Проект, модуль 2

Проект предполагает самостоятельную домашнюю работу по разработке консольного приложения.
Вам потребуется:

1. Изучить предложенные 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 заархивированную папку с решением Visual Studio, где содержится проект(ы) с решением вашей задачи.

Время выполнения работы
Определяется датами, назначенными в SmartLMS.
Что сдаём?
На проверку предоставляется заархивированная папка с решением Visual Studio, в которое включен(ы) проект(ы), через который(е) реализована программа.

Общие требования к работе (для всех вариантов)

В индивидуальном варианте вы обнаружите файл с CSV-данными, изучив которые вы спроектируете нестатические классы, для представления этих данных в своей программе. Каждый класс следует размещать в отдельном файле с исходным кодом.

Предупреждение: В данных присутствуют номера строк, они не должны попадать в ваши объекты, но будут присутствовать в файлах, которые создаёт ваша программа, чтобы структура файла соотвестствовала условию.

Ваша программа представляет собой справочную систему, которая за счёт использования экранного меню и диалогов с пользователем, предоставляет возможность работы с данными, представленными в csv-формате. Программа должна обеспечивать выполнение действий, определённых общими требованиями и индивидуальным вариантом работы.

При работе с данными:

1. Программа должна сохранять работоспособность при вводе некорректных адресов и имён файлов, с учётом различных платформ запуска файлов.
2. Некорректно структурированный файл вашей программой не обрабатывается, пользователю выводится сообщение об ошибке и выводится экранное меню.
3. Программа обязательно должна корректно открывать созданные ей файлы и позволять выполнять над ними все операции из меню.
4. Программа автоматически разбирается с кодировками файла и отображает данные на экран и в файлы в человекочитаемом виде.
5. В данных есть пропуски и пустые поля. Выводить на экран такие данные не нужно, но их требуется учитывать, если пропуски встречаются при сортировках или фильтрах. Например, размещать вначале или конце списка при сортивке.

Пример экранного меню

Прямой порядок сортировки – по возрастанию для числовых значений; по алфавиту для строк и символов. Обратный – по убыванию.

```
Введите  номер пункта меню для запуска действия:
    1. Загрузить данные из файла
    2. Произвести выборку по значению Area
    3. Произвести выборку по значению Name
    4. Произвести выборку по значению Area и Name
    5. Отсортировать таблицу по значению Year (прямой порядок)
    6. Отсортировать таблицу по значению Name (прямой порядок)
    7. Выйти из программы
```

При структуризации программы и программного кода:

1. В этой работе мы опираемся на создание нестатических классов и принципы объектно-ориентированного программирования (ООП).
2. Разделите программный код по отдельным файлам, т.е. каждый класс должен быть реализован в отдельном файле с исходным кодом.

Ограничения

В основной и дополнительной задачах требуется, чтобы:

- имена классов соответствовали соглашениям об именовании;
- спроектированные классы должны соответствовать принципу единственной ответственности (**single responsibility principle**) и не нарушать инкапсуляцию (посмотрите пример через переводчик Яндекс https://translated.turbopages.org/proxy_u/en-ru.ru.f0dd9ac0-672eef33-d66451ef-74722d776562/https/www.c-sharpcorner.com/article/solid-single-responsibility-principle-with-c-sharp/ кнопка в правом верхнем углу позволит переключиться в исходную статью на английском);
- запрещено использование готовых библиотек, Nuget-пакетов и проч., для получения, записи и обработки данных из CSV-файла;
- групповые операции над данными следует реализовать самостоятельно, не прибегая с возможностям LINQ.

Комментарии по работе с CSV файлом

Предупреждение: Просмотр файла табличным процессором, например Excel, может привести к нарушению csv-структуры данных и некорректной работе программы.

В представленных заданиях для обработки используются CSV-файлы, которые служат для представления таблиц в текстовом виде. Заголовки таблиц - первая строка файла. Записи (строки таблицы) располагаются по строкам файла. Данные (ячейки) в рамках одной строки файла отделяются запятой или точкой с запятой. Подробности про эти файлы можно найти в последней лекции первого модуля дисциплины «Программирование на C#».

