# **ии­­­­№ 1 Основы CLR и .NET. Типы. Массивы, кортежи и строки**

## Задание

1. **Типы**
   1. Определите переменные всех возможных примитивных типов С# и проинициализируйте их. **Осуществите ввод и вывод их значений используя консоль.** ( <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types>, <https://docs.microsoft.com/en-us/dotnet/api/system.console?view=netframework-4.8>)
   2. Выполните 5 операций явного и 5 неявного приведения. (<https://docs.microsoft.com/en-us/dotnet/csharp/languagereference/language-specification/conversions#implicit-conversions> ) **Изучите возможности класса *Convert*.**
   3. Выполните упаковку и распаковку **значимых** типов.
   4. Продемонстрируйте работу с неявно типизированной переменной.
   5. Продемонстрируйте пример работы с ***Nullable*** переменной (<https://docs.microsoft.com/en-us/dotnet/api/system.nullable-1?view=netframework-4.8>

https://docs.microsoft.com/en-us/dotnet/api/system.nullable-1?view=netcore-3.1).

* 1. . Определите переменную типа *var* и присвойте ей любое значение. Затем следующей инструкцией присвойте ей значение другого типа. Объясните причину ошибки.

1. **Строки**
   1. Объявите строковые литералы. Сравните их.
   2. Создайте три строки на основе *String*. Выполните: сцепление, копирование, выделение подстроки, разделение строки на слова, вставки подстроки в заданную позицию, удаление заданной подстроки. (<https://docs.microsoft.com/ru-ru/dotnet/api/system.string?view=netcore-3.1> ) Продемонстрируйте интерполирование строк.
   3. Создайте пустую и *null* строку. Продемонстрируйте использование метода *string.IsNullOrEmpty*. Продемонстрируйте что еще можно выполнить с такими строками
   4. Создайте строку на основе *StringBuilder*. Удалите определенные позиции и добавьте новые символы в начало и конец строки. (<https://docs.microsoft.com/ru-ru/dotnet/api/system.text.stringbuilder?view=netcore-3.1> )
2. **Массивы (** <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/> **)**
   1. Создайте целый двумерный массив и выведите его на консоль в отформатированном виде (матрица).
   2. Создайте **одномерный** массив строк. Выведите на консоль его содержимое, длину массива. Поменяйте произвольный элемент (пользователь определяет позицию и значение).
   3. Создайте **ступечатый** (не выровненный) массив вещественных чисел с 3-мя строками, в каждой из которых 2, 3 и 4 столбцов соответственно. Значения массива введите с консоли.
   4. Создайте **неявно типизированные переменные** для хранения массива и строки.
3. **Кортежи (** <https://docs.microsoft.com/en-us/dotnet/csharp/tuples> **)**
   1. Задайте кортеж из 5 элементов с типами *int, string, char, string, ulong*.
   2. Выведите кортеж на консоль целиком и выборочно ( например 1, 3, 4 элементы)
   3. Выполните распаковку кортежа в переменные. Продемонстрируйте различные способы распаковки кортежа. Продемонстрируйте использование переменной ( \_ ). (доступно начиная с C#7.3)
   4. Сравните два кортежа.
4. Создайте **локальную функцию** в main и вызовите ее (<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/local-functions> ) . Формальные параметры функции – массив целых и строка. Функция должна вернуть кортеж, содержащий: максимальный и минимальный элементы массива, сумму элементов массива и первую букву строки .
5. Работа с *checked*/*unchecked*: ( <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/checked-and-unchecked> )
   1. Определите две **локальные** функции.
   2. Разместите в одной из них блок *checked*, в котором определите переменную типа *int* с максимальным возможным значением этого типа. Во второй функции определите блок *unchecked* с таким же содержимым.
   3. Вызовите две функции. Проанализируйте результат.
6. Загрузите проект в свой репозиторий на GitHub.
7. Подготовить ответы на все вопросы используя

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-types>

## Дополнительно

1. Ознакомьтесь с концепцией «небезопасного кода и указателей» в .NET. Познакомьтесь с ключевыми словами unsafe и fixed. <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/unsafe-code>
2. Ознакомьтесь с работой сборщика мусора (garbage collector, GC) в .NET. <https://docs.microsoft.com/en-us/dotnet/standard/garbagecollection/fundamentals>

<http://sergeyteplyakov.blogspot.com/2012/10/net.html> <https://habr.com/ru/post/463213/>

