Лабораторная работа № 5

Классы в C#, методы классов

Цель работы: изучить описание методов классов, функций-членов в C#, их создание в C# и некоторые алгоритмы их обработки.

Теоретические сведения

Пример 1. Простой класс *Time*

```
using System;
public class Time
{
    // открытые методы
    public void DisplayCurrentTime()
    {
        Console.WriteLine("заглушка для DisplayCurrentTime");
    }

    // закрытые переменные
    int Year;
    int Month;
    int Date;
    int Hour;
    int Minute;
    int Second;
}
public class Tester
{
    static void Main()
    {
        Time t = new Time();
        t.DisplayCurrentTime();
    }
}
```

Единственный метод, объявленный в определении класса *Time*, — метод *DisplayCurrentTime()*. Тело этого метода определено внутри определения класса. C# не требует объявления методов до их описания. В C# все методы встраиваются в определение класса, как метод *DisplayCurrentTime()*. В определении метода *DisplayCurrentTime()* указано, что тип возвращаемого значения — void, то есть он не возвращает значения методу, вызвавшему его. В конце определения класса *Time* объявляется ряд переменных: *int Year, int Month, int Date, int Hour, int Minute, int Second*.

После закрывающей фигурной скобки определен другой класс, *Tester*. Он включает в себя метод *Main()*. В методе *Main()* создается экземпляр класса *Time*, и его адрес присваивается объекту t. Поскольку t является экземпляром *Time*, метод *Main()* может вызывать метод *DisplayCurrentTime()*, предоставляемый объектами этого типа, и тем самым выводить на экран текущее время.

Желательно определять переменные класса как закрытые (private). Это гарантирует, что только методы того же класса будут иметь доступ к их значениям. Поскольку уровень доступа private устанавливается по умолчанию, нет необходимости указывать его явно.

Так, в примере 1 объявления переменных лучше переписать следующим образом:

```
// закрытые переменные
int Year;
int Month;
int Date;
int Hour;
int Minute;
int Second;
```

Класс *Time* и метод *DisplayCurrentTime()* объявлены открытыми, чтобы любой другой класс мог обратиться к ним.

Аргументы метода

Методы могут принимать любое количество параметров. Список параметров указывается в круглых скобках после имени метода, причем каждый параметр предваряется своим типом. Например, в следующе участке кода определяется метод по имени *MyMethod*, который возвращает void (то есть не возвращает значения). Он имеет два параметра с типами *int* и *button*:

```
void MyMethod(int firstParam, button secondParam)
{
    // ..
}
```

В теле метода параметры ведут себя как локальные переменные, как будто бы они были объявлены и инициализированы значениями, переданными методу. В примере 2 иллюстрируется передача значений методу. В этом случае значения имеют типы *int* и *float*.

Пример 2. Передача значений методу *SomeMethod()*

```
using System;
public class MyClass
{
    public void SomeMethod(int firstParam, float secondParam)
    {
        Console.WriteLine("Получены параметры {0}, {1}", firstParam, secondParam);
    }
}
public class Tester
{
    static void Main()
    {
        int howManyPeople = 5;
        float pi = 3.14f;
        MyClass me = new MyClass();
        me.SomeMethod(howManyPeople, pi);
    }
}
```

Здесь метод *SomeMethod()* принимает значения типов *int* и *float* и выводит их с помощью метода *Console.WriteLine()*. Параметры, названные *firstParam* и *secondParam*, в теле метода *SomeMethod()* трактуются как локальные переменные.

Создание объектов

Базовые типы C# (int, char и т. д.) являются размерными типами: они хранятся в стеке. Объекты, со своей стороны, имеют ссылочный тип и хранятся в куче, а создаются с помощью ключевого слова *new*. Здесь переменная t, на самом деле, не содержит значение объекта *Time*. Вместо этого она содержит адрес объекта (безымянного), созданного в куче. Сама же переменная t является лишь ссылкой на этот объект.

