**Раздел 1. Практическая значимость разрабатываемого программного средства**

Разрабатываемое программное средство (ПС) — CSV-Viewer — предназначено для удобного преобразования данных из CSV-файлов в HTML-таблицу, обеспечивая визуально понятное представление информации.

CSV-файлы широко используются для хранения и обмена табличными данными, однако их чтение в исходном текстовом виде затруднительно. Разработка данного ПС позволяет автоматически формировать структурированное представление данных в виде HTML-таблицы, обеспечивая удобное форматирование, проверку ошибок и выбор визуального оформления.

Основными пользователями могут быть разработчики, аналитики, офисные работники и все, кто работает с CSV-данными и нуждается в их удобном отображении. Программа будет полезна как для личного использования, так и для применения в организациях, где необходимо преобразовывать CSV-данные в читаемый формат.

Данное ПС может применяться в различных сценариях:

* автоматизация отчетов и анализа данных: например, в компании регулярно формируются CSV-отчеты о продажах. Без CSV-Viewer сотрудникам приходится вручную загружать файлы в Excel, настраивать стили и преобразовывать данные в HTML для отчётности. С помощью разработанного ПС можно быстро создать таблицу в нужном формате и визуальном стиле без лишних действий;
* использование в качестве программного модуля на веб-ресурсах: CSV-Viewer может быть использован на сайтах, сервисах и блогах для автоматического представления табличных данных. Например, финансовые ресурсы могут динамически отображать биржевые котировки, интернет-магазины — обновляемые списки товаров, а блоги — сравнительные таблицы характеристик товаров или услуг;
* публикация результатов опросов: допустим, организация проводит опрос среди сотрудников или клиентов, собирая данные в CSV-файл. С помощью программы можно легко преобразовать эти данные в наглядную HTML-таблицу и разместить на сайте или внутреннем портале, чтобы результаты были удобны для просмотра.

Если бы не было данного программного средства, то пользователям пришлось бы либо вручную редактировать CSV-файлы в текстовых редакторах, что неудобно и может привести к ошибкам, либо использовать сложные табличные процессоры, которые не всегда подходят для простого и быстрого преобразования данных в HTML-формат. Также без данного ПС отсутствовала бы возможность автоматизированной проверки корректности CSV-файлов перед их конвертацией.

**Раздел 2. Функциональные требования к ПС**

Программное средство CSV-Viewer должно обеспечивать выполнение следующих функций:

1. Проверка исходного файла: пользователь вводит путь к CSV-файлу через консоль; программа проверяет корректность пути и формата файла, анализирует структуру данных. Реакция программы на возможные ошибки:

* путь не был введён: выводит сообщение об ошибке — «Путь к CSV-файлу не может быть пустым.»; (тут что-то с форматированием текста)
* некорректный путь к CSV-файлу: выводит сообщение об ошибке — «CSV-файл не найден. Пожалуйста, укажите корректный путь к CSV-файлу.»;
* CSV-файл пустой: выводит сообщение об ошибке — «CSV-файл не содержит данных.»;
* пустые значения в строке заголовка: выводит сообщение об ошибке — «Строка заголовка содержит пустые значения.»;
* пустая строка в CSV-файле или пустые значения в строке данных: программа информирует об этом пользователя, указав номер некорректной строки; программа пропускает некорректную строку, продолжая свою работу;
* неверное количество столбцов в строке данных: информирует пользователя о номере некорректной строки и количествах найденных и ожидаемых столбцов;
* отсутствие данных в CSV-файле: выводит сообщение об ошибке — «В CSV-файле не найдено достоверных данных.». Противоречит требованию, когда файл пуст. Надо остановиться на чем-то одном

В качестве разделителя данных выступает запятая («,»). – это плохо, запятая может являться корректным значением в каком-то столбце. В CSV, как правило, используют в качестве разделителя точку с запятой

Пример корректного исходного файла:

Id,Имя,Возраст

1,Артём,25

2,Валерия,30

Пример некорректного исходного файла:

Id,Имя,,

1;Артём

2,Валерия,30,438

1. Генерация HTML-таблицы: преобразование содержимого CSV-файла в HTML-таблицу; корректное отображение данных, включая специальные символы; возможность выбора темы оформления.

