#2. Основы CLR и .NET. Типы данных

Для выполнения работы рекомендуется использовать .NET Framework ***v4.7.2*** *или выше.*

Базовый уровень: теория.

1. Изучите основные типы .NET. В чем заключается разница между int и System.Int32? double и System.Double и т.д.? https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/built-in-types-table
2. Ознакомьтесь с **var** и **dynamic**: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/using-type-dynamic
3. Ознакомьтесь с основными методами работы с консолью из класса Console: *Read*, *ReadLine*, *Write*, *WriteLine*. https://docs.microsoft.com/en-us/dotnet/api/system.console?view=netframework-4.8
4. Определите 5 переменных различных типов. Осуществите ввод и вывод их значений используя консоль.
5. Ознакомьтесь с возможностями работы со строками:
    a. Изучите: https://docs.microsoft.com/en-us/dotnet/api/system.string?view=netframework-4.8
    b. Изучите класс и работу со *StringBuilder*: https://docs.microsoft.com/en-us/dotnet/api/system.text.stringbuilder?view=netframework-4.8
6. Изучите возможности форматирования строк:
    a. Используя *string.format*: https://docs.microsoft.com/en-us/dotnet/api/system.string.format?view=netframework-4.8
    b. Используя интерполирование строк https://docs.microsoft.com/en-us/dotnet/api/system.string.format?view=netframework-4.8 (доступно начиная с **C#6**)
7. Изучите методы класса Object: https://docs.microsoft.com/en-us/dotnet/api/system.object?view=netframework-4.8#methods . Охарактеризуйте открытые и закрытые методы класса.
8. Ознакомьтесь с возможностями Nullable-типов.
   Подсказка:
   - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/nullable-types/

- https://docs.microsoft.com/en-us/dotnet/api/system.nullable-1?view=netframework-4.8

9. Изучите работу с массивами в .NET https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/
10. Познакомьтесь с операциями приведения/преобразования типов:
    a. https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/conversions#implicit-conversions
    b. Изучите возможности класса *Convert*: https://docs.microsoft.com/en-us/dotnet/api/system.convert?view=netframework-4.8
    c. Познакомьтесь с операциями «*упаковки*» и «*распаковки*» типов.
11. Изучите код, приведенный ниже. Будьте готовы к вопросам, связанным с ним.

Базовый уровень: практика.

1. Определите несколько переменных различных типов. Продемонстрируйте на них операции «упаковки» и «распаковки».
2. Определите переменную типа **int** со значением, равным вашему номеру в подгруппе. Выполните операции явного и неявного приведения к 3 отличным от **int** типам.
3. Определите переменную типа string со значением, равным вашему имени. Выведите на консоль строку: “My name is <name>”, где вместо <name> - значение переменной, используя:
    a. *string.format*;
    b. интерполирование строк.
4. Продемонстрируйте использование открытых методов класса *Object* на ваших переменных.
5. Создайте две переменных типа **string**. Продемонстрируйте на них работу следующих методов:
    a. *Compare*
    b. *Contains*
    c. *Substring*
    d. *Insert*
    e. *Replace*
6. Продемонстрируйте использование метода *string.IsNullOrEmpty* на примере пустой и *null*-строк.
7. Определите переменную неопределенного типа и присвойте ей любое значение. Затем следующей инструкцией присвойте ей значение

другого типа. Объясните причину ошибки.
Подсказка:

```
var variable = "";

variable = 5;
```

8. Продемонстрируйте работу с Nullable-переменной.

Средний уровень: теория.

1. Ознакомьтесь с понятием «*локальная функция*». https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/local-functions
2. Ознакомьтесь с типом *Tuple*. https://docs.microsoft.com/en-us/dotnet/csharp/tuples
3. Ознакомьтесь с концепцией «небезопасного кода и указателей» в .NET. Познакомьтесь с ключевыми словами **unsafe** и **fixed**. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/unsafe-code-pointers/index
4. Познакомьтесь с ключевыми словами **checked** и **unchecked**. https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/checked-and-unchecked

Средний уровень: практика.