Конструкторы

Обратите внимание на оператор, создающий объект *Time* в примере 1. Внешне он выглядит как вызов метода. И действительно, при создании экземпляра происходит обращение к методу. Он называется конструктором, и программист должен либо определить его как часть определения класса, либо положиться в этом на среду CLR, которая сама его предоставит. Задача конструктора — создать объект указанного класса и перевести его в действующее состояние. До вызова конструктора объект представляет собой неинициализированную область памяти; по окончании его работы эта память содержит действующий экземпляр данного класса.

Класс Time из примера 1 не определяет никакого конструктора. Если конструктор не объявлен, компилятор предоставляет его самостоятельно. Конструктор, вызванный по умолчанию, создает объект, но не предпринимает никаких других действий. Переменные класса инициализируются безвредными значениями (целые — нулем, строки — пустой строкой и т. д.).

Как правило, программисты предпочитают определять собственные конструкторы и снабжать их аргументами, чтобы конструктор мог устанавливать начальное состояние объекта. В примере 1 было бы разумно передавать конструктору информацию о текущем годе, месяце, дне месяца и т. д., чтобы объект был заполнен осмысленными данными. При определении конструктора объявляется метод с тем же именем, что и класс, в котором он определяется. У конструкторов нет возвращаемого типа, и они обычно объявляются открытыми. Если планируется передавать конструктору какие-либо аргументы, их список указывается, как для любого другого метода. В примере 3 объявляется конструктор для класса *Time*, который принимает единственный аргумент, объект типа *DateTime*.

Пример 3. Объявление конструктора

```
public class Time
{ // открытые методы доступа
    public void DisplayCurrentTime()
    {
```

```
        System.Console.WriteLine("{0},{1},{2},{3},{4},{5}", Month, Date, Year,
Hour, Minute, Second);
    }
    // конструктор
    public Time(System.DateTime dt)
    {
        Year = dt.Year;
        Month = dt.Month;
        Date = dt.Day;
        Hour = dt.Hour;
        Minute = dt.Minute;
        Second = dt.Second;
    }
    // закрытые переменные класса
    int Year;
    int Month;
    int Date;
    int Hour;
    int Minute;
    int Second;
}
public class Tester
{
    static void Main()
    {
        System.DateTime currentTime = System.DateTime.Now;
        Time t = new Time(currentTime);
        t.DisplayCurrentTime();
    }
}
```

В этом примере конструктор принимает объект *DateTime* и инициализирует все переменные класса, беря за основу значения из этого объекта.

После того как конструктор закончит свою работу, в памяти компьютера будет находиться объект *Time* с инициализированными значениями. Когда метод *Main()* вызывает метод *DisplayCurrentTime()*, эти значения выводятся на экран. Закомментируйте какое-нибудь присваивание и снова выполните программу. Окажется, что компилятор инициализирует соответствующую переменную нулем. Целые переменные класса устанавливаются в 0, если не указано другое значение. Помните, что переменные, имеющие размерный тип (например, целочисленные), не могут оставаться неинициализированными; если конструктор не получит конкретных указаний, он установит значения по умолчанию.

В примере 3 в методе *Main()* класса *Tester* создается объект *DateTime*. Этот объект, поставляемый библиотекой *System*, предлагает ряд открытых

значений: *Year, Montn, Day, Hour, Minute* и *Second*, которые в точности соответствуют переменным объекта *Time*. Кроме того, объект *DateTime* предоставляет статический метод *Now()*, который возвращает ссылку на экземпляр объекта *DateTime*, инициализированный текущим временем.

Рассмотрим строчку в методе *Main()*, где объект *DateTime* создается вызовом статического метода *Now()*. Он создает *DateTime* в куче и возвращает ссылку на него.

Возвращенная ссылка присваивается переменной *currentTime*, которая объявлена как ссылка на объект *DateTime*. Затем *currentTime* передается в качестве параметра конструктору *Time*. Параметр этого конструктора, *dt*, тоже представляет собой ссылку на объект *DateTime*. Теперь *ct* ссылается на тот же объект, что и переменная *currentTime*. Таким образом, конструктор *Time* получает доступ к открытым переменным объекта *DateTime*, созданного в методе *Main()*. Когда объект передается в качестве параметра, он передается по ссылке, то есть передается указатель на объект, а копия объекта не создается.