1. Ознакомьтесь с конструкцией using

## <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-statement>

## Вопросы

* 1. **Что такое .Net Framework и из чего он состоит?**

**.Net Framework** – платформа, которая создана ***Microsoft*** для разработки приложений

Состоит из:

* общеязыковая исполняющая среда **(CLR) Common Language Runtime)** Виртуальная Машина. Обеспечивает выполнение сборки (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация)
* **BCL/FCL (Base Class Library / Framework Class Library)** — набор библиотек классов, предоставляющих основные функциональные возможности, такие как работа с файлами, сетями, коллекциями, потоками и т.д.
  1. **Что такое CLR, FCL/BCL, CLI, IL?**

 **CLR (Common Language Runtime)** — это виртуальная машина для выполнения управляемого кода, которая управляет выполнением программ, компиляцией и сборкой мусора.

 **FCL/BCL (Framework Class Library / Base Class Library)** — библиотека классов.

 **CLI (Common Language Infrastructure)** —  архитектуру исполнительной системы (PowerShell)

 **IL (Intermediate Language)** — промежуточный код

* 1. **Пояснить работу JIT-компилятора?**

JIT-компилятор не компилирует весь IL-код сразу. Вместо этого, он компилирует методы **по мере их вызова**. Когда метод вызывается впервые:

* 1. JIT-компилятор компилирует IL-код этого метода в машинный код, который может выполняться непосредственно процессором.
  2. Скомпилированный машинный код кэшируется, чтобы его не нужно было компилировать заново при последующих вызовах.
  3. **Что такое CTS (Common Type System)?**

Система общих типов **CTS** определяет способ объявления, использования и управления типами в среде CLR, а также является важной составной частью поддержки межъязыковой интеграции в среде выполнения.

* 1. **Какие аспекты поведения определяет тип System.Object?**

Тип **System.Object** в .NET является базовым классом для всех типов, и он определяет несколько ключевых аспектов поведения, которые доступны каждому объекту

 Equals(): сравнение объектов.

 GetHashCode(): получение хэш-кода объекта.

 ToString(): строковое представление объекта.

 GetType(): получение типа объекта.

* 1. **Что находится в mscorlib dll?**

**mscorlib.dll** — это основная библиотека классов .NET, которая содержит базовые типы, такие как System.Object, System.String, коллекции, потоки и другие важные компоненты

* 1. **Что такое «сборка»? Из чего состоит сборка .NET?**

*Сборка* (assembly) — 1) это абстрактное понятие, для логической группировки одного или нескольких управляемых модулей или файлов ресурсов. 2) дискретная единица многократно используемого кода внутри CLR

* 1. **Какие виды сборок существуют?**
* с нестрогими именами (weakly named assemblies)
* со строгими именами (strongly named assemblies).
  + подписаны при помощи пары ключей, уникально идентифицирующей издателя сборки (безопасность, управление ее версиями, развертывание в любом месте пользовательского жесткого диска или в Интернете)
  + атрибуты: имя файла (без расширения), номер версии, идентификатор регионального стандарта и открытый ключ
  1. **Что такое assembly manifest?**

Манифест - ключевой компонент сборки.

Он хранит в себе:

* Имя сборки
* Номер версии
* Язык и региональные стандарты
* Список всех файлов сборки
* и т.д
  1. **Что такое GAC?**

**Global Assembly Cache** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) ***GAC*** — Глобальный Кэш Сборок) — в инфраструктуре [.NET](https://ru.wikipedia.org/wiki/.NET) — подсистема, хранящая сборки CLI в централизованном репозитории.

* 1. **Чем managed code отличается от unmanaged code**

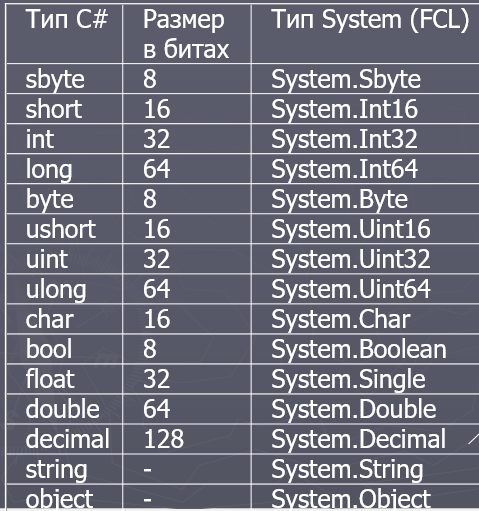
**Managed code** — код, который выполняется под управлением CLR, предоставляющей такие функции, как управление памятью, сборка мусора и безопасность. **Unmanaged code** — код, который выполняется напрямую процессором без вмешательства CLR (например, код на C++).