Вариант 1. Неизученные свойства покемонов

Первичные данные для работы программы находятся в файле **Pokemon.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **Pokemon.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Pockemon**.
Используйте разработанные классы и объекты для реализации следующих операций текстового экранного меню:

1. Вводить адрес файла, из которого загружаются данные о покемонах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **Pokemon.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех покемонах, имеющих свойство *Type 1 = Poison*.
3. Сохранять в файл **Pokemon-Poison.csv** выборку обо всех покемонах, имеющих свойство *Type 1* или *Type 2* равное значению, введённому пользователем с клавиатуры. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество строк с данными о покемонах в файле (без учёта строки заголовков).
   4.2. Количество групп покемонов по типам (свойство *Type 1*) и количество покемонов в каждой группе.
   4.3. Количество покемонов по конкретному типу, тип вводится с клавиатуры пользователем. Обратите внимание, что покемоны могут быть иметь разный тип по разным свойствам *Type 1* и *Type 1*.
   4.4. Количество ядовитых (*Poison*) покемонов по основному типу *Type 1*, которые являются драконами (*Flying*) по типу *Type 2*.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран выборку покемонов, не имеющих дополнительных особенностей (*Type 2* отсутствует / пуст). Перед выводом перечня покемонов должно выводиться среднее значение здоровья (*HP*) покемонов этой выборки.
   6.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о покемонах, в котором выделены группы по полю основной тип (*Type 1*), при этом в каждой группе следует упорядочить покемонов по возрастанию атаки (*Attack*). Перед упорядоченным набором должна выводиться строка с указанием модуля разницы (дельту) между максимальной и минимальной атакой в группе покемонов *Type 1 = Dark*.
   7.1. Сохранять результат переупорядочения в файле **Sorted-Pokemon.csv** с сохранением порядка.

Вариант 2. Легенда о покемонах

Первичные данные для работы программы находятся в файле **Pokemon.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **Pokemon.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Pockemon**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о покемонах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **Pokemon.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию о группах покемонов, являющихся легендарными и нелегендарными (легендарность устанавливается по полю *Legendary*).
   2.1. Сохранять в файлы **Pokemon-Legendary.csv** и **Pokemon-Usual.csv** списки легендарных и нелегендарных покемонов соответственно.
3. Выводить на экран список покемонов, относящихся к одному и тому же поколению (*Generation*). Перед каждой группой поколения выводить строку с данными о самом мощном покемоне (*Attack* в рамках поколения).
   3.1. Сохранять перечень покемонов поколения в файле **Pokemon-Gen-N.csv,** где *N* - номер поколения.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество покемонов по поколениям (*Generation*).
   4.2. Полные данные о самом мощном (*Attack*) и самом слабом покемоне среди всех покемонов, загруженных из файла.
   4.3. Количество ядовитых (*Poison*) жуков (*Bug*). Обратите внимание, что эти свойства могут быть выражены разными полями *Type*.
   4.4. Количество покемонов второго поколения, у которых защита меньше 50.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран выборку покемонов, у которых разброс между атакой и защитой составляет более 15 единиц.
   6.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о покемонах, в котором выделены группы по полю скорость (*Speed*), при этом в каждой группе следует упорядочить покемонов по возрастанию здоровья (*HP*). Перед упорядоченным набором должна выводиться строка с диапазона здоровья в указанной группе, диапазон задается минимальным и максимальным значением по группе.
   7.1. Сохранять результат переупорядочения в файле **Sorted-Pokemon.csv** с сохранением порядка.

Вариант 3. Потомственные студенты

Первичные данные для работы программы находятся в файле **student_data.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **student_data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Student**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о студентах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **student_data.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран информацию обо всех студентах, чьи родители учились на оценки выше или равные 4 (столбцы Medu и Fedu).
   2.1. Сохранять в файл **student-parents-education.csv** выборку обо всех студентах, чьи родители подходят под вышеуказанное условие.
3. Выводить на экран информацию обо всех студентах, количество пропусков (столбец absences) у которых находится в диапазоне [*min* + 10, *max* - 10], где *min* - минимальное количество пропусков в выборке, *max* - максимальное.
   3.1. Сохранять в файл **students-absences.csv** выборку обо всех студентах, количество пропусков у которых находится в вышеуказанном диапазоне.
4. Выводить сводную статистику по данным загруженного файла:
   4.1. Общее количество данных о студентах в наборе, без учёта строки заголовков.
   4.2. Статистику по столбцу *romantic* в процентном отношении. Пример: *yes 63% no 37%*.
   4.3. Средний возраст девушек, у которых хотя бы один из родителей работает из дома (столбцы *Mjob* и *Fjob*).
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран исходный набор данных о студентах, сгруппированный по столбцу reason, при этом в каждой группе следует упорядочить студентов по убыванию значений столбца age.
   6.1. Сохранять в файл **grouped-students.csv** результат группировки с сохранением порядка.
7. **[дополнительная задача]** Выводить на экран исходный набор данных о студентах, сгруппированный по признаки платности обучения (*paid*), при этом внутри каждой группы студенты упорядочены по средней оценке среди *G*1, *G*2, *G*3.
   7.1. Сохранять в файл **sorted-paid-students.csv** результат группировки с сохранением порядка.