Инициализаторы

Вместо того чтобы инициализировать значения переменных класса в каждом конструкторе, можно сделать это в инициализаторе. Инициализатор создается присваиванием начального значения элементу класса:

```
private int Second = 30; // инициализатор
```

Предположим, семантика объекта Time такова, что независимо от установленного времени секунды всегда инициализируются значением 30.

Класс Time можно переписать с применением инициализатора. Тогда, какой бы конструктор ни был вызван, переменная Second будет всегда получать начальное значение либо явно, от конструктора, либо неявно, от инициализатора. Это продемонстрировано в примере 4.

Пример 4. Использование инициализатора

```
public class Time
{
    // открытые методы доступа
    public void DisplayCurrentTime()
    {
        System.DateTime now = System.DateTime.Now;
        System.Console.WriteLine("\nОтладка\t: {0}.{1}.{2} {3}:{4}:{5}",now.Day,
now.Month, now.Year, now.Hour, now.Minute, now.Second);
    }
    //конструкторы
    public Time(System.DateTime dt)
    {
        Year = dt.Year;
        Month = dt.Month;
        Date = dt.Day;
        Hour = dt.Hour;
        Minute = dt.Minute;
        Second = dt.Second;//явное присваивание
    }
    public Time(int Year, int Month, int Day, int Hour, int Minute)
    {
        this.Year = Year;
        this.Month = Month;
        this.Date = Day;
        this.Hour = Hour;
        this.Minute = Minute;
    }
    // закрьгые переменные класса
    private int Year;
    private int Month;
    private int Date;
    private int Hour;
    private int Minute;
    private int Second = 30; // инициализатор
}
public class Tester
{
    static void Main()
    {
        System.DateTime currentTime = System.DateTime.Now;
        Time t = new Time(currentTime);
        t.DisplayCurrentTime();
        Time t2 = new Time(2007, 02, 20, 17, 24);
        t2.DisplayCurrentTime();
    }
}
```

Если инициализатор не указан, конструктор присвоит каждой переменной нулевое начальное значение. Однако в приведенном примера переменная *Second* получает значение 30:

```
private int Second = 30; // инициализатор
```

Если для переменной *Second* не будет передано значение, то при создании она будет проинициализирована числом 30:

```
Time t2 = new Time(2007, 02, 20, 17, 24);
t2.DisplayCurrentTime();
```

Однако если переменной *Second* значение присвоено, как это делается в конструкторе, принимающем объект *DateTime*, новое значение замещает первоначальное.

При первом выполнении программы вызывается конструктор, принимающий объект *DateTime*, причем секунды устанавливаются в значение 24. Если бы у программы не было инициализатора и переменной *Second* не присваивалось никакого значения, то компилятор установил бы ее в ноль.

Копирующие конструкторы

Копирующий конструктор (copy constructor) создает новый объект, копируя переменные из существующего объекта того же типа. Пусть, например, требуется передать объект *Time* конструктору *Time()* так, чтобы новый объект *Time* содержал те же значения, что и старый. Язык C# не добавляет в класс копирующий конструктор, так что программист должен написать такой конструктор самостоятельно. Подобный конструктор всего лишь копирует элементы исходного объекта во вновь создаваемый:

```
public Time(Time existingTimeObject)
{
    Year = existingTimeObject.Year;
    Month = existingTimeObject.Month;
    Date = existingTimeObject.Date;
    Hour = existingTimeObject.Hour;
    Minute = existingTimeObject.Minute;
    Second = existingTimeObject.Second;
}
```

Копирующий конструктор вызывается путем создания объекта типа *Time* и передачи ему имени копируемого объекта *Time*:

```
Time t3 = new Time(t2);
```

Здесь переменная t2 передается в качестве аргумента *existingTimeQbject* копирующему конструктору, который создаст новый объект *Time.*

Задание на лабораторную работу.

