Основы программирования

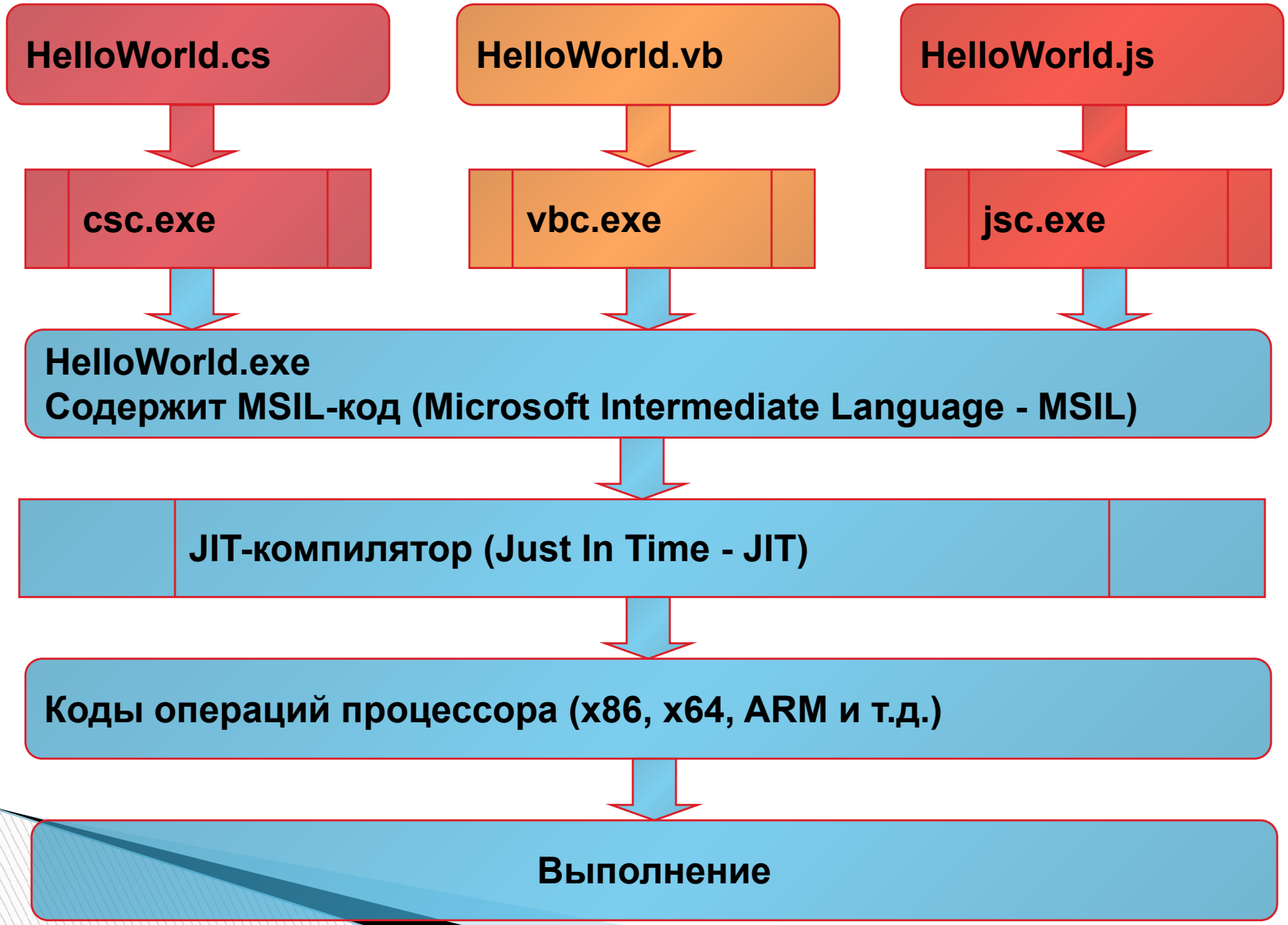
Виденин Сергей Александрович

ЛЕКЦИЯ №1

Типы данных

Алгоритмические структуры

MSIL-компиляция



JIT компиляция и CLR

- Программный код компилируется в промежуточный код (Intermediate Language – IL, MSIL, CIL)
- CLR – не интерпретатор. Компиляция происходит 1 раз. Повторно не компилируется, а используется уже откомпилированный код

