№ 3 Проектирование типов. Классы.

Задание

1) Определить класс, указанный в варианте, содержащий:

- Не менее трех конструкторов (с параметрами и без, а также с параметрами по умолчанию);
- **статический конструктор** (конструктор типа);
- определите **закрытый** конструктор; предложите варианты его вызова;
- поле - **только для чтения** (например, для каждого экземпляра сделайте поле только для чтения ID - равно некоторому уникальному номеру (хэшу) вычисляемому автоматически на основе инициализаторов объекта);
- поле- константу;
- **свойства** (get, set) – для всех поле класса (поля класса должны быть закрытыми); Для одного из свойств ограните доступ по set
- в одном из методов класса для работы с аргументами используйте **ref - и out-параметры**.
- создайте в классе **статическое поле**, хранящее количество созданных объектов (инкрементируется в конструкторе) и **статический метод** вывода информации о классе.
- сделайте касс **partial**
- переопределяете методы класса Object: Equals, для сравнения объектов, GetHashCode; для алгоритма вычисления хэша руководствуйтесь стандартными рекомендациями, ToString – вывода строки – информации об объекте.

2) Создайте несколько объектов вашего типа. Выполните вызов конструкторов, свойств, методов, сравнение объекты, проверьте тип созданного объекта и т.п.
3) Создайте массив объектов вашего типа. И выполните задание, выделенное курсивом в таблице.

4) Создайте и выведите анонимный тип (по образцу вашего класса).

5) Ответьте на вопросы, приведенные ниже

Используйте документацию http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx

Далее приведен перечень классов:

Вариант 1	Создать класс **Вектор**, который содержит массив int, число элементов и переменную состояния. Определить индексатор. В переменную состояния устанавливать код ошибки, при определенной ситуации. Определить методы: сложения и умножения, которые производят эти операции с данными класса вектор и числом int. *Создать массив объектов. Вывести:* *a) список векторов, содержащих 0;* *b) список векторов с наименьшим модулем.*
Вариант 2	Создать класс **Стек** вещественных. Определить индексатор, методы проверки стека, добавления и удаления элементов. *Создать массив объектов. Вывести:* *a) стеки с наименьшим/наибольшим верхним элементом;* *b) список стеков, содержащих отрицательные элементы.*
Вариант 3	Создать класс типа - **Множество**. Методы: добавляют элемент к множеству, удаляют элемент, выводят текущее количество элементов множества. *Создать массив объектов. Вывести:* *a) множества с наименьшей/наибольшей суммой элементов;* *b) список множеств, содержащих отрицательные элементы.*
Вариант 4	Создать класс типа - **Дата** с полями: день (1-31), месяц (1-12), год (целое число). Свойства установки дня, месяца и года и конструкторы должны проверять корректность задаваемых значений. Добавить методы печати по шаблону: “5 января 2018 года” и “05/01/2018”. *Создать массив объектов. Вывести:* *a) список дат для заданного года;* *b) список дат, которые имеют заданное число;*
Вариант 5	Создать класс **SuperString**. Методы: вывода длины строки, проверки существует ли в строке заданный символ, замены одного символа в строке на другой. *Создать массив объектов. Вывести:* *a) список строк определенной длины;* *b) список строк, которые содержат заданное слово.*