Формулы для разложения в ряд Тейлора:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}, x \in R,$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^{m-1}\frac{x^{2m-1}}{(2m-1)!}, x \in R,$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots + (-1)^{m}\frac{x^{2m}}{(2m)!}, x \in R,$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots + (-1)^{n-1}\frac{x^n}{n}, x \in [-1;1],$$

$$arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots + (-1)^{m-1}\frac{x^{2m-1}}{2m-1}, x \in [-1;1].$$

Задание для защиты лабораторных работ модуля 2

Вариант 1

Описать класс для работы с одномерным массивом:

- конструктор, выделяющий память под заданное в его параметре количество элементов массива;
- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции l*n x* для заданного *x*.
- свойство, доступное только для чтения, для получения количества элементов массива, меньших 0,2;
- метод, вычисляющий сумму модулей элементов, расположенных до (левее) первого элемента, равного нулю

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 2

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное количество элементов массива заданным значением;
- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции *cos x* для заданного *x*.
- свойство, доступное только для чтения, для получения количества элементов массива, модуль которых больше 0,5;
- метод, вычисляющий сумму модулей элементов, расположенных после первого элемента, равного нулю

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 3

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное в его параметре количество элементов массива числами Фибоначчи;
- конструктор, заполняющий заданное количество элементов массива из файла с заданным именем;
- свойство, доступное только для чтения, для получения количества элементов массива в диапазоне от 10 до 100;
- метод, вычисляющий сумму модулей элементов, расположенных до (левее) минимального по модулю элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 4

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное в его параметре количество элементов массива случайными числами в заданном диапазоне;
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются ровно одним пробелом);
- свойство, доступное только для чтения, для получения количества элементов массива, равных 50;
- метод, вычисляющий сумму модулей элементов, расположенных до (левее) максимального по модулю элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 5

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное в его параметре количество элементов массива натуральным рядом чисел;

- конструктор, заполняющий заданное количество элементов массива из файла с заданным именем;
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий сумму модулей элементов, расположенных после максимального по модулю элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 6

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное количество элементов массива из файла с заданным именем;
- конструктор, заполняющий элементы массива по заданным начальному и конечному значению и шагу;
- свойство, доступное только для чтения, для получения количества положительных элементов массива;
- метод, вычисляющий сумму элементов массива, расположенных между первым и вторым положительными элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 7

Описать класс для работы с одномерным массивом:

- конструктор, в параметрах которого задаются значения элементов массива (произвольное количество, использовать params);
- конструктор, заполняющий заданное количество элементов массива из файла с заданным именем;
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 8

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное в его параметре количество элементов массива случайными числами в заданном диапазоне;
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются точками);
- свойство, доступное только для чтения, для получения количества элементов массива, равных нулю;
- метод, вычисляющий произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 9

Описать класс для работы с одномерным массивом:

- конструктор, выделяющий память под заданное в его параметре количество элементов массива;
- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции $sinh\ x$ для заданного x.
- свойство, доступное только для чтения, для получения количества элементов массива, меньших 0,9;
- метод, вычисляющий сумму элементов массива, расположенных между первым и последним положительными элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 10

Описать класс для работы с одномерным массивом:

- конструктор, выделяющий память под заданное в его параметре количество элементов массива;
- конструктор, заполняющий заданное в его параметре количество элементов массива квадратами натурального ряда чисел, знак числа задается случайным образом;
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 11

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции e^x для заданного x.
- конструктор, в параметрах которого задаются значения элементов массива (произвольное количество, использовать params);
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 12

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное количество элементов массива заданным значением;
- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции $sin\ x$ для заданного x.
- свойство, доступное только для чтения, для получения количества элементов массива, модуль которых меньше 0,3;
- метод, вычисляющий сумму модулей элементов, расположенных до (левее) последнего элемента, по модулю большего 0,0001.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 13

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий элементы массива из строки string (числа в строке разделяются двоеточиями);
- конструктор, заполняющий заданное в его параметре количество элементов массива натуральным рядом чисел, знак числа задается случайным образом;
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий произведение модулей элементов, расположенных до (левее) первого положительного элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 14

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное в его параметре количество элементов массива квадратами натурального ряда чисел, знак числа задается случайным образом;
- конструктор, заполняющий заданное количество элементов массива из файла с заданным именем;
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий произведение элементов массива, расположенных между максимальным и минимальным элементами.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 15