- Более медленный старт и работа приложения

- Экономия памяти
  - Код на IL обычно занимает меньше места
  - Компилируется только тот код, который выполняется
- JIT компилятор получает высоко оптимизированный код (заточенный под конкретную аппаратную модель)
- CLR отслеживает частоту вызова и может производить оптимизацию налету

Общеязыковая среда выполнения

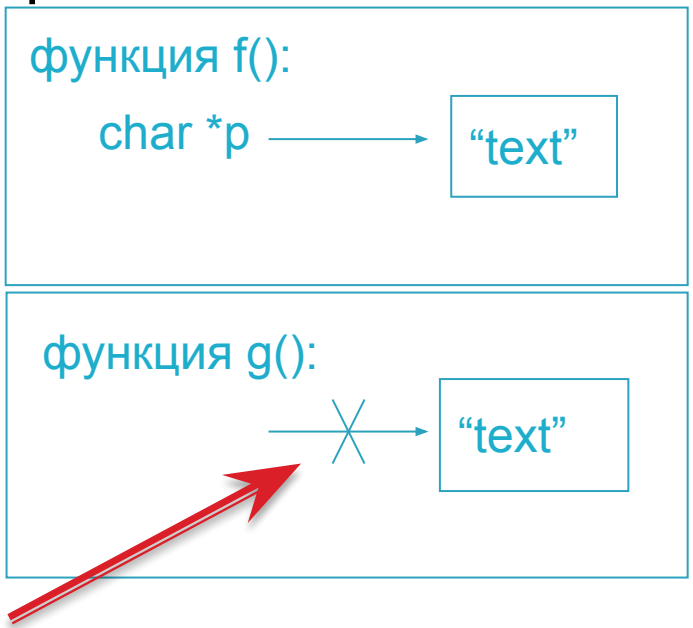
- **Common Language Runtime - CLR**
- Виртуальная исполняющая среда
- Отвечает за:
  - Загрузку сборок
  - Just In Time компиляцию
  - Управление памятью
  - Управление безопасностью

Управление памятью

- Автоматическая сборка мусора

```
// Утечка памяти в C
char *f(int a)
{
    char *p =
(char)malloc(…);
    …
    return p;
}
…
void g(){ f(1); }
```

Утечка памяти



Сборщик мусора (Garbage Collector - GC) отслеживает ссылки на объекты. Он обнаружит, что на область памяти p больше нет ссылок и освободит эту область.

- CLR может перенести часто используемые объекты для оптимизации доступа к страницам памяти

Hello, World!

```
using System;

namespace HelloWorld
{
  class HelloWorld
  {
    /// <summary> Entry point </summary>
    static void Main(string[] args)
    {
      Console.WriteLine("Hello, C# World!");

    }  // end of Main()

  }  // end of HelloWorld

}  // namespace HelloWorld
```

