основные понятия языка.

Алфавит C# включает:

-буквы (латинские и национальных алфавитов);

-цифры;

-специальные символы (+,-,_,{,},<,>)

-пробельные символы(табуляция и пробел)

Лексема – это минимальная единица языка, имеющая самостоятельный смысл. Существуют след. виды лексем: имена(идентификаторы), ключевые слова, знаки операций, разделители, литералы(константы).

Типы данных.

Данные, с которыми работает программа, хранятся в оперативной памяти. Естественно, что компилятору необходимо точно знать, сколько места они занимают, как именно закодированы и какие действия с ними можно выполнять. Все это задается при описании данных при помощи типа. Тип данных однозначно определяет внутренне представление данных и допустимые действия над данными.

Различают типы данных простые и структурированные; статические и динамические; встроенные и определенные программистом.

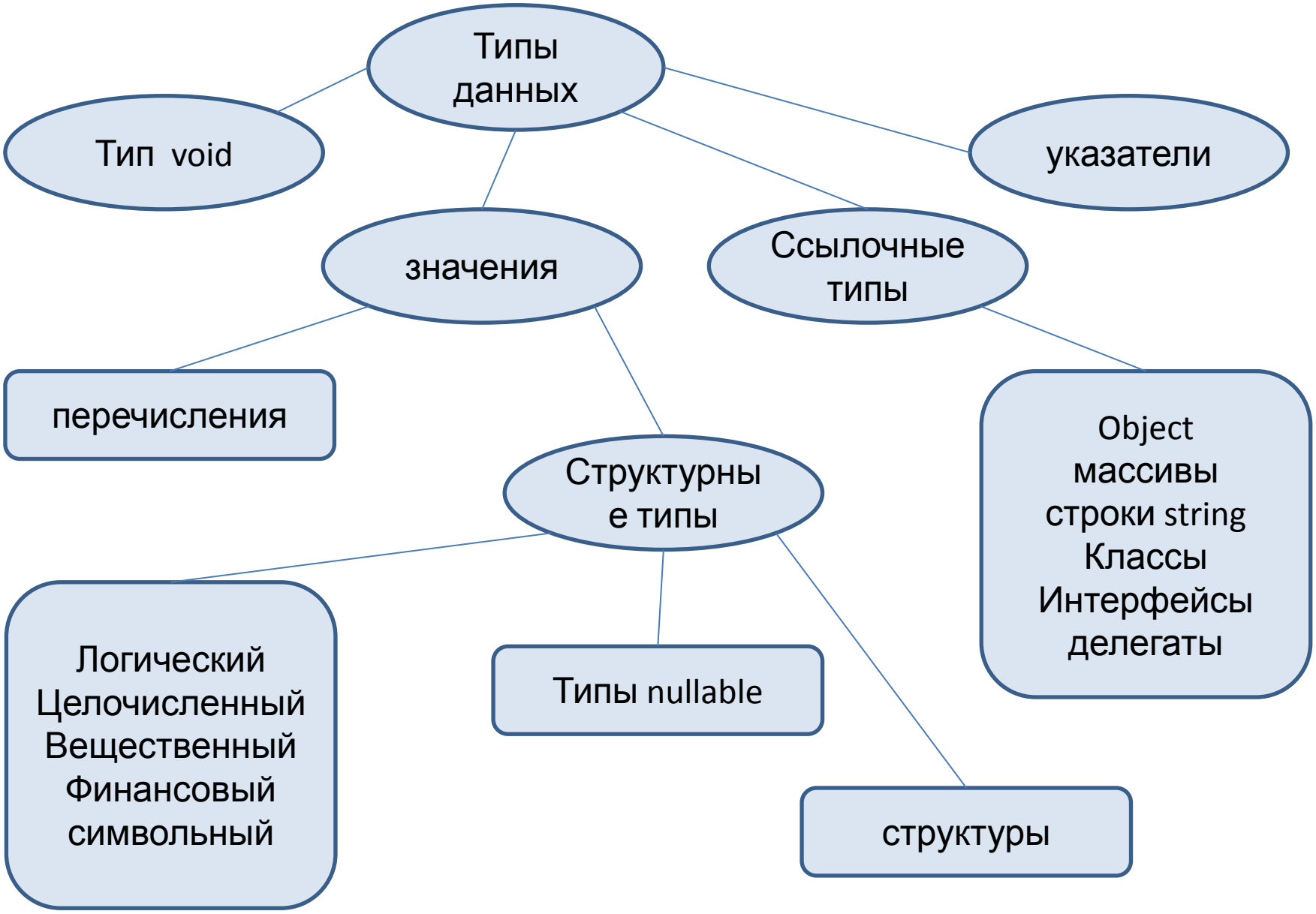
Встроенные типы не требуют предварительного определения и имеют следующие ключевые

Встроенные типы C#

Название	Ключевое слово	Диапазон значений	Размер битов
Логический тип	bool	true, false	
Целые типы	sbyte	-128…127	8
	byte	0…255	8
	short	-32768…32767	16
	int	$-2*10^{9}$…$2*10^{9}$	32
	long	$-9*10^{18}$…$9*10^{18}$	64
Символьный тип	char	U+0000…U+ffff	16
Вещественные	float	$1,5*10^{-45}$…$3.4*10^{38}$	32
	double	$5.0*10^{-324}$…$1.7*10^{308}$	64
Финансовый	decimal	$1.0*10^{-28}$…$7.9*10^{28}$	128
Строковый тип	string	Длина ограничена объемом доступной памяти	
Тип объект	object	Можно хранить все что угодно	

Типы-значения и ссылочные типы.

Чаще всего типы C# разделяют по способу хранения элементов на типы-значения и ссылочные типы. Элементы типов-значений или значимых типов представляют собой просто последовательность битов в памяти, необходимый объем которой выделяет компилятор. Величина ссылочного типа хранит не сами данные, а ссылку на них (адрес, по которому расположены данные). Сами данные хранятся в хипе.



Классификация типов данных C# по способу хранения

Разницу между величинами ссылочного и значимого типов можно рассмотреть на примере проверки на равенство. Величины значимого типа равны, если равны их значения. Величины ссылочного типа равны, если они ссылаются на одни и те же данные (b и c равны, но a и b не равны даже при одинаковых значениях).