**Вопросы к лабораторной работе №1**

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

**.**Net Framework это программная платформа, на которой выполняется код, написанный на любом из этих языков. Платформа . NET Framework состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов.

**2. Что такое CLR, FCL/BCL, CLI, IL?**

CLR (Common language runtime) - общеязыковая среда исполнения, виртуальная машина, на которой исполняются все приложения,

работающие в среде .NET.

FCL (англ. Framework Class Library), сокращённо BCL — стандартная библиотека классов платформы

CLI (англ. Common Language Infrastructure) — спецификация общеязыковой инфраструктуры, описывающая исполняемый код и среду выполнения, которые составляют платформу .NET.

IL -интерпретируемый код

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

JIT (Just-in-time) компиляция включает в себя компиляцию исходного кода или байт-кода в машинный код и его выполнение.

Как правило, результат записывается в память и исполняется сразу же, без промежуточного сохранения на диск или его вызов как отдельной программы.

**4. Что такое CTS (Common Type System)?**

Формальная спецификация, определяющая, как какой-либо тип (класс, интерфейс, структура, встроенный тип данных) должен быть определён для его правильного выполнения средой .NET.

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

Класс объектов, который хранит в себе другие классы

**6. Что находится в mscorlib dll?**

Содержит функции, которые могут вызываться из других программ (mscorlib.dll). Динамически подключаемая библиотека, справочник, хранящий информацию и инструкции для исполняемых файлов.

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

Сборки представляют собой базовые элементы развертывания, управления версиями, повторного использования, назначения областей активации и прав доступа для приложений на основе платформы .НЕТ

Дискретная единица многократно используемого кода внутри CLR. exe, dll.

**8. Какие виды сборок существуют?**

С нестрогими именами и со строгими именами (подписаны при помощи пары ключей, уникально идентифицирующей издателя сборки).

Атрибуты: имя файла, номер версии, идентификатор регионального стандарта и открытый ключ.

**9. Что такое assembly manifest?**

Манифест сборки (assembly manifest) – это внутренняя часть сборки, которая позволяет идентифицировать сборку, указывает файлы, которые включаются в реализацию сборки, описывает типы и ресурсы, используемые в сборке, указывает зависимости от других сборок, а также набор прав доступа, которые необходимы сборке.

**10. Что такое GAC?**

GAC- подсистема, хранящая сборки CLI (CLI assembly) в централизованном репозитории. Система разработана для борьбы с проблемами разделяемых библиотек, например таких, как DLL hell.

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

Мanaged code and unmanaged code - Управляемый код: код написан на языке . NET, например С#, VB.NET. UnManaged Код: код не написан на языке . NET и MSIL не понимает, что это такое и не может работать под CLR; как сторонние элементы управления, которые мы использовали в наших приложениях.

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

Метод Main() содержит параметры командной строки, тип возвращаемого значения может быть любой, int или void.

**13. Варианты использования директивы using( using Directive ) в C#.**

Конструкция using оформляет блок кода и создает объект некоторого класса, который реализует интерфейс IDisposable, в частности, его метод Dispose. При завершении блока кода у объекта вызывается метод Dispose.

Важно, что данная конструкция применяется только для классов, которые реализуют интерфейс IDisposable.

**14. Как связаны между собой сборки и пространства имен?**

Перед использованием using SomeNameSpace;, вы должны указать в проекте ссылку на сборку, в которой объявлены типы данного пространства имен.

**15. Что такое примитивные типы данных? Перечислите их.**

**Примитивный** **тип** - это несоставной **тип** **данных**, то есть такие **типы** как int, double, boolean, single, byte и т.д., занимающие в памяти строго определенное количество байт.

Целые числа - byte, short, int, long

Числа с плавающей точкой (иначе вещественные) - float, double

Логический - boolean

Символьный - char

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

В переменных ссылочных типов хранятся ссылки на их данные (объекты)

Тип object

Тип string

Классы (class)

Интерфейсы (interface)

Делегаты (delegate)

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

Переменные **типа** значений содержат свои данные непосредственно.

Целочисленные типы (byte, sbyte, short, ushort, int, uint, long, ulong)

Типы с плавающей запятой (float, double)

Тип decimal

Тип bool

Тип char

Перечисления enum

Структуры (struct)

**18. В чем отличие между ссылочными и значимыми типами данных?**

* типы-значения (значимые типы) описываются структурами. Ссылочные типы описываются классами;
* при объявлении переменной значимого типа экземпляр этой переменной размещается в стеке полностью. При объявлении ссылочной переменной, в стеке размещается только ссылка на экземпляр, а сам экземпляр (данные + методы) размещается в так называемой управляемой «куче» (managed heap);
* при присваивании переменных значимого типа создается полный дубликат переменной. При присваивании переменных ссылочного типа копируется только ссылка на объект, таким образом обе ссылки указывают на один и тот же участок памяти.

**19. Что такое упаковка и распаковка значимых типов?**

Доступный для программиста механизм преобразования размерных типов данных языка C# из значимых в ссылочные и обратно через задействование свойств фундаментального базового класса Object.