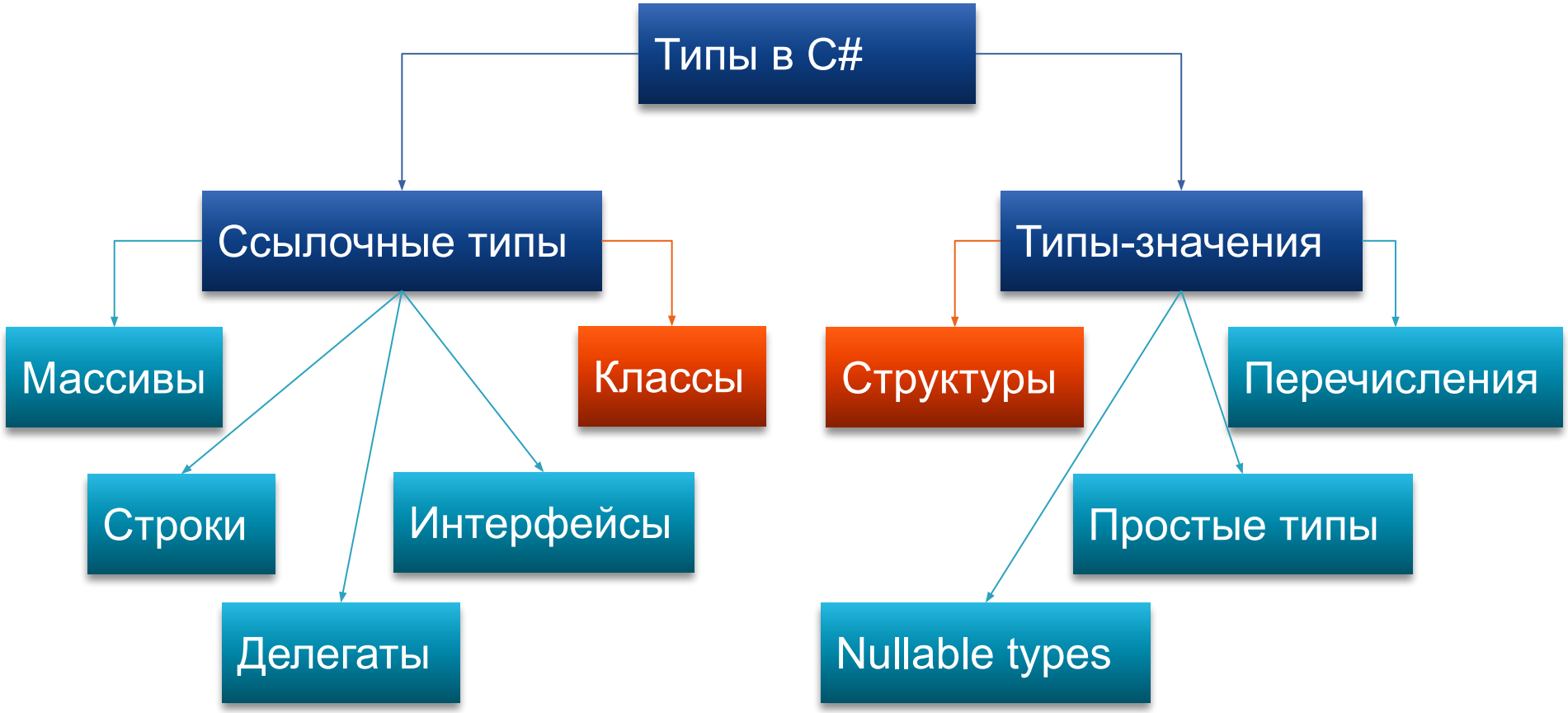
Замечания

- Пространство имен
  - объединяет группу семантически связанных между собой типов
  - Позволяет отделять типы с одинаковыми названиями
- Варианты метода Main
  - `static void Main(){…}`
  - `static int Main(){… return 0; }`
  - `static void Main(string[] args){…}`
  - `static int Main(string[] args){… return 0; }`
- ***using*** позволяет сократить полное название типа (System.Console). Как бы объединяет пространства имен* с текущим (*или тип в C# 6)
- .NET использует Unicode.
  - Название типов можно заводить и на русском языке (но не рекомендуется)
- Языки для .NET чувствительны к регистру
  - Main() и main() разные методы
- Вывод на консоль: *System.Console.WriteLine(“текст”)*
- Чтение данных с консоли: *string s = System.Console.ReadLine()*

Строгая типизация в C#

- Каждая переменная и экземпляр объекта в системе относится к четко определенному типу !!!
- Все типы происходят от одного корневого предка – типа **object** *

Типы в C#



* - условная схема, поскольку все ссылочные типы (кроме интерфейсов) – классы, все типы значения - структуры

Самые важные типы

- **int** – 32-битное целое (System.Int32)
- **bool** – логический тип (System.Boolean). Значения только **true** и **false**
- **float**, **double** – вещественные типы (System.Single и System.Double)
- **char** – символьный тип Unicode
- **string** – строка текста (Unicode)
- **DateTime** – дата и время

