Темы урока

Тип данных object

- Обратить внимание на неэффективность для некоторых (значимых) типов данных.

Самостоятельная работа

- Задание 1
  - Определить переменную i типа object
  - Положить туда число 10
  - Увеличить переменную на 5
  - Вывести значение переменной на экран

```
object o = 10;
o = (int) o + 5;
Console.WriteLine(o);
```

- Задание 2
  - Определить переменную s типа object
  - Положить туда строку “abcd”
  - Вывести длину строки на экран

```
object s = "abcd";
Console.WriteLine(((string)s).Length);
```

Тип данных dynamic

- Обратить внимание на то, что во время выполнения может произойти ошибка, если тип данных окажется неподходящим.

Самостоятельная работа

- Задание 1
  - Определить переменную s типа object
  - Положить туда число 10
  - Увеличить переменную на 5
  - Вывести значение переменной на экран

```
dynamic s = 10;
s = s + 5;
Console.WriteLine(s);
```

- Задание 2
  - Определить переменную s типа object
  - Положить туда строку “abcd”
  - Вывести длину строки на экран

```
dynamic s = "abcd";
Console.WriteLine(s.Length);
```

  - Поменять начальное значение на 10 (число)
    Не меняя вторую строчку, запустить код на исполнение

```
dynamic s = "abcd";
Console.WriteLine(s.Length);
```

Обратить внимание, что в связи с увеличением опасности упасть в runtime использовать dynamic лучше не нужно, если вам кажется, что здесь необходим dynamic, значит, скорее всего, вы что-то делаете не так.

Объявление переменных с помощью var

- Обратить внимание на хорошие и плохие практики использования var.
- Рекомендовать поначалу использовать явные типы данных для переменных.
- Рассказать про метод GetType(), который есть у любой переменной в C#.

Самостоятельная работа

- Определить следующие переменные через var:
  - переменную f равную 3.14 так, чтобы тип был float (Single)
  - переменную d равную 1 так, чтобы тип был double (Double)
  - переменную l равную 49 так, чтобы тип был long (Int64)
  - переменную b равную 255 так, чтобы тип был byte (Byte)

```
var f = 3.14F;
Console.WriteLine(f.GetType());
var d = 1D;
Console.WriteLine(d.GetType());
var l = 49L;
Console.WriteLine(l.GetType());
var b = 255; // impossible to do it!
Console.WriteLine(b.GetType());
```

Ссылочные и значимые типы данных

- В этом месте - короткое введение:
  - Числа и булевы переменные являются значимыми типами (value types). Значимые переменные обязательно должны иметь значение.
  - string, object, dynamic являются ссылочными типами (reference types). Ссылочные типы могут не иметь значения, это записывается как null.
- Описание разницы хранения в памяти самого значения и ссылки на него.
- string - единственный из известных нам пока ссылочный тип данных (и ещё object, но мы им пользоваться не будем)
- Функция default() для вывода значения, которым инициализируется переменная при отсутствии ининциализирующего значения.

Самостоятельная работа

- вывести на экран значения по умолчанию для следующих типов данных и (опираясь на вывод) определить их тип – ссылочный или значимый
  byte, long, double, string, object, dynamic

Nullable для значимых типов

- Использование знака вопроса ? после имени типа при определении переменной превращает значимый тип данных в nullable тип.
- Свойство HasValue помогает определить задана значимая часть переменной
- Свойство Value возвращает значимую часть переменно, если значимая часть не задана происходит ошибка выполнения.

Самостоятельная работа:

- Объявить переменные
    - nullable int равную 5,
    - nullable float без инициализации
- Вывести на экран для каждой из них
    - Значение свойства HasValue
    - Значение свойства Value
        - Показать на ошибку времени выполнения
        - Закомментировать строку с ошибкой и запустить
        - Инициализировать переменную float? и запустить еще раз,
- Попробовать объявить nullable string, object или dynamic: указать, что происходит ошибка компиляции.

Массивы (введение)

- Удобны, когда нужно обрабатывать несколько **однотипных** значений
- Необходимо **заранее** знать количество элементов массива
- Для удобства обработки удобно использовать **циклы**
- Про циклы рассказать **очень поверхностно**, этому будет посвящена отдельная тема, пока мы просто изучаем массивы.
- Свойство Length.

Самостоятельная работа

- Определить 2 массива одинаковой длины:
    - trees со значениями “Ясень”, “Липа”, “Кедр”
    - ages со значениями 32, 24, 43
    - Вывести на экран значения каждого из массивов рядом
        - Ясень - возраст в годах: 32
        - Липа - возраст в годах: 24
        - Кедр - возраст в годах: 43.

Работа с консолью

- WriteLine, Write
- ReadLine, ReadKey
- Шаблонирование строк через $"template { var}"

Самостоятельная работа

- Прочитать из консоли 5 переменных строкового типа и положить в массив.

Пространство имен

- Объяснить на примере System, использующейся для доступа к классу Console.
- Показать, что будет, если убрать импорт пространства имён System.

Константы

Хорошо бы на этом этапе упомянуть про константы.

Константы — это постоянные значения, которые известны во время компиляции и не изменяются во время выполнения программы. Константы должны объявляться с модификатором const.

Только встроенные типы C# (за исключением System.Object) можно объявлять как const:

- bool System.Boolean
- byte System.Byte
- sbyte System.SByte
- char System.Char
- decimal System.Decimal
- double System.Double
- float System.Single
- int System.Int32
- uint System.UInt32
- long System.Int64
- ulong System.UInt64
- object System.Object
- short System.Int16
- ushort System.UInt16
- string System.String

Показать синтаксис:

Домашнее задание

Вариант 1 (попроще)

Задание

Написать консольное приложение, запрашивающее у пользователя имена трех человек. Затем также запрашивающее возраст этих людей. Затем программа должна вывести на экран информацию о людях и их возрастах через 4 года в следующем формате:

```
Name: <name of the person # 1>, age in 4 years: <age of the person #1 in 4 years>
Name: <name of the person # 2>, age in 4 years: <age of the person #2 in 4 years>
Name: <name of the person # 3>, age in 4 years: <age of the person #3 in 4 years>
```

Программа не должна закрываться пока не нажата любая клавиша.
Необходимо выполнить задание с использованием массивов!

Пример вывода

```
Name: Andrey, age in 4 years: 40
Name: Alex, age in 4 years: 23
Name: Artem, age in 4 years: 5
```

Решение

Без циклов: **L03_HW1_solution_1_without_loops**.
С циклами: **L03_HW1_solution_2_with_loops**.

Вариант 2 (посложнее)

Задание

Вывести на экран таблицу умножения Пифагора 10×10 элементов от 1 до 10. Исходные значения множителей должны храниться в массивах.

(* сложная часть) Спроектировать приложение так, чтобы изменение количества или значений множителей потребовало минимум изменений в коде (1–2 изменения).

Пример вывода

```
   *   1   2   3   4   5   6   7   8   9  10
   1   1   2   3   4   5   6   7   8   9  10
   2   2   4   6   8  10  12  14  16  18  20
   3   3   6   9  12  15  18  21  24  27  30
   4   4   8  12  16  20  24  28  32  36  40
   5   5  10  15  20  25  30  35  40  45  50
   6   6  12  18  24  30  36  42  48  54  60
   7   7  14  21  28  35  42  49  56  63  70
   8   8  16  24  32  40  48  56  64  72  80
   9   9  18  27  36  45  54  63  72  81  90
  10  10  20  30  40  50  60  70  80  90 100
```

Решение

См. **L03_HW2_solution**.