Вариант 4. “От сессии до сессии…”

Первичные данные для работы программы находятся в файле **student_data.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **student_data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Student**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о студентах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **student_data.csv**.Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран информацию обо всех студентах, у которых средняя оценка за три периода выше одиннадцати (столбцы *G1, G2, G3*).
   2.1. Сохранять выборку в файл **Student-Grades.csv**.
3. Выводить на экран информацию обо всех студентах, у которых значение столбца *freetime* превышает значение столбца *studytime* и при этом количество пропусков (столбец absences) не более семи.
   3.1. Сохранять в файл **Students-Time.csv** выборку обо всех студентах, удовлетворяющих вышеуказанному условию.
4. Выводить сводную статистику по данным загруженного файла:
   4.1. Общее количество данных о студентах, без учёта строки заголовка.
   4.2. Статистику по столбцу *reason* в количественном отношении – одной строкой *reputation = ...; home=....;.....*
   4.3. Средний возраст юношей.
   4.4. Средний возраст девушек.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о студентах. Данные упорядочены по убыванию по времени обучения (*studytime*).
   6.1. Результаты сортировки сохранять в файл с именем **sort-students.csv**.
7. **[дополнительная задача]** Выводить на экран исходный набор данных о студентах, сгруппированный по столбцу с причинами поступления (*reason*), при этом в каждой группе следует упорядочить студентов по возрастанию значений столбца *absences*.
   7.1. Сохраните выборку с сохранением порядка в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню..

Вариант 5. Back in the game

Первичные данные для работы программы находятся в файле **computer_games.csv**. Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **computer_games.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Game**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о компьютерных играх. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **computer_games.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех компьютерных играх, имеющих свойство *Developer = Maxis*
   2.1. Сохранять в файл **Developer_Maxis.csv** выборку обо всех компьютерных играх, имеющих свойство Developer = Maxis.
3. Выводить на экран данные по играм, выходившим к рождественским праздникам (в декабре).
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество игр по каждому производителю (*Producer*)
   4.2. Данные о самой ранней выпущенной игре, причем *Operating System = Microsoft Windows*, а жанр *Genre = First-person shooter*.
   4.3. Информация о среднем значении года выпуска игры (*Date Released*) по жанрам (*Genre*). Иначе говоря, для каждого жанра найти среднее значение года выхода компьютерных игр, которые с ним связаны)
   4.4. Продюсер, который связан с минимальным количеством компьютерных игр.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных об играх, в котором выделены группы по *Operating System* (а именно - *Microsoft Windows* и *Microsoft Windows, macOS*, последняя пара может быть указана в произвольном порядке), при этом в каждой группе следует упорядочить игры по возрастанию *Date Released*. Перед упорядоченным набором должна выводиться строка с указанием модуля разницы (дельту) между максимальной и минимальной датой релиза (*Date Released*).
   6.1. Обратите внимание на формат данных *Date Released*, его нужно привести к единому виду (можно делать, опираясь только на год релиза)
   6.2. Сохранять результат переупорядочения в файлы **Microsoft_Windows_Sort.csv** и **Microsoft_Windows_Mac_Sort.csv** с сохранением порядка.
7. **[дополнительная задача]** Вывести на экран выборку игр, год релиза которых больше среднего значения Date Released.
   7.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.