Вариант 6	Определить класс Булева матрица (**BoolMatrix**). Реализовать методы для логического сложения (дизъюнкции), умножения и инверсии матриц. Реализовать методы для подсчета числа единиц в матрице. *Создать массив объектов. Вывести:* *a) список матриц с наибольшим/наименьшим количеством единиц;* *b) список матриц с равным количеством заданного символа в каждой строке*
Вариант 7	Создать класс типа - **Окружность**. Поля – координаты центра, радиус. Методы вычисляют площадь, длину окружности. Свойства должны проверять корректность задаваемых параметров. *Создать массив объектов. Вывести:* *a) группы окружностей, центры которых лежат на одной прямой;* *b) наибольший и наименьший по площади (периметру) объект;*
Вариант 8	Создать класс типа - **Прямоугольник.** Поля - высота и ширина. Методы вычисления площади, периметра, Свойства должны проверять корректность задаваемых параметров. *Создать массив объектов. Вывести:* *a) количество четырехугольников разного типа (квадрат, прямоугольник, ромб, произвольный)* *b) определить для каждой группы наибольший и наименьший по площади (периметру) объект.*
Вариант 9	Определить класс **Вектор**. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. *Создать массив объектов. Вывести:* *a) вектора с заданным модулем;* *b) определить вектор с наибольшей/наименьшей суммой элементов.*
Вариант 10	Построить класс Булев вектор (**BoolVector**). Реализовать методы для выполнения поразрядных конъюнкции, дизъюнкции и отрицания векторов, а также подсчета числа единиц и нулей в векторе. *Создать массив объектов. Вывести:* *a) вектора с заданным числом единиц/нулей;* *b) определить и вывести равные вектора.*

Вариант 11	Создать класс типа - **Время** с полями: часы (0-23), минуты (0-59), секунды (0-59). Свойства класса должны проверять корректность задаваемых параметров. *Создать массив объектов. Вывести:* *a) время с заданным числом часов;* *b) список времен по группам: ночь, утро, день, вечер.*
Вариант 12	Создать класс - **Одномерный массив** целых чисел (вектор). Создать индексатор, методы поэлементного сложения и вычитания со скалярным значением. *Создать массив объектов. Вывести:* *a) массивы только с четными/нечетными элементами;* *b) массив с наибольшей суммой элементов.*
Вариант 13	Создать класс – **Треугольник**, заданного тремя точками (Создать класс Point). Методы класса обеспечивают вывод на экран координат, рассчитывают длины сторон и периметр треугольника. *Создать массив объектов.* *a) подсчитать количество треугольников разного типа (равносторонний, равнобедренный, прямоугольный, произвольный).* *b) определить для каждой группы наибольший и наименьший по периметру объект.*
Вариант 14	Создать класс - **Множество**. Методы класса реализуют добавление и удаление элемента, пересечение и разность множеств. *Создать массив объектов. Вывести:* *a) множества только с четными/нечетными элементами;* *b) множества, содержащие отрицательные элементы.*
Вариант 15	Создать класс **Airline**: Содержит : Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Свойства и конструкторы должны обеспечивать проверку корректности. *Создать массив объектов. Вывести:* *a) список рейсов для заданного пункта назначения;* *b) список рейсов для заданного дня недели;.*
Вариант 16	Создать класс **Student**: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс, Группа. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод расчет возраста студента *Создать массив объектов. Вывести:* *a) список студентов заданного факультета;* *d) список учебной группы.*

Вариант 17	Создать класс **Customer**: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, баланс. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить методы зачисления и списания сумм на счет. *Создать массив объектов. Вывести:* *a) список покупателей в алфавитном порядке;* *b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.*
Вариант 18	Создать класс **Abiturient**: id, Фамилия, Имя, Отчество, Адрес, Телефон, массив оценок. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить методы расчёта среднего балла, поиска максимального и минимального балла *Создать массив объектов. Вывести:* *a) список абитуриентов, имеющих неудовлетворительные оценки;* *b) список абитуриентов, у которых сумма баллов выше заданной;*
Вариант 19	Создать класс **Book**: id, Название, Автор (ы), Издательство, Год издания, Количество страниц, Цена, Тип переплета. Свойства и конструкторы должны обеспечивать проверку корректности. *Создать массив объектов. Вывести:* *a) список книг заданного автора;* *b) список книг, выпущенных после заданного года.*
Вариант 20	Создать класс **House:** id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод расчета возраста задания (необходимость в кап. ремонте). *Создать массив объектов. Вывести:* *a) список квартир, имеющих заданное число комнат;* *b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;*