1. Определите локальную функцию, принимающую параметром кортеж из 2 элементов **int** типа. Продемонстрируйте вызов этой функции.
2. Создайте кортеж из трех именованных аргументов. Продемонстрируйте различные способы распаковки кортежа. Продемонстрируйте использование переменной (_). (доступно начиная с **C#7.3**)
3. Работа с **checked**/**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.

Повышенный уровень: теория.

1. Ознакомьтесь с работой сборщика мусора (garbage collector, GC) в .NET.
   https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals
   https://metanit.com/sharp/tutorial/8.1.php
   http://sergeyteplyakov.blogspot.com/2012/10/net.html
   https://habr.com/ru/post/463213/
2. Ознакомьтесь с конструкцией **using**
   https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-statement

Повышенный уровень: практика.

1. Продемонстрируйте использование конструкции **using** с использованием класса, представленного ниже:
   a. Создайте экземпляр класса, используя **using**.
   b. Вызовите метод *GetState* и выведите результат в консоль.

```
2 references
class Example : IDisposable
{
    private readonly int _state;

    1 reference
    public Example(int state)
    {
        _state = state;
    }

    1 reference
    public int GetState() => _state;

    0 references
    public void Dispose()
    {
    }
}
```

Вопросы.

1. Что такое .NET Framework? Из чего он состоит?
2. Что такое CLR, FCL/BCL, CLI, IL?
3. Как работает JIT-компилятор?
4. Что такое CTS?
5. Что такое «сборка»? Из чего состоит сборка .NET?
6. Какие виды сборок существуют?
7. Что такое GAC?
8. Как происходит поиск нужной сборки?
9. Что содержит mscorlib.dll?
10. Какая наименьшая исполняемая единица .NET?
11. Для чего служит метод Main? Какова его сигнатура? Для чего служат его параметры?
12. Для чего служит директива using? Какие существуют варианты ее использования?
13. Какие типы данных существуют в .NET?
14. Что такое примитивные типы данных? Перечислите их.
15. В чем разница между ссылочными типами данных и значимыми?
16. Какие примитивные типы относятся к ссылочным? К значимым?
17. В чем основная разница между string и StringBuilder?
18. В чем заключается главное отличие между var и dynamic?
19. Что такое упаковка и распаковка типов?
20. Для чего используется тип Nullable?
21. Как можно преобразовать один тип в другой? Перечислите все возможные способы.
22. Какие виды массивов существуют в .NET?

Средний уровень

23. Что такое локальная функция? Какова область ее видимости?
24. В чем разница между кодом, заключенным в блок checked и кодом, заключенным в блок unchecked?
25. Какой контекст (**checked**/**unchecked**) применяется по умолчанию? Как можно переопределить это поведение?
26. Для чего используется ключевое слово **fixed**? Каковы особенности его использования?

Повышенный уровень

27. Объясните использование интерфейса *IDisposable*.
28. Объясните работу конструкции **using**.
29. Поясните (вплоть до поколений) работу сборщика мусора в .NET.
30. В чем разница между *short weak reference* и *long weak reference*?

Пример кода:

```
static void Main(string[] args)
{
      int i = 5;

      object o = i;
      int i2 = (int)o;

      long l = i;
      long l2 = (long) i;
      long l3 = Convert.ToInt64(i);

      int i3 = l;
      int i4 = (int) l;

      string str = string.Empty;
      string str2 = null;

      int i5 = null;
      int? i6 = null;
      Nullable<int> i7 = 6;

      int[,] array = { { 1, 2, 3 }, { 4, 5, 6 } };

      Console.WriteLine(array.Length);

      Console.WriteLine(string.Format("My name is {0}", "Program"));
      Console.WriteLine($"My name is {"Program"}");

      Console.WriteLine(5 + 10);

      string i5String = i5.ToString();
}
```