* 1. **Как и для чего определен метод Main?**

Метод Main — это точка входа для любого приложения на C#

* 1. **Варианты использования директивы using( using Directive ) в C#.**
  2. **Как связаны между собой сборки и пространства имен?**
  3. **Что такое примитивные типы данных? Перечислите их.**

Типы данных, которые поддерживаются компилятором напрямую, называются примитивными (primitive types) или встроенными у них существуют прямые аналоги в библиотеке классов .NET Framework Class Library



* 1. **Что такое ссылочные типы? Какие типы относятся к ним?**

**Ссылочные типы** — это типы данных, которые хранятся в памяти как ссылки на объекты, а не сами объекты.

* 1. **Какие типы относятся к типам-значениям?**

**Типы-значения** в C# — это типы данных, которые хранят значения непосредственно в переменной. Когда вы работаете с переменной типа-значения, вы работаете с копией значения, а не с ссылкой на объект.

### Основные типы-значения в C#:

1. **Примитивные типы**:

**Структуры (struct)**:

**Перечисления (enum)**:

**Nullable типы**:

* 1. **В чем отличие между ссылочными и значимыми типами данных?**
  2. **Что такое упаковка и распаковка значимых типов?**

Упаковка переменные из стека в кучу, распаковка – наоборот.

* 1. **В чем заключается разница между int и System.Int32? double и System.Double и т.д.?**

Int примитивный доступный для компилятора в C#, а System.Int32 это тип FSL и он доступен во всех языках .NET

* 1. **Для чего используется тип dynamic?**
  2. **В чем заключается главное отличие между var и dynamic?**
  3. **Что такое неявно типизированная переменная?(var)**
  4. **Для чего используют Nullable тип?**

**Nullable типы** (например, int?, double?) позволяют переменным, которые обычно хранят значения, принимать значение null. Это полезно в следующих ситуациях:

* 1. **Как объявить строковый литерал? Какие операции можно выполнять со строкой?**
  2. **Какие методы есть у типа String?**
  3. **В чем отличие пустой и null строки?**
  4. **Как можно выполнить сравнение строк? (==, ===,** Equals()**)**
  5. **В чем отличие типов String и StringBuilder?**
  6. **Поясните явные преобразования переменных с помощью команд Convert.**
  7. **Как выполнить консольный ввод/вывод?**
  8. **Приведите примеры определения и инициализации одномерных и двумерных массивов.**
  9. **Что такое ступенчатый массив? Как его задать?**
  10. **Какие типы можно использовать в foreach? Приведите пример.**
  11. **Что такое кортеж? Для чего и как он используется?**
  12. **Что такое локальная функция? Какова область ее видимости?**
  13. **В чем разница между кодом, заключенным в блок checked и кодом, заключенным в блок unchecked?**
  14. **Какой контекст (checked/unchecked) применяется по умолчанию? Как можно переопределить это поведение?**
  15. **Для чего используется ключевое слово fixed? Каковы особенности его использования?**

**Приведенные здесь и далее теоретические сведения не являются достаточными для освоения тем (это краткий вводный материал). Необходимо использовать дополнительную литературу!!!!!**

Язык программирования C# является прямым наследником языка С++. Он унаследовал многие синтаксические конструкции языка С и объектно-ориентированную модель С++. В отличие от С++ С# является чисто объектно-ориентированным языком. В объектно-ориентированном программировании ход выполнения программы определяется объектами. Объекты это экземпляры класса. Класс это абстрактный тип данных, определяемый пользователем ( программистом). Класс включает в себя данные и функции для обработки этих данных. В С# запрещены глобальные функции. Все функции должны быть обязательно определены внутри класса. Не является исключением и главная функция языка С# Main( ) (в отличии от языка С пишется с прописной буквы).

Объявление класса синтаксически имеет следующий вид:

сlass имя\_класса

{

// члены класса

}

Члены класса это данные и функции для работы с этими данными. Рассмотрим шаблон приложения, подготовленный для нас мастером:

using System;

namespace ConsoleApplication10

{

/// <summary>

/// Summary description for Class1.