Вариант 21	Создать класс **Phone**: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод расчет баланса на текущий момент. *Создать массив объектов. Вывести:* *a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное;* *b) сведения об абонентах, которые пользовались междугородной связью;*
Вариант 22	Создать класс **Car**: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод вывода возраста машины. *Создать массив объектов. Вывести:* *a) список автомобилей заданной марки;* *b) список автомобилей заданной модели, которые эксплуатируются больше n лет;*
Вариант 23	Создать класс **Product**: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод вывода общей суммы продукта. *Создать массив объектов. Вывести:* *a) список товаров для заданного наименования;* *b) список товаров для заданного наименования, цена которых не превосходит заданную;*
Вариант 24	Создать класс **Train**: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс). Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод вывода общего числа мест в поезде. *Создать массив объектов. Вывести:* *a) список поездов, следующих до заданного пункта назначения;* *b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;*

Вариант 25	Создать класс **Bus**: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Свойства и конструкторы должны обеспечивать проверку корректности. Добавить метод вывода возраста автобуса. *Создать массив объектов. Вывести:* *a) список автобусов для заданного номера маршрута;* *b) список автобусов, которые эксплуатируются больше заданного срока;*
Вариант 26	Создать класс **Airline**: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Свойства и конструкторы должны обеспечивать проверку корректности. *Создать массив объектов. Вывести:* *a) список рейсов для заданного пункта назначения;* *b) список рейсов для заданного дня недели;*

Вопросы

1. Назовите принципы ООП. Поясните каждый из них.
2. Назовите класс .NET, от которого наследуются все классы.
3. Охарактеризуйте открытые методы System.Object.
4. Охарактеризуйте закрытые методы System.Object.
5. Приведите пример определения класса.
6. Какие ключевые слова можно использовать при определении класса?
7. В чем отличие между объектом и классом?
8. Что такое конструктор? Когда вызывается конструктор?
9. Перечислите свойства конструктора?
10. Что такое деструктор (destructor) ?
11. Что такое this?
12. Что будет выведено в результате выполнения

```
class A
        {
            private int _num;
            public A(int num) { Num = num; }
            public int Num { get { return _num; } set { _num = value; } }
        }

        static void Main(string[] args)
        {
            A a = new A(5);
            A b = a;
            Console.WriteLine(a.Num + " "  +b.Num);

            a.Num = 7;
            Console.WriteLine(a.Num + " " + b.Num);
        }
```

13. Какие спецификаторы доступа для класса и методов класса существуют в C#?
14. Опишите модификатор protected internal.
15. Зачем и как используются ref и out параметры функции?
16. Приведите пример необязательных и именованных параметров метода.
17. Приведите пример полей класса – статические, константные, только для чтения.
18. Приведите пример определения свойств класса. Как свойства связаны с инкапуляцией?
19. Назовите явное имя параметра, передаваемого в метод set свойства класса?
20. Что такое автоматические свойства?
21. Что такое индексаторы класса? Какие ограничения существуют на индексатор?
22. Что такое перегруженный метод?
23. Что такое partial класс и какие его преимущества?
24. Что такое анонимный тип в C#?
25. Для чего делают статические классы?
26. В чем отличие статического поля от экземплярного?
27. Поясните работу статических конструкторов.
28. Какая разница между поверхностным (shallow) и глубоким (deep) копированием?
29. В чем разница между равенством и тождеством объектов?
30. Что такое частичные классы и частичные методы?
31. Что будет выведено на консоль результате выполнения следующего кода:

```
class Program
    {
        static void Main(string[] args)
        {
            var age = 15;
            Type ageType = age.GetType();
            Console.WriteLine(ageType);

        }
    }
```

32. Что будет выведено на консоль результате выполнения следующего кода:

```
class MyClass
    {
        static void Main()
        {
            int a = 1, b = 2;
            change(ref a, ref b);
            Console.WriteLine("a=" + a + ", b=" + b);
            Console.ReadLine();
        }

        private static void change(ref int a, ref int b)
        {
            int c = a;
            a = b;
            b = c;
        }
    }
```

33. Пусть задан следующий класс.

```
internal class A
        {
            public A() { }  //1
            public int A() { } //2
            public A(int someI) { } //3
            public A(A somA) { } //4
        }
```

Какой из конструкторов задан неверно?