Вариант 6. Релизы и жанры

Первичные данные для работы программы находятся в файле **computer_games.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **computer_games.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Game**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о компьютерных играх. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **computer_games.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех компьютерных играх, дата выхода которых позже 1997 года.
   2.1. Сохранять в файл **date_Released.csv** выборку обо всех компьютерных играх, имеющих свойство *Date Released* больше 1997.
   2.2. Обратите внимание на формат данных *Date Released*, его нужно привести к единому виду (можно делать, опираясь только на год релиза).
3. Выводить на экран данные по игре, название которой (*Name*) вводит с клавиатуры пользователь. Если такая игра в данных отсутствует, вывести сообщение об этом и экранное меню.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество строк с данными о компьютерных играх (без учёта заголовков)
   4.2. Данные об игре, выпущенной перед последней, причем *Developer* = *Maxis* , а *Producer* = *Electronic Arts*.
   4.3. Информация о среднем значении года выпуска игры (*Date Released*) по производителю (*producer*). Другими словами, для каждого продюсера найти среднее значение года выхода компьютерных игр, которые с ним связаны).
   4.4. Год, когда было выпущено наибольшее количество компьютерных игр.
5. Завершить работу программы.
6. **[дополнительная задача]** Вывести на экран выборку игр, жанр которой - *First-person shooter*, а год выхода позже 2003, данные должны быть сгруппированы по годам.
   6.1. Сохраните выборку с сохранением порядка в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных об играх, в котором выделены группы по жанру (*Genre*), при этом в каждой группе следует упорядочить игры по убыванию *Date Released*. Перед упорядоченным набором должна выводиться строка с указанием модуля разницы (дельту) между максимальной и минимальной датой релиза (*Date Released*).
   7.1. Сохранять результат переупорядочения в файлы **Game_Sort.csv** с сохранением порядка.

Вариант 7. Неужели кто-то это слушает?

Первичные данные для работы программы находятся в файле **spotify_artist_data.csv**. Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **spotify_artist_data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **SpotifyArtist**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные об артистах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **spotify_artist_data.csv.** Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех артиста, достигших миллиарда прослушиваний хотя бы один раз (*One Billion* не нулевое).
3. Сохранять в файл **Artists.csv** выборку обо всех артистах, имеющих количество треков (*Tracks*) менее 100.
4. Выводить на экран сводную статистику по данным загруженного файла **spotify_artist_data.csv**:
   4.1. Общее количество артистов, исключая строки непригодные для анализа и заголовки файла.
   4.2. Полные данные о самом прослушиваемом артисте и наименее прослушиваемом артисте.
   4.3. Количество исполнителей, имеющих в названии (*Artist Name*) цифры.
   4.4. Общее количество исполнителей 100-миллионников с любым количеством треков.
   4.5. Количество исполнителей миллиардников, менее чем с 150 треками.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран список артистов, последнее изменение у которых произошли в один день (сгруппировать по дню изменения).
   6.1. Сохраните выборку с сохранением порядка в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню..
7. **[дополнительная задача]** Выводить на экран выборку артистов, у которых больше десяти треков и которых прослушивали в июле и августе. Выборку группировать по месяцам и сортировать имена исполнителей по алфавиту.
   7.1. Сохраните выборку с сохранением порядка в файл **Meed-artist.csv**

Вариант 8. Студент ради зачёта идет на всё, даже на занятия

Первичные данные для работы программы находятся в файле **exams.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **exams.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Student**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные об успеваемости. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **exams.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех студентах, имеющих свойство *test preparation course = completed*.
   2.1. Сохранять в файл **Test_Preparation.csv** выборку обо всех студентах, закончивших подготовительный курс (*test preparation course = completed*).
3. Выводить на экран данные о студентах, отобранных по стандартным ланчам (*lunch=standard*).
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество строк с данными о студентах (без учёта заголовков)
   4.2. Количество групп, к которым относятся студенты (*race*/*ethnicity*) и количество людей в каждой группе.
   4.3. Количество студентов, которые сдали экзамены (*math score*, *reading score*, *writing score*) больше, чем на 50 баллов. Статистику необходимо вывести по каждому экзамену.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран выборку студентов, которые являются женщинами (*female*), посчитать для каждой строки среднюю оценку за все экзамены (*math score*, *reading score, writing score*)
   6.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя.Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о студентах, в котором выделены группы по типу обеда (*standard* и *free*/*reduced*), при этом в каждой группе следует упорядочить студентов по возрастанию оценки по математике (*math score*). Перед упорядоченным набором должна выводиться строка с указанием модуля разницы (дельты) между максимальной и минимальной оценкой по математике.
   7.1. Сохранять результат переупорядочения в файле **Sorted_Students.csv** с сохранением порядка.