Простые целые типы

Тип (C#)	Полное название типа	Диапазон	Описание	Размер (бит)
sbyte	**System.Sbyte**	-128 до 127	Знаковое целое	8
byte	**System.Byte**	0 до 255	Без знаковое целое	8
short	**System.Int16**	-32 768 до 32 767	Знаковое целое	16
ushort	**System.UInt16**	0 до 65 535	Без знаковое целое	16
int	**System.Int32**	-2 147 483 648 до 2 147 483 647	Знаковое целое	32
uint	**System.UInt32**	0 до 4 294 967 295	Без знаковое целое	32
long	**System.Int64**	$-9 * 10^{19}$ до $9 * 10^{19}07$	Знаковое целое	64
ulong	**System.UInt64**	0 до $18 * 10^{19}$	Без знаковое целое	64
char	**System.Char**	U+0000 до U+ffff	Символ в Unicode	16

* Все типы – типы значения

Вещественные типы

Тип (C#)	Полное название типа	Диапазон	Точность	Размер (бит)
float	**System.Single**	$\pm 1.5 * 10^{-45}$ до $\pm 3.4 * 10^{38}$	7 знаков	32
double	**System.Double**	$\pm 5.0 * 10^{-324}$ до $\pm 1.7 * 10^{-308}$	15-16 знаков	64
decimal **	**System.Decimal**	$(-7.9 \times 10^{28}$ до $7.9 \times 10^{28}) / (10^{0}$ до $^{28})$	28-29 знаков	128

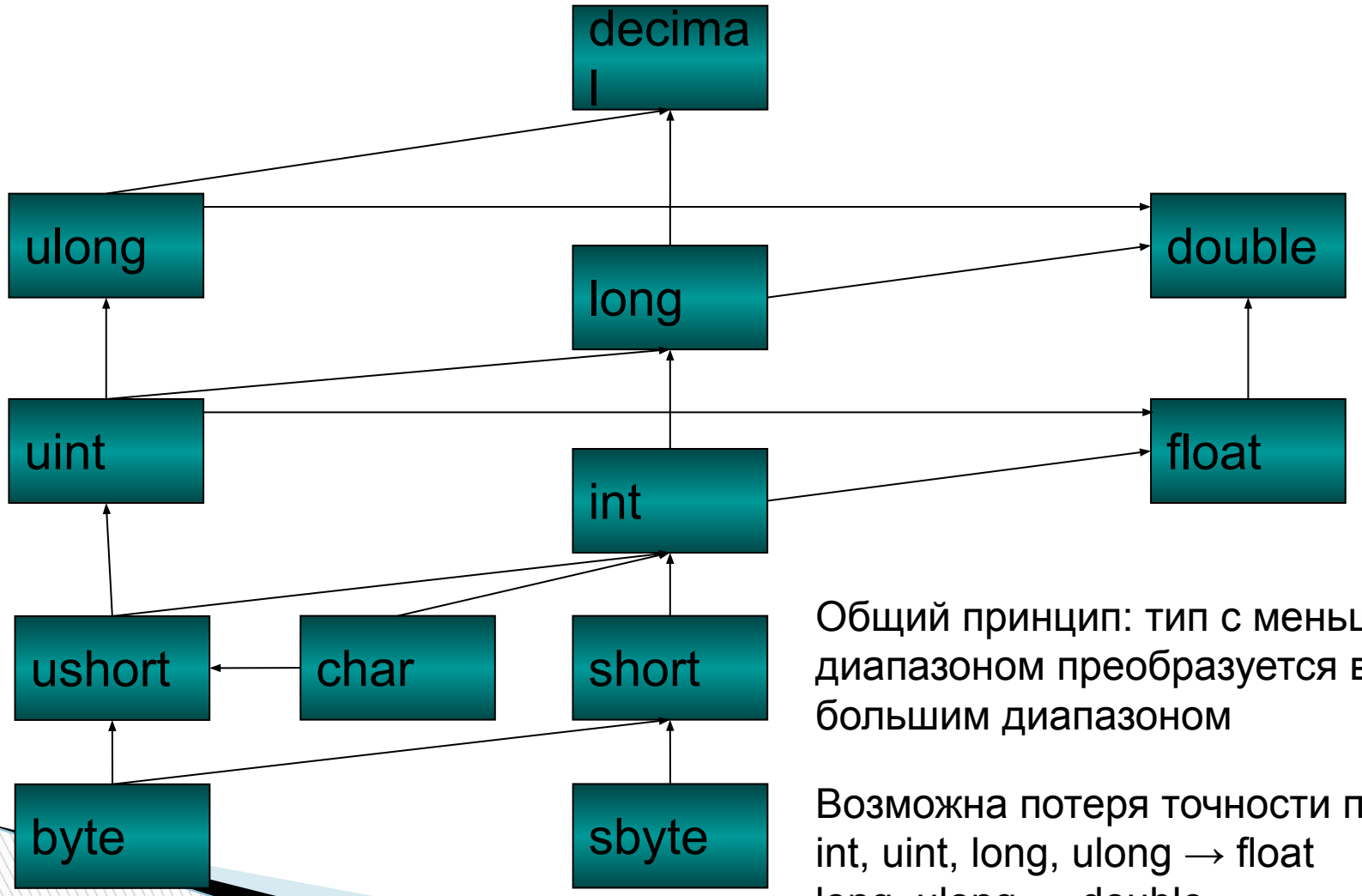
** Не имеет аппаратной поддержки
Всегда проверяет диапазон