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

Разницы никакой, int и Int32 после компиляции в *IL* код станут одним и тем же. int это ключевое слово конкретно для c#

**21. Для чего используется тип dynamic?**

**Тип** **dynamic** позволяет писать код, который обходит проверку времени компиляции. Компилятор предполагает, что любая операция, определенная для объекта **типа** **dynamic**, является допустимой

**22. В чем заключается главное отличие между var и dynamic?**

Первый — не более чем синтаксический сахар, который на стадии компиляции подменяется фактическим типом переменной.  
Второй — фактически говорит компилятору: "на стадии компиляции не трогай код с использованием этой переменной, просто предположи, что он рабочий. А вот во время выполнения снова запусти компилятор и попробуй откомпилировать этот код, заменив dynamic на фактический тип, который будет присвоен данной переменной. Если это не получится — вали всё приложение, т.к. я сам себе злобный буратино."

*вар-неявная типизация,а дайнемик это как дополнение к типу данных*

**23. Что такое неявно типизированная переменная?**

Все переменные в C# должны быть объявлены. Как правило, при объявлении переменной сначала указывается тип, например int или bool, а затем имя переменной. Но начиная с версии C# 3.0, компилятору предоставляется возможность самому определить тип локальной переменной, исходя из значения, которым она инициализируется. Такая переменная называется неявно типизированной.

**24. Для чего используют Nullable тип?**

Значение null по умолчанию могут принимать только объекты ссылочных типов. Однако в различных ситуациях бывает удобно, чтобы объекты числовых типов данных имели значение null, то есть были бы не определены. Стандартный пример - работа с базой данных, которая может содержать значения null. И мы можем заранее не знать, что мы получим из базы данных - какое-то определенное значение или же null. Для этого надо использовать знак вопроса ? после типа значений.

Но фактически запись ? является упрощенной формой использования структуры **System.Nullable<T>**. Параметр T в угловых скобках представляет универсальный параметр, вместо которого в конкретной задача уже подставляется конкретный тип данных. Следующие виды определения переменных будут эквивалентны.

*ссылаемся на адресс нулевого элемента*

**25. Как объявить строковый литерал? Какие операции можно выполнять со строкой?**

Строковые литералы представляют строки. Строки заключаются в двойные кавычки:

|  |  |
| --- | --- |
| 1  2  3 | Console.WriteLine("hello");  Console.WriteLine("кириллицей");  Console.WriteLine("hello word"); |

Если внутри строки необходимо вывести двойную кавычку, то такая внутренняя кавычка предваряется обратным слешем:

|  |  |
| --- | --- |
| 1 | Console.WriteLine("Компания \"Coca Cola\""); |

* Конкатенация
* Сравнение
* Поиск
* Разделение
* Обрезка
* Вставка
* Удаление
* Замена
* Смена регистра

**26. Какие есть способы для задания и инициализации строк?**

**описание строки может быть выполнено одним из следующих способов:**

char string1[50]; – где 50 – максимальное допустимое количество символов в строке.

char string2[]={‘С’, ‘т’, ‘р’, ‘о’, ‘к’, ‘а’, ‘1’, ‘.’, ‘\0’}; – помещение строки в массив при инициализации.

char string3[] =”Еще один пример строки, описанной при инициализации”;

Кроме перечисленных способов поместить строку в массив можно с помощью функций ввода. В функциях printf(), scanf() для символьных строк используется спецификация преобразования %s.

char str[]=”String one”;

printf(“%s”,str);

Кроме описанных выше, существует еще один способ описания строковых констант. Такой способ возможен только для массивов символов.

Использование указателей, а не массивов фиксированной длины целесообразно, когда предложения или слова должны быть разной длины.

**27. Какие методы есть у типа String?**

concat(): объединяет строки

valueOf(): преобразует объект в строковый вид

join(): соединяет строки с учетом разделителя

сompare(): сравнивает две строки

charAt(): возвращает символ строки по индексу

getChars(): возвращает группу символов

equals(): сравнивает строки с учетом регистра

**28. В чем отличие пустой и null строки?**

**null** - это отсутствие какого-либо значения. Если вы попытаетесь обратиться к членам экземпляра **s1**, получите **NullReferenceException**. Вторая строка, хоть и пустая, но память под неё в куче выделена, и вы можете к ней обращаться.

**29. Как можно выполнить сравнение строк?**

сompare(): сравнивает две строки или ==

**30. В чем отличие типов String и StringBuilder?**

Принципиальное **отличие** строки **StringBuilder** от строки **string** заключается в том, что элементы строки **StringBuilder** можно изменять напрямую, при этом такая строка заново не создаётся.

