Контрольное домашнее задание № 2, модуль 2

Контрольное домашнее задание предполагает самостоятельную домашнюю работу. Вам потребуется:

1. Изучить некоторые теоретические материалы самостоятельно.
2. Самостоятельно поработать с документацией по языку C#, в т.ч. осуществлять информационный поиск.
3. Разработать программы, определённые основной задачей и индивидуальным вариантом.
4. Сдать в SmartLMS вовремя заархивированный проект с кодом проекта консольного приложения и библиотеки классов, определённые заданием и вариантом.

Время выполнения

На выполнение работы отводится одна неделя, точные даты выполнения устанавливаются в SmartLMS.

Формат сдачи работы

Для проверки предоставляется решение, содержащие два проекта: консольное приложение и библиотеку классов. Решение должно быть заархивировано и приложено в качестве ответа на задание в SmartLMS.

Общее задание

В одном решении разместить проект библиотеки классов и проект консольного приложения. Подробные описание библиотеки и приложения см. в индивидуальном варианте.

Требования к библиотеке классов

1. Реализации классов не должны нарушать инкапсуляцию данных и принцип единственной ответственности (Single Responsibility Principle).
2. Реализации классов должны содержать регламентированный доступ к данным.
3. Классы библиотеки должны быть доступны за пределами сборки.
4. Каждый нестатический класс обязательно должен содержать, в числе прочих, конструктор без параметров или эквивалентные описания, допускающие его прямой вызов или неявный вызов.
5. Запрещено изменять набор данных (удалять / дополнять), хранящихся в классах или не использовать указанные в задании открытые варианты поведения.
6. Допускается расширение открытого поведения или добавление закрытых функциональных членов класса.
7. Допускается использование абстрактных типов данных, таких как List, ArrayList, Set, Stack, их обобщённых реализаций и проч., но не в качестве замены определённых вариантом структур данных.

8. Поскольку в описаниях классов присутствует «простор» для принятия решений, то каждое такое решение должно быть описано в комментариях к коду программы. Например, если выбран тип исключения, то должно быть письменно обоснованно, почему вы считаете его наиболее подходящим в рамках данной задачи.

Требования к консольному приложению

1. Предусмотреть проверку корректности для каждого ввода данных, обработку исключений, в т.ч. порождаемых классами библиотеки, организацию повторения решения.
2. Допускается использование абстрактных типов данных, таких как List, ArrayList, Set, Stack, их обобщённых реализаций и проч., но не в качестве замены определённых вариантом структур данных.
3. Все данные приложения читают из текстовых файлов, результат работы **и** выводится на экран, **и** сохраняется в другом текстовом файле. Файлы размещаются обязательно рядом с исполняемым файлом консольного приложения (.EXE). Имена входного и выходного файлов вводятся пользователем с клавиатуры.

Общие требования к работе

1. Цикл повторения решения и проверки корректности получаемых данных обязательны.
2. Соблюдение определённых программой учебной дисциплины требований к программной реализации работ – обязательно.
3. Соблюдение соглашений о качестве кода – обязательно (https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/coding-style/coding-conventions).
4. Весь программный код должен быть написан на языке программирования C# с учётом использования .net 6.0;
5. исходный код должен содержать комментарии, объясняющие неочевидные фрагменты и решения, резюме кода, описание целей кода (см. материалы лекции 1, модуль 1);
6. при перемещении папки проекта библиотеки (копировании / переносе на другое устройство) файлы должны открываться программой также успешно, как и на компьютере создателя, т.е. по относительному пути;
7. текстовые данные, включая данные на русском языке, успешно декодируются при представлении пользователю и человекочитаемы;
8. программа не допускает пользователя до решения задач, пока с клавиатуры не будут введены корректные данные;
9. консольное приложение обрабатывает исключительные ситуации, связанные (1) со вводом и преобразованием / приведением данных как с клавиатуры, так и из файлов; (2) с созданием, инициализацией, обращением к элементам массивов и строк; (3) вызовом методов библиотеки.
10. представленная к проверке библиотека классов должна решать все поставленные задачи, успешно компилироваться.

Вариант 10

В библиотеке классов объявить класс **JaggedMatrix**

- Поле **x** – массив вещественных значений
- **Double[][] arr** – каждый массив – значения членов ряда Маклорена, получаемых при разложении функции **cos(x)** в ряд, при заданном значении аргумента **x** (см. метод **CosArray()**)

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

- Конструктор с параметрами **min**, **max, n** (количество элементов массивов **arr** и **x**). Конструктор создаёт массив **x** из **n** случайных вещественных значений из диапазона (**min**, **max**) не включая границы. По значению элемента **x[i]** формируется, с помощью метода **CosArray(),** каждая ссылка массив по ссылке **arr[i]**.
- Метод **CosArray()** – закрытый метод, формирующий вещественный массив, содержащий значения элементов разложения функции **cos(x)** в ряд Маклорена в точке **x0** (параметр метода), количество элементов разложения определяется неразличимостью для компьютера текущей суммы ряда и полученной на предыдущем шаге вычислений (машинная точность).
- Метод **AsStrings()** – открытый метод, формирующий массив строк, каждая из которых представляет элементы массивов **arr**. Форматирование задавать в формате экспоненты, точность – **3** знака после десятичного разделителя.

В основной программе, используя библиотеку классов, создать объект класса **JaggedMatrix**. Количество элементов **n** во внутреннем массиве **x** и значения **min**, **max** – получать из входного файла. Значения элементов массива массивов **arr** и объекта класса **JaggedMatrix** вывести с выходной файл. Для формирования строк использовать метод **AsStrings()**.