* Все типы – типы значения

Важные типы

- **bool** – логический тип
  - System.Boolean
  - Значения только **true** и **false**
  - Тип значение
- **string** – строка текста (Unicode)
  - System.String
  - неограниченной длины
  - Ссылочный тип
- **DateTime** – дата и время
  - Структура (тип-значение)
  - От 1 января 1 года до 31 декабря 9999 года
  - Точность 100 нс
  - Работает с временными зонами

Неявное приведение типов



Общий принцип: тип с меньшим диапазоном преобразуется в тип с большим диапазоном

Возможна потеря точности при:
int, uint, long, ulong → float
long, ulong → double.

Явное приведение типов

- Синтаксис:
  - (*type*)*expression*
- Пример:

```
double d = 5.5;
int i = (int)d;
```

- Применяется при преобразованиях типов с возможной потерей точности
- При «зашкаливании» результат определяется контекстом

Контекст проверки вычисления

- 2 контекста
  - **checked** – проверяет на переполнение
  - **unchecked** – не проверяет

- Устанавливаются
  - Глобально (опции проекта)
  - Локально (блоки **checked** и **unchecked**)
  - Не распространяется на вызовы функций

- По умолчания проверка выключена.
  - Однако, если значение выражения может быть вычислено в процессе компиляции, то употребляется контекст checked
  - byte h = (byte) (255 + 100); // вызовет ошибку в процессе компиляции

Типы данных по умолчанию

- Если значение целое и оно помещается в int – то подразумевается int
  - 5 – тип int Пример: int i= 45;
- Если значение вещественное – то подразумевается double
  - 5.6 – тип double Пример: double d= 12.277;
- Для обозначения конкретных типов служат “суффиксы”
  - 5l – long Пример: long l = 5l;
  - 5.4f – float Пример: float f = 5f;
  - 4m – decimal Пример: decimal d = 0m;
- Шестнадцатеричное число 0xЧИСЛО
  - 0x0099 Пример: int i= 0x1234FFFF;
- Восьмеричное число 0ЧИСЛО
  - 06789 Пример: int i= 05777;

- Выражения. Тип выражения определяется в порядке приоритета:
  - Если в выражении присутствует decimal, то результат операции – decimal
  - Если присутствует вещественное число, то результат операции – double
  - ulong, если присутствует тип ulong
  - long, если присутствует тип long
  - Результат операции с целыми числами – int

Перечисление

- Служит для кодирования возможных значений или магических чисел
- enum MyEnum {
  - Monday,
  - Thursday,
  - …
- }
- enum OneMoreEnum {
  - Monday = 2,
  - Thursday,
  - Среда = 4,
  - …}

enum Имя [:базовый целочисленный тип]
{
Имя1 [=значение1]
[, … ИмяN [=значениеN]]
}

- По умолчанию “наследуются” от int, но могут “наследоваться” от другого целочисленного типа
- Если не указано значение, то для первого по умолчанию – 0, для каждого последующего – предыдущее +1
- Объявление и использование:

  OneMoreEnum my = OneMoreEnum. Среда ;
- Возможно приведение типов: int I = (int)my; int j = (int)OneMoreEnum. Среда;