34. Пусть задан следующий класс.

```
class Motorcycle
{
    private int driverIntensity;
    private string driverName;
    ссылок: 0
    public Motorcycle(int intensity = 0, string name = "")
    {
        if (intensity > 10)
        {
            intensity = 10;
        }
        driverIntensity = intensity;
        driverName = name;
    }
}
```

Сколько аргументов может быть задано при вызове конструктора данного класса?

35. Почему не удается создать объект класса A?

```
internal class A
        {
            A() { }  //1
            A(String st) { } //2
            A(int a) { } //3
            public A(int a, int b) { } //4
        }
        static void Main(string[] args)
        {
            A obj = new A(5);
        }
```

36. Что будет выведено в консоль при выполнении данной программы?

```
class A
    {
        static A() { Console.WriteLine("A static "); }
        public A() { Console.WriteLine("A "); }
    }


    class Program
    {
        static void Main()
        {
            new A();
        }
    }
```

37. Какая строка приведенного далее класса вызовет ошибку компиляции?

```
class Points
    {
        public readonly int a =10;
        public static readonly Int32 b = new Int32();
        public static string c = "New";
        private int d;

        public Points()
        {
            c = "Method"; //1
            a = 20;//2
            b = 30;//3
        }
    }
```

Краткие теоретические сведения

Классы, член данные и член функции класса

Класс это абстрактный тип данных, определяемый программистом (пользователем). С помощью классов определяются свойства объектов. Объекты это экземпляры класса. Объявление класса синтаксически имеет следующий вид:

```
class имя_класса
    {
        // члены класса
    }
```

Члены класса – это данные и функции для работы с этими данными.
Имя класса – это, по сути дела, имя нового типа данных.

Создание экземпляра (объекта) класса осуществляется с помощью оператора new:

Имя_класса имя_объекта = new имя_класса();

Доступ к членам класса управляем. Управление доступом осуществляется с помощью спецификаций доступа:

- public – общедоступный член класса.
- private – член класса доступен только внутри данного класса.
- protected – член класса доступен только внутри данного класса и внутри классов, производных от данного.
- internal – член класса доступен только внутри данной сборки (программы).

По умолчанию в классе устанавливается спецификация доступа private. Спецификация доступа, отличная от private, должна указываться явно перед каждым членом класса.

Данные класса подразделяются на *поля, константы и события*.
Поле – это обычная член-переменная, содержащая некоторое значение.
Можно, например, объявить класс, членами которого являются только поля:

```
class CA
{
      public           int         x;
      protected  float       z;
                       double      m;
      public           char  sim;
      private          decimal     sum;

}
```

Поле **m** здесь объявлено по умолчанию приватным.

Константы – это поле, объявленное с модификатором const, или, другими словами. Это поле, значение которого изменить нельзя, например:

```
      public     const      int x = 25;
```

Все член функции класса имеют неограниченный доступ ко всем член данным класса независимо от спецификации доступа. Член функции класса в свою очередь подразделяются на:

- методы
- свойства
- конструкторы
- деструкторы
- индексаторы
- операторы.

Методы

В основном, с помощью методов класса осуществляется обработка член данных класса. Другими словами, методы определяют поведение экземпляров данного класса. Методы класса это обычные функции С - стиля. В отличии от функций С, при передаче методу параметров по адресу, необходимо указывать ключевое слово **ref** или **out**. Эти ключевые слова сообщают компилятору, что адреса параметров функции совпадают с адресами переменных, передаваемых в качестве параметров. Любое изменение значения параметров в этом случае приведет к изменению и переменных вызывающего кода. Рекомендуется для входного параметра использовать ключевое слово **ref,** а для выходного параметра ключевое слово **out**, так как параметр функции с ключевым словом **ref** должен быть обязательно проинициализирован перед вызовом функции. При вызове методов указание ключевых слов **ref** и **out** обязательно. Методы могут быть объявлены с ключевым словом static например:

```
public static int minabs(ref int x,ref int y)
{
      //тело функции
}
```

В этом случае для вызова метода имя класса, в котором она определена, и через точку имя метода:

```
Cmin.minabs(ref a,ref b);
```

Точка в C# означает принадлежность функции данному классу (в нашем случае Cmin).