Вариант 9. Первая сессия

Первичные данные для работы программы находятся в файле **exams.csv** .
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **exams.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Examinee**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные об успеваемости. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **exams.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех студентах, имеющих свойство уровень образования родителей *(parental level of education)* = ***либо*** *high school,* ***либо*** *some college.*
3. Выводить на экран данные о студентах, отобранных по значению поля *lunch.* Значение отбора получить от пользователя с клавиатуры.
   3.1. Результат сохранить в файл **lunch-type.csv**. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество студентов по группам (*race* / *ethnicity*)
   4.2. Студент (студенты, если есть кто-то одинаковым минимальным баллом) с самым большим суммарным баллом за все экзамены *(math score, reading score, writing score*), аналогично с минимальным баллом.
   4.3. Количество мужчин (*gender* = *male*), которые входят в диапазон сдачи экзамена по математике (*math score*) от 0 до 100 с шагом 10.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о студентах, в котором выделены группы по типу *test preparation course*, при этом в каждой группе следует упорядочить студентов по группе (*race* / *ethnicity*) в лексикографическом порядке (например, group **A**, затем group **B**…).
   6.1. Сохранять результат переупорядочения в файле **Sorted_Students.csv** с сохранением порядка.
7. **[дополнительная задача]** Выводить на экран выборку студентов, у которых test preparation *course* = *completed* и баллы за каждый экзамен больше средней оценки по этому экзамену.
   7.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.

Вариант 10. Сколько стоит дата-сайнс?

Первичные данные для работы программы находятся в файле **Data_Science_Fields_Salary_Categorization.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **Data_Science_Fields_Salary_Categorization.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Worker**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о работниках сферы наук о данных. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **Data_Science_Fields_Salary_Categorization.csv.** Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию обо всех работниках больших (*Company_Size=L*) компаний.
   2.1. Сохранять эту выборку в файл **LC-Workers.csv**.
3. Выводить на экран сводную статистику по данным загруженного файла:
   3.1. Общее количество строк с данными о дата-саентистах, без учёта строки заголовков в исходном файле.
   3.2. Количество групп работников по полю *Designation* и количество работников в каждой группе.
   3.3. Количество работников, которые иногда работают дистанционно, то есть с ненулевым *Remote_Working_Ratio*.
   3.4. Количество средних компаний, зарплата которых на 25% больше средней по данным.
4. Выводить на экран выборку работников, зарплата которых находится в диапазоне от 0 до 50% от максимально возможной по выборке.
   4.1. Сохраните выборку в файл **Salary50-workers.csv**
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран выборку работников, место работы которых не совпадает с местом расположения компании.
   6.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран упорядоченный набор исходных данных о работниках, в котором выделены группы по полю *Experience*, при этом в каждой группе работники упорядочены по возрастанию зарплаты.
   7.1. Сохранять результат упорядочения в файле **Sorted-DS.csv** с сохранением порядка.

Вариант 11. Опытные инженеры данных

Первичные данные для работы программы находятся в файле **Data_Science_Fields_Salary_Categorization.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **Data_Science_Fields_Salary_Categorization.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Employee**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о работниках сферы наук о данных. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию о группах работников по Experience. Значение поля получать от пользователя с клавиатуры.
   2.1. Сохранять в файл **employees.csv** группируя по опыту работы. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
3. Выводить на экран сводную статистику по данным загруженного файла:
   3.1. Общее количество строк с данными без учёта строки с заголовками.
   3.2. Работников с наибольшей и наименьшей зарплатой.
   3.3. Количество *Data Engineer* работающих из Великобритании (*Employee_Location=GB*) .
   3.4. Количество работников, работающих в компаниях из Великобритании (*Company_Location=GB*), но работающих из иной страны, перед каждым из них написать из какой страны он работает.
4. Выводить на экран выборку работников, зарплата которых находится в диапазоне от 70 до 80% от максимально возможной по выборке.
   4.1. Сохраните выборку в файл **Salary7080-employees.csv**
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран список работников, группируя их по году работы, для каждой группы перед ней в отдельной строке указать диапазон зарплат, для границ диапазона выбрать меньшую и большую запрплаты по группе.
7. **[дополнительная задача]** Сохранять перечени работников по годам в файле **Employees-N.csv,** где *N* - номер года. Каждый перечень упорядочен по проценту дистанционной работы (Remote_Working_Ratio) от меньшего к большему.