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное в его параметре количество элементов массива арифметической прогрессией, знак числа задается случайным образом;
- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции $\cosh x$ для заданного x.
- свойство, доступное только для чтения, для получения количества элементов массива, больших 1;
- метод, вычисляющий произведение модулей элементов, расположенных до (левее) последнего отрицательного элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 16

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное в его параметре количество элементов массива геометрической прогрессией, знак числа задается случайным образом;
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются запятыми);
- свойство, доступное только для чтения, для получения количества положительных элементов массива;
- метод, вычисляющий произведение элементов массива, расположенных между максимальным и минимальным элементами

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 17

Описать класс для работы с одномерным массивом:

- конструктор, в параметрах которого задаются значения элементов массива (произвольное количество, использовать params);
- конструктор, заполняющий заданное в его параметре количество элементов массива натуральным рядом чисел, умноженных на заданный коэффициент;
- свойство, доступное только для чтения, для получения количества отрицательных элементов массива;
- метод, вычисляющий произведение модулей элементов, расположенных после последнего элемента, равного нулю.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 18

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное в его параметре количество элементов массива арифметической прогрессией;
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются точкой с запятой);
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий произведение модулей элементов, расположенных после первого элемента, равного нулю.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 19

Описать класс для работы с одномерным массивом:
- конструктор, выделяющий память под заданное в его параметре количество элементов массива и заполняющий их с клавиатуры;
- конструктор, заполняющий заданное в его параметре количество элементов массива натуральным рядом чисел, умноженных на заданный коэффициент;
- свойство, доступное только для чтения, для получения количества отрицательных элементов массива;
- метод, вычисляющий произведение модулей элементов, расположенных до (левее) максимального по модулю элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 20

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное в его параметре количество элементов массива случайными числами в заданном диапазоне;
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются запятыми);
- свойство, доступное только для чтения, для получения количества положительных элементов массива;
- метод, вычисляющий произведение модулей элементов, расположенных после максимального по модулю элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 21

Описать класс для работы с одномерным массивом:
- конструктор, заполняющий заданное количество элементов массива из файла с заданным именем;
- конструктор, заполняющий заданное в его параметре количество элементов массива квадратами натурального ряда чисел;
- свойство, доступное только для чтения, для получения количества элементов массива, больших 10;
- метод, вычисляющий сумму модулей элементов, расположенных до (левее) первого положительного элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 22

Описать класс для работы с одномерным массивом:

- конструктор, в параметрах которого задаются значения элементов массива (произвольное количество, использовать params);
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются запятыми);
- свойство, доступное только для чтения, для получения количества элементов массива, равных нулю;
- метод, вычисляющий сумму модулей элементов, расположенных после первого положительного элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 23

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное в его параметре количество элементов массива числами Фибоначчи;
- конструктор, заполняющий элементы массива из строки string (числа в строке разделяются двоеточиями);
- свойство, доступное только для чтения, для получения количества отрицательных элементов массива;
- метод, вычисляющий сумму модулей элементов, расположенных до (левее) последнего отрицательного элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 24

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное количество элементов массива заданным значением;
- конструктор, заполняющий заданное в его параметре количество элементов массива значениями членов ряда Тейлора для функции $\cosh x$ для заданного x.
- свойство, доступное только для чтения, для получения количества элементов массива, больших 0,8;
- метод, вычисляющий сумму модулей элементов, расположенных после последнего элемента, меньшего по модулю 0,0001.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.

Вариант 25

Описать класс для работы с одномерным массивом:

- конструктор, заполняющий заданное в его параметре количество элементов массива случайными числами в заданном диапазоне;
- конструктор, заполняющий заданное количество элементов массива из строки string (числа в строке разделяются запятыми);
- свойство, доступное только для чтения, для получения количества элементов массива;
- метод, вычисляющий сумму модулей элементов, расположенных после первого отрицательного элемента.

Вывод на экран выполнять только в методе Main класса-клиента. Программа должна адекватно реагировать на ошибки пользователя и различные варианты исходных данных. Все тестовые данные предъявить преподавателю.