ПРИМЕР.

```
using System;

namespace ConsoleApplication12
{
      /// <summary>
      /// Summary description for Class1.
      /// </summary>
      class Cmin
      {
            public static int min(int x,int y)
            {
                  int z = (x<y)?x:y;
                  return z;
            }
            public static int minabs(ref int x,ref int y)
            {
                  x = (x<0)?-x:x;
                  y = (y<0)?-y:y;
                  int z = (x<y)?x:y;
                  return z;
            }

      }
```

```
	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			//
			// TODO: Add code to start application here
			int a=-4;
			int b=2;
			Console.WriteLine("a={0}  b={1}",a,b);
			int k =Cmin.min(a,b);
			Console.WriteLine("a={0}  b={1}",a,b);
			Console.WriteLine("k="+k);
			k =Cmin.minabs(ref a,ref b);
			Console.WriteLine("a={0}  b={1}",a,b);
			Console.WriteLine("k="+k);

			//
		}
	}
}
```

Свойства

Свойства в C# состоят из объявления поля и методов-аксессоров для работы с этим полем.

Эти методы- аксессоры называются получатель (get) и установщик (set). Например, простейшее свойство **y**, работающее с полем **m,** можно представить следующим образом:

```
		private int m=35;
		public  int y
		{
			get
			{
				return m;
			}
			set
			{
				m=value;
			}
		}
```

Свойство, определяется, так же как и поле, но после имени свойства идет блок кода, включающий в себя два метода get и set. Код этих методов может быть сколь угодно сложным, но в нашем случае это всего лишь один оператор. Аксессор **get** всегда возвращает значение того типа, который указан в определении свойства. Аксессор **set** всегда принимает в качестве параметра переменную **value**, которая передается ему неявно. Один из аксессоров может

быть опущен, в этом случае мы получаем поле только для чтения или только для записи.

Обращение к свойству осуществляется точно так же как и к полю.

ПРИМЕР.

```
using System;

namespace ConsoleApplication11
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>

	class CStatic
	{
		private int m=35;
		public  int y
		{
			get
			{
				return m;
			}
			set
			{
				m=value;
			}
		}
	}

	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			//
			// TODO: Add code to start application here
			CStatic p=new CStatic();//создается экземпляр класса
			Console.WriteLine("{0}",p.y);
			p.y=75;
			int z = p.y;
			Console.WriteLine("{0}",z);

			//
		}
	}
}
```

Индексаторы

Индексаторы позволяют приложению обращаться с объектом класса так, как будто он является массивом. Индексатор во многом напоминает свойство,

но в отличие от свойства он принимает в качестве параметра индекс массива. Так как объект класса используется как массив, то в качестве имени класса используется ключевое слово **this**. Определение индексатора синтаксически выглядит следующим образом:

```
public float this[int j]
	{
		get
		{
			//Возврат необходимых данных
		}
		set
		{
			//Установка необходимых данных
		}
	}
```

Пример приложения, использующего индексаторы, приведен ниже:

ПРИМЕР

```
using System;

namespace ConsoleApplication13
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>
	class Rmas
	{
		protected float[] msf=new float[10];
		public float this[int j]
		{
			get
			{
				return msf[j];
			}
			set
			{
				msf[j]=value;
			}
		}

	}
	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			Rmas obj = new Rmas();
			for(int i=0; i<10;i++)
			{
```

```
                        obj[i] = (float)1.5*i;
                }
                for(int i=0; i<10;i++)
                {
                        Console.WriteLine("{0}",obj[i]);
                }
        }
    }
}
```

Конструкторы, поля только для чтения, вызов конструкторов.

Конструктор – это метод класса, имеющий имя класса. Конструкторов в классе может быть несколько или ни одного.