Вариант 12. В Кейптаунском порту…

Первичные данные для работы программы находятся в файле **Port_Data.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **Port_Data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Port**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о портах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **Port_Data.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию о группе портов, являющихся *Anchorage* (устанавливается по полю *Type*).
   2.1. Сохранять в файл **Port-Anchorage.csv** список *Anchorage* портов, имеющих тип (по полю *Type*).
3. Выводить на экран из набора исходных данных информацию обо всех портах, имеющих свойство *Country* = *Spain*.
   3.1. Сохранять в файл **Port_Data-Spain.csv** выборку обо всех портах, имеющих свойство *Country* = *Spain*.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество строк с данными о портах (без учёта заголовков)
   4.2. Количество групп портов по типам (свойство *Area Local*) и количество портов в каждой группе.
   4.3. Количество портов *Anchorage* (свойство *Type*) в каждой из групп портов по свойству *Area Local*.
   4.4. Количество портов в жёлтом море *(Yellow Sea)* по типу *Local Area,* которые являются *Anchorage* по типу *Type*.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран выборку портов, не имеющих дополнительных особенностей (*UN Code* отсутствует / пуст). Перед выводом перечня портов должно выводиться среднее значение судов (*Vessels in Port*) портов этой выборки.
   6.1. Сохраните выборку в CSV-файл с именем, полученным от пользователя. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о портах, в котором выделены группы по полю основной тип (*Country*), при этом в каждой группе следует упорядочить порты по возрастанию индекса (#). Перед упорядоченным набором должна выводиться строка с указанием модуля разницы (дельта) между максимальным и минимальным индексом в группе портов *Country* = *China*.
   7.1. Сохранять результат переупорядочения в файле **Sorted-Port_Data.csv** с сохранением порядка, полученного при сортировке.

Вариант 13. Как провожают пароходы?

Первичные данные для работы программы находятся в файле **Port_Data.csv**.
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **Port_Data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Port**.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о покемонах. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **Port_Data.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию о группе портов, являющихся *Port* (устанавливается по полю *Type*).
   2.1. Сохранять в файл **Port-Port.csv** список портов, имеющих тип (по полю *Type*) *Port*.
3. Выводить на экран список портов, относящихся к одной и той же стране (*Country*). Перед каждой группой поколения выводить строку с данными о порте, у которого больше всего судов (*Vessels in Port в рамках страны*).
   3.1. Сохранять перечень портов в файлах **Port-Country-N.csv,** где *N* - название страны.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Общее количество портов по странам (*Country*).
   4.2. Полные данные о самом загруженном судами портом (*Vessels in Port*) и самом менее загруженном порте среди всех портов, загруженных из файла.
   4.3. Количество *Yellow Sea* (свойство *Area Local*) *Bohai Sea* (свойство *Area Local*).
   4.4. Количество портов страны *China*, у которых *Arrivals* меньше среднего значения по этому показателю в данных всего файла.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран переупорядоченный набор данных, где данные упорядочены в алфавитном порядке по полю *Port Name.*
   6.1. Сохранять результат переупорядочения в CSV-файле имя которого вводит с клавиатуры пользователь. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран выборку портов, *Departures* которых не более, чем на 10 единиц меньше максимального значения *Departures* среди всех портов.
   7.1. Сохраните выборку в файл **Departures-Port.csv**

Вариант 14. Почта Австралии

Первичные данные для работы программы находятся в файле **au_postcodes.csv.**
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **au_postcodes.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **PostOffice**. Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные о почтовых отделениях Австралии. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **au_postcodes.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию о почтовых отделениях, расположенных в Новом южном Уэллсе (*state_name = New South Wales*).
3. Сохранять в файл **NS-Wales-postcodes.csv** выборку о об отделениях, расположенных в конкретном штате (*state_name*), название штата вводит с клавиатуры пользователь. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Количество почтовых отделений в каждом штате с разным почтовым индексом.
   4.2. Процентное отношение почтовых отделений в конкретном месте (*place_name*) от общего числа почтовых отделений в стране и штате, где данное отделение расположено.
   4.3. Среднее количество почтовых отделений в города штата, имя штата запросить у пользователя.
   4.4. Статистику по количеству почтовых отделений в каждом штате.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о записях, в котором выделены группы по месту расположения почтового отделения по коду штата (*state_code*), при этом в каждой группе следует упорядочить по алфавиту названия места, где расположено почтовое отделение (*place_name*).
   6.1. Сохранять результат переупорядочения в файле **grouped-postcodes.csv** с сохранением порядка групп и записей внутри этих групп.
7. **[дополнительная задача]** Выводить на экран выборку записей по почтовым отделениям, расположенных на одной и той же широте (*latitude*) и долготе (*longitude*), но только если таких записей больше трёх.
   7.1. Сохраните только выборку в CSV-файл с именем, полученным от пользователя. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.