**31. Поясните явные преобразования переменных с помощью команд Convert.**

Команды для явных преобразований

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| Convert.ToBoolean(Var1) | Var1, преобразованная в bool |
| Convert.ToByte(Var1) | Var1, преобразованная в byte |
| Convert.ToChar(Var1) | Var1, преобразованная в char |
| Convert.ToDecimal(Var1) | Var1, преобразованная в decimal |
| Convert.ToDouble(Var1) | Var1, преобразованная в double |
| Convert.ToInt16(Var1) | Var1, преобразованная в short |
| Convert.ToInt32(Var1) | Var1, преобразованная в int |
| Convert.ToInt62(Var1) | Var1, преобразованная в long |
| Convert.ToSbyte(Var1) | Var1, преобразованная в sbyte |
| Convert.ToSingle(Var1) | Var1, преобразованная в float |
| Convert.ToString(Var1) | Var1, преобразованная в string |
| Convert.ToUInt16(Var1) | Var1, преобразованная в ushort |
| Convert.ToUInt32(Var1) | Var1, преобразованная в uint |
| Convert.ToUINT64(Var1) | Var1, преобразованная в ulong |

Здесь на месте Var1 может указываться переменная практически любого типа (о том, могут ли данные команды работать с этим типом, будет сообщать сам компилятор).

**32. Как выполнить консольный ввод/вывод?**

Для вывода информации на консоль используется встроенный метод **Console.WriteLine**.

Кроме Console.WriteLine() можно также использовать метод **Console.Write()**, он работает точно так же за тем исключением, что не осуществляет переход на следующую строку.

Кроме вывода информации на консоль мы можем получать информацию с консоли. Для этого предназначен метод **Console.ReadLine()**. Он позволяет получить введенную строку.

**33. Приведите примеры определения и инициализации одномерных и двумерных массивов.   
Одномерные:**

int[] nums2 = new int[4] { 1, 2, 3, 5 };

int[] nums3 = new int[] { 1, 2, 3, 5 };

int[] nums4 = new[] { 1, 2, 3, 5 };

int[] nums5 = { 1, 2, 3, 5 };

**Двумерные:**

int[,] nums1;

int[,] nums2 = new int[2, 3];

int[,] nums3 = new int[2, 3] { { 0, 1, 2 }, { 3, 4, 5 } };

int[,] nums4 = new int[,] { { 0, 1, 2 }, { 3, 4, 5 } };

int[,] nums5 = new [,]{ { 0, 1, 2 }, { 3, 4, 5 } };

int[,] nums6 = { { 0, 1, 2 }, { 3, 4, 5 } };

**34. Что такое ступенчатый массив? Как его задать?**

Ступенчатый массив представляет собой массив массивов, в котором длина каждого массива может быть разной.

int[][] nums = new int[3][];

nums[0] = new int[2] { 1, 2 };

nums[1] = new int[3] { 1, 2, 3 };

nums[2] = new int[5] { 1, 2, 3, 4, 5 };

или

int[][] numbers = {

    new int[] { 1, 2 },

    new int[] { 1, 2, 3 },

    new int[] { 1, 2, 3, 4, 5 }

};

**35. Какие типы можно использовать в foreach? Приведите пример.**

Оператор foreach можно использовать с экземпляром любого типа, который удовлетворяет следующим условиям:

-Тип имеет открытый метод без параметров GetEnumerator. Начиная с C# 9.0 метод GetEnumerator может быть методом расширения типа.

-тип возвращаемого значения метода GetEnumerator должен содержать открытое свойство Current и открытый метод MoveNext без параметров с типом возвращаемого значения bool.

foreach(char c in "Tom")

{

    Console.WriteLine(c);

}

**36. Что такое кортеж? Для чего и как он используется?**

Кортеж — упорядоченный набор фиксированной длины.

Кортежи могут выступать в качестве результата функции. Например, одной из распространенных ситуаций является возвращение из функции двух и более значений, в то время как функция может возвращать только одно значение. И также кортеж может передаваться в качестве параметра в метод.

**37. Что такое локальная функция? Какова область ее видимости?**

Локальные функции представляют функции, определенные внутри других методов.

**Локальная** **область** **видимости** – это часть исходного текста программы, содержащаяся в определении **функции** (или блоке внутри тела **функции**).

**38. В чем разница между кодом, заключенным в блок checked и кодом, заключенным в блок unchecked?**

При возникновении целочисленного арифметического переполнения контекст проверки переполнения определяет, что происходит следующим образом:

В проверяемом контексте (checked )System.OverflowException возникает исключение; если переполнение происходит в константном выражении, возникает ошибка времени компиляции.

В незавершенном контексте (unckecked) результат операции усекается путем удаления любых битов высокого порядка, которые не помещаются в целевой тип. Например, в случае сложения он переносится из максимального значения к минимальному значению, как показано в предыдущем примере.

**39. Какой контекст (checked/unchecked) применяется по умолчанию? Как можно переопределить это поведение?**

По умолчанию арифметические операции выполняются в непроверенном контексте (unchecked).

**40. Для чего используется ключевое слово fixed? Каковы особенности его использования?**

Оператор fixed задает указатель на управляемую переменную и "закрепляет" эту переменную во время выполнения оператора. Указатели на перемещаемые управляемые переменные полезны только в контексте fixed. Без контекста fixed при сборке мусора эти переменные могут переноситься непредсказуемым образом. Компилятор C# позволяет присвоить указатель только управляемой переменной в операторе fixed.