/// </summary>

class Class1

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main(string[] args)

{

//

// TODO: Add code to start application here

//

}

}

}

Первая строчка проекта *using System;*, включает в себя директиву *using*, которая сообщает компилятору, где он должен искать классы (типы), не определенные в данном пространстве имен. Мастер, по умолчанию, указывает стандартное пространство имен *System*, где определена большая часть типов среды .NET.

Следующей строчкой *namespace ConsoleApplication10* мастер предложения определяет пространство имен для нашего приложения. По умолчанию в качестве имени выбирается имя проекта. Область действия пространства имен определяется блоком кода, заключенного между открывающей и закрывающей фигурными скобками. Пространство имен обеспечивает способ хранения одного набора имен отдельно от другого. Имена, объявленные в одном пространстве имен не конфликтуют, при совпадении, с именами, объявленными в другом пространстве имен.

В шаблоне приложения имеется множество строк, которые являются комментариями.

В C# определены три вида комментариев:

* многострочный (/\*…\*/)
* однострочный (//…)
* XML (///) – комментарий для поддержки возможности создания самодокументированного кода.

Строчка [STAThread] является атрибутом. Атрибуты задаются в квадратных скобках. С помощью атрибута в программу добавляется дополнительная описательная информация, связанная с элементом кода, непосредственно перед которым задается атрибут. В нашем случае указывается однопоточная модель выполнения функции Main. Заголовок функции:

static void Main(string[] args)

Функция Main определена как статическая (static) с типом возвращаемого значения void. Функция Main( ) C# как и функция main( ) языка С может принимать аргументы. Аргумент - это строковый массив, содержащий элементы командной строки. Тело функции пустое и в нем содержится, в виде комментария, предложение добавить туда код для запуска приложения:

// TODO: Add code to start application here

Воспользуемся этим предложением и добавим в тело функции одну строчку:

static void Main(string[] args)

{

//

// TODO: Add code to start application here

**Console.WriteLine("Привет!");**

//

}

Функции консольного ввода-вывода являются методами класса Console библиотеки классов среды .NET.

Для ввода строки с клавиатуры используется метод Console.ReadLine(), а для ввода одного символа метод Console.Read().

Для консольного вывода также имеются две метода

* метод Console.Write(), который выводит параметр, указанный в качестве аргумента этой функции, и
* метод Console.WriteLine(),который работает так же, как и Console.Write(), но добавляет символ новой строки в конец выходного текста.

Для анализа работы этих методов модифицируйте функцию Main( ) так, как показано ниже :

static void Main(string[] args)

{

//

// TODO: Add code to start application here

**Console.WriteLine("Введите ваше имя");**

**string str=Console.ReadLine();**

**Console.WriteLine("Привет "+str+"!!!");**

**Console.WriteLine("Введите один символ с клавитуры");**

**int kod=Console.Read();**

**char sim=(char)kod;**

**Console.WriteLine("Код символа "+sim+" = "+kod);**

//

}

Добавим

**Console.WriteLine("Код символа {0} = {1}",sim,kod);**

Первым параметром списка является строка, содержащая маркеры в фигурных скобках. Маркер это номер параметра в списке. При выводе текста вместо маркеров будут подставлены соответствующие параметры из остального списка. После маркера через запятую можно указать, сколько позиций отводится для вывода значений. Например, запись {1,3} означает, что для печати первого элемента списка отводится поле шириной в три символа. Причем, если значение ширины положительно, то производится выравнивание по правому краю поля, если отрицательно то по левому.

Добавим 4 новые строчки в конец кода функции Main():

**int s1=255;**

**int s2=32;**

**Console.WriteLine(" \n{0,5}\n+{1,4}\n-----\n{2,5}",s1,s2,s1+s2);**

**Console.WriteLine(" \n{1,5}\n+{0,4}\n-----\n{2,5}",s1,s2,s1+s2);**

//

Кроме того, после поля ширины через двоеточие можно указать форматную строку, состоящую из одного символа и необязательного значения точности.

Существует 8 различных форматов вывода:

* С – формат национальной валюты,
* D – десятичный формат,
* E – научный (экспоненциальный) формат,
* F – формат с фиксированной точкой,
* G – общий формат,
* N – числовой формат,
* P – процентный формат,
* X – шестнадцатеричный формат

Например, запись {2,9:C2} – означает, что для вывода второго элемента из списка, отводится поле шириной в 9 символов. Элемент выводится в формате денежной единицы с количеством знаков после запятой равной двум. При выводе результата происходит округление до заданной точности.