Вариант 15. Может кофейку?

Первичные данные для работы программы находятся в файле **reviews_data.csv.**
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **reviews_data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Review**. Данные о ссылке на изображения в объекты программы не загружать.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные об отзывах посетителей кофейни Starbucks. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **reviews_data.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию об отзывах с самым высоким рейтингом, размещённых в 2020 и 2021 годах (столбцы *Rating u Date*).
3. Сохранять в файл **top-reviews-20-21.csv** выборку об отзывах с рейтингом N (1 <=N <=5), введённым пользователем с клавиатуры. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Количество отзывов, полученных в каждом году.
   4.2. Процентное отношение отзывов с разным рейтингом по отношению к общему числу отзывов.
   4.3. Соотношение отзывов с рейтингом, равным 1, у которых есть приложенное изображение к таким же отзывам, но с отсутствующим изображением
   4.4. Количество отзывов с рейтингом 1-2 и 4-5 по разным годам.
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран выборку записей по отзывам, полученным в одном и том же месте (*location*), но без подтверждающих изображений (*Image*).
   6.1. Сохраните только выборку в CSV-файл с именем, полученным от пользователя. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
7. **[дополнительная задача]** Выводить на экран переупорядоченный набор исходных данных о записях, в котором выделены группы по рейтингу (*Rating*), при этом все записи внутри групп упорядочены по убыванию дат (*Date*), т.е. самые свежие отзывы идут первыми.
   7.1. Сохранять результат переупорядочения в файле **grouped-rates.csv** с сохранением порядка групп и записей внутри этих групп.

Вариант 16. Лучше чай…

Первичные данные для работы программы находятся в файле **reviews_data.csv.**
Разработайте нестатические классы для чтения и записи CSV-файлов со структурой как у файла **reviews_data.csv.** Для представления данных из файла в программе разработайте минимум один нестатический класс **Review**. Данные о ссылке на изображения в объекты программы не загружать.
Используйте разработанные классы и объекты для реализации следующий операций текстового меню:

1. Вводить адрес файла, из которого загружаются данные об отзывах посетителей кофейни Starbucks. Обратите внимание, что название файла может быть произвольным, структура файла должна совпадать со структурой первичного файла **reviews_data.csv**. Этот пункт меню показывается до загрузки данных, если данные уже загружены в программу, пункт следует заменить на пункт, позволяющий сменить набор данных.
2. Выводить на экран из набора исходных данных информацию об отзывах с рейтингом **N**, где **0 < N <= 5** вводится с клавиатуры пользователем. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.
3. Сохранять в файл **rated-reviews.csv** выборку об отзывах с самым высоким и самым низким рейтингами. Сначала в файле идут записи с самым низким рейтингом, затем с самым высоким.
4. Выводить на экран сводную статистику по данным загруженного файла:
   4.1. Количество отзывов с разным рейтингом (*Rating*), полученных в разные годы.
   4.2. Процентное отношение отзывов с самым низким рейтингом (*Rating = 1*), по отношению к общему числу отзывов в наборе данных.
   4.3. Распределение по годам количества записей, в которых отсутствует текст отзыва (*Review*).
   4.4. Распределение по годам количества записей, в которых (1) отсутствует оценка (*Rating*) и текст отзыва (*Review*); (2) отсутствует оценка (*Rating*) и приложенное изображение (*Image*).
5. Завершить работу программы.
6. **[дополнительная задача]** Выводить на экран из набора исходных данных информацию об отзывах, сгруппированные по отношению к одному и тому же штату (размещено после запятой в значении поля *Location*) и упорядоченные по возрастанию рейтинга внутри каждого штата.
   6.1. Сохранять результат переупорядочения в файле **grouped-rates.csv** с сохранением порядка групп и записей внутри этих групп.
7. **[дополнительная задача]** Выводить на экран выборку записей по отзывам, полученным в один и тот же день, без учёта локации.
   7.1. Сохранять только выборку в CSV-файл с именем, полученным от пользователя. Если введённые данные некорретны, то выходной файл не создаётся, на экран пользователю выводится сообщение о некорректности данных и экранное меню.