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

Задание

1) **Типы**
   a. Определите переменные всех возможных примитивных типов C# 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)
   b. 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.***
   c. Выполните упаковку и распаковку **значимых** типов.
   d. Продемонстрируйте работу с неявно типизированной переменной.
   e. 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).

   f. . Определите переменную типа *var* и присвойте ей любое значение. Затем следующей инструкцией присвойте ей значение другого типа. Объясните причину ошибки.
2) **Строки**
   a. Объявите строковые литералы. Сравните их.
   b. 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.
   c. Создайте пустую и *null* строку. Продемонстрируйте использование метода *string.IsNullOrEmpty*. Продемонстрируйте что еще можно выполнить с такими строками
   d. 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)
3) **Массивы (** https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/ **)**
   a. Создайте целый двумерный массив и выведите его на консоль в отформатированном виде (матрица).

b. Создайте **одномерный** массив строк. Выведите на консоль его содержимое, длину массива. Поменяйте произвольный элемент (пользователь определяет позицию и значение).
c. 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.
d. Создайте **неявно типизированные переменные** для хранения массива и строки.

4) **Кортежи (** https://docs.microsoft.com/en-us/dotnet/csharp/tuples **)**
   a. Задайте кортеж из 5 элементов с типами *int, string, char, string, ulong*.
   b. Выведите кортеж на консоль целиком и выборочно (например 1, 3, 4 элементы)
   c. Выполните распаковку кортежа в переменные. Продемонстрируйте различные способы распаковки кортежа. Продемонстрируйте использование переменной (_). (доступно начиная с C#7.3)
   d. Сравните два кортежа.
5) 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 .
6) Работа с *checked*/*unchecked*: (https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/checked-and-unchecked)
   a. Определите две **локальные** функции.
   b. 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.
   c. Вызовите две функции. Проанализируйте результат.
7) Загрузите проект в свой репозиторий на GitHub.
8) 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/

3) 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?
2. Что такое CLR, FCL/BCL, CLI, IL?
3. Пояснить работу JIT-компилятора?
4. Что такое CTS (Common Type System)?
5. Какие аспекты поведения определяет тип System.Object?
6. Что находится в mscorlib dll?
7. Что такое «сборка»? Из чего состоит сборка .NET?
8. Какие виды сборок существуют?
9. Что такое assembly manifest?
10. Что такое GAC?
11. Чем managed code отличается от unmanaged code
12. **Как и для чего определен метод Main?**
13. **Варианты использования директивы using**(using Directive) в C#.
14. Как связаны между собой сборки и пространства имен?
15. **Что такое примитивные типы данных?** Перечислите их.
16. **Что такое ссылочные типы? Какие типы относятся к ним?**
17. **Какие типы относятся к типам-значениям?**
18. **В чем отличие между ссылочными и значимыми типами данных?**
19. **Что такое упаковка и распаковка значимых типов?**
20. В чем заключается разница между int и System.Int32? double и System.Double и т.д.?
21. Для чего используется тип dynamic?
22. **В чем заключается главное отличие между var и dynamic?**
23. **Что такое неявно типизированная переменная?**
24. **Для чего используют Nullable тип?**
25. Как объявить строковый литерал? Какие операции можно выполнять со строкой?
26. Какие есть способы для задания и инициализации строк?
27. Какие методы есть у типа String?
28. В чем отличие пустой и null строки?
29. **Как можно выполнить сравнение строк?**
30. **В чем отличие типов String и StringBuilder?**
31. Поясните явные преобразования переменных с помощью команд Convert.
32. Как выполнить консольный ввод/вывод?
33. Приведите примеры определения и инициализации одномерных и двумерных массивов.

34. **Что такое ступенчатый массив?** Как его задать?
35. **Какие типы можно использовать в foreach**? Приведите пример.
36. Что такое кортеж? Для чего и как он используется?
37. **Что такое локальная функция?** Какова область ее видимости?
38. В чем разница между кодом, заключенным в блок checked и кодом, заключенным в блок unchecked?
39. Какой контекст (checked/unchecked) применяется по умолчанию? Как можно переопределить это поведение?
40. 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 C++. Он унаследовал многие синтаксические конструкции языка C и объектно-ориентированную модель C++. В отличие от C++ C# 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 C# запрещены глобальные функции. Все функции должны быть обязательно определены внутри класса. Не является исключением и главная функция языка C# Main( ) (в отличии от языка C пишется с прописной буквы).

Объявление класса синтаксически имеет следующий вид:

```
class 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 C 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:

- C – 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.