На конструкторы накладываются следующие ограничения:

1. Конструктор не может иметь возвращаемого значения даже **void**
2. Как следствие 1 нельзя использовать оператор **return()**
3. Конструкторы нельзя объявлять виртуальными.

Конструктор автоматически вызывается на этапе компиляции при создании экземпляра данного класса. Попытка вызвать конструктор явным образом вызовет ошибку компиляции.

Различают следующие типы конструкторов:

1. Конструктор по умолчанию
2. Конструктор с аргументами

Конструктор по умолчанию

Конструктор, объявленный без аргументов, называется конструктором по умолчанию.

Если в классе не определен ни один конструктор, то компилятор сам автоматически создает конструктор по умолчанию, который инициализирует все поля класса своими значениями по умолчанию (для числовых значений ноль, для булевской переменной FALSE, для строк пустые ссылки).

Пример:

// В классе CA нет явно объявленных конструкторов

```
using System;

namespace ConsoleApplication15
{
        /// <summary>
        /// Summary description for Class1.
        /// </summary>


        class CA
```

```
      {
            public int x,y,z;
      }

      class Class1
      {
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main(string[] args)

            {
                  //
                  // TODO: Add code to start application here
            CA obj=new CA();
            Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj.x,obj.y,obj.z);
                  //
            }
      }
}
```

Добавим в определение класса CA конструктор по умолчанию:

```
      class CA
      {
            public int x,y,z;
            public CA()
            {
                  x=3;
                  y=2;
                  z=x+y;
            }
      }
```

Конструктор с аргументами

Большинство конструкторов в C# принимают аргументы, с помощью которых разные объекты данного класса могут быть по разному инициализированы.

Пример

```
using System;

namespace ConsoleApplication15
{
      /// <summary>
      /// Summary description for Class1.
      /// </summary>

      class CA
      {
            public int x,y,z;
```

```
            public CA()
            {
                  x=3;
                  y=2;
                  z=x+y;
            }
            public CA(int a,int b)
            {
                  x=a;
                  y=b;
                  z=a+b;
            }
            public CA(int a,int b,int c)
            {
                  x=a;
                  y=b;
                  z=c;
            }

      }

      class Class1
      {
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main(string[] args)
            {
                  //
                  // TODO: Add code to start application here
            CA obj=new CA();
            Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj.x,obj.y,obj.z);
            CA obj1=new CA(5,7);
            Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj1.x,obj1.y,obj1.z);
            CA obj2=new CA(5,7,25);
            Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj2.x,obj2.y,obj2.z);

                  //
            }
      }
}
```

Поля только для чтения

Классы могут содержать поля только для чтения. Поле только для чтения – это константное поле, значение которого изменить нельзя. В отличие от обычных констант, значение которых задается непосредственно в тексте программы, начальное значение поля только для чтения может быть вычислено в процессе выполнения приложения. Начальное значение поля

только для чтения может быть установлено только внутри конструктора. Для объявления поля для чтения используется ключевое слово readonly.

Пример:

```
using System;

namespace ConsoleApplication15
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>

	class CA
	{
		public int x,y,z;
		public readonly int rd;
		public CA()
		{
			x=3;
			y=2;
			z=x+y;
			rd=x+y+z;

		}
		public CA(int a,int b)
		{
			x=a;
			y=b;
			z=a+b;
			rd=x+y+z;

		}
		public CA(int a,int b,int c)
		{
			x=a;
			y=b;
			z=c;
			rd=x+y+z;
		}

	}

	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			//
			// TODO: Add code to start application here
		CA obj=new CA();
		Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj.x,obj.y,obj.z);
```

```
            CA obj1=new CA(5,7);
            Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj1.x,obj1.y,obj1.z);
            CA obj2=new CA(5,7,25);
            Console.WriteLine("x={0,2} y={1,2}
z={2,2}",obj2.x,obj2.y,obj2.z);
            Console.WriteLine("поля для чтения{0,2}  {1,2} {2,2}",
obj.rd, obj1.rd, obj2.rd);
                  //
            }
      }
}
```