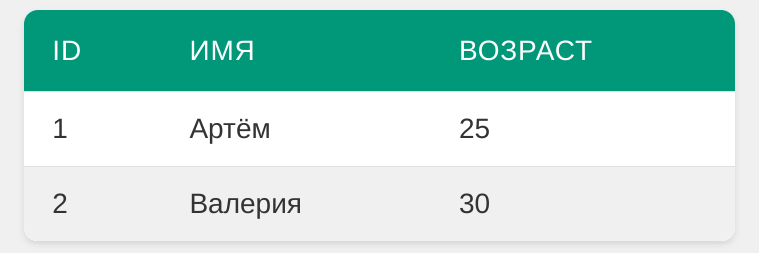
Пример исходного файла:

Id,Имя,Возраст

1,Артём,25

2,Валерия,30

Пример сгенерированной HTML-таблицы:



1. Выбор темы оформления и шрифта: пользователь через консоль выбирает одни из доступных тем и шрифтов; программа применяет выбранное оформление к итоговому HTML-файлу.

Доступные темы оформления HTML-таблицы: светлая (light), темная (dark), зеленая (green).

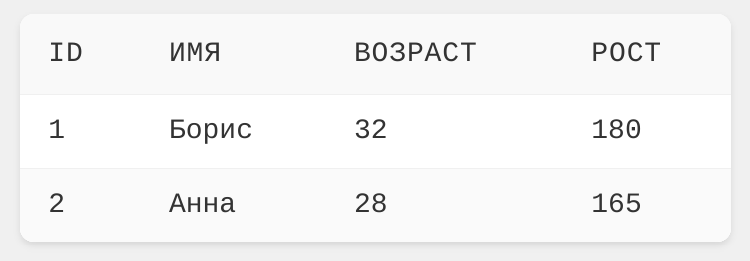
Доступные варианты шрифтов: Arial, Courier New, Times New Roman.

Способ указания темы и шрифта: программа просит пользователя ввести одну из тем и шрифтов, предоставляя доступные варианты и их номера; в квадратных скобках программа указывает номер варианта по умолчанию, который будет использоваться, если пользователь не станет вводить значение (нажав клавишу «Enter», можно пропустить ввод значения); пользователь по желанию вводит номер темы и шрифта через консоль.

При вводе значения, отличного от корректного номера (1, 2 или 3) темы или шрифта, программа выводит сообщение об ошибке о том, что данной темы или шрифта не существует, ~~завершая свою работу~~.

Примеры различных стилей оформления HTML-таблицы:

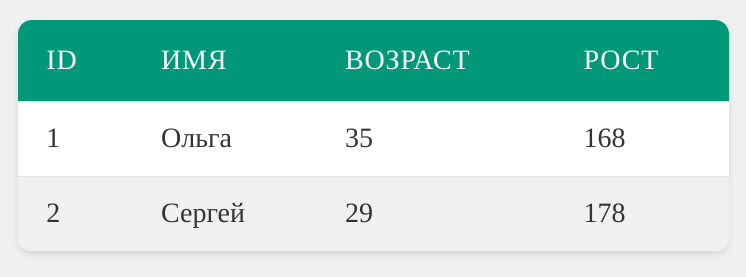
1. Тема - светлая (light), шрифт - Courier New:



1. Тема - темная (dark), шрифт - Arial:



1. Тема - зеленая (green), шрифт - Times New Roman:



1. Сохранение и просмотр результата: итоговый HTML-файл сохраняется в указанной пользователем директории; возможность посмотреть сгенерированный файл, открыв его в браузере.

Пользователь указывает абсолютный или относительный (относительно папки, в которой запускается программа) путь до директории, в которую желает сохранить итоговый файл (например, C:\Users\Alex\Desktop\).

Программа проверяет введенный путь к директории:

* если директория существует, программа сохраняет в неё итоговый файл, выводя сообщение от успехе, указав полный путь до сгенерированного файла;
* в случае отсутствия введенной директории программа выводит сообщение об ошибке о том, что указанной директории не существует, ~~завершая свою работу~~;
* если отсутствуют права на запись в указанную директорию, программа выводит сообщение об ошибке, указав полный путь до файла, который не удалось записать, ~~завершая свою работу~~.

**Раздел 3. Модульная структура проектируемого ПС**

**Раздел должен начинаться диаграммой классов!!!**

Проектируемое программное средство (ПС) состоит из нескольких модулей, представленных классами, каждый из которых выполняет строго определённую функцию. Такая модульная структура повышает читаемость кода, облегчает его поддержку и расширение.

3.1. Перечень классов и их назначение

|  |  |
| --- | --- |
| Класс | Назначение |
| Program | Точка входа в программу. |
| UserInterface | Управляет взаимодействием с пользователем, запрашивает входные данные, вызывает обработку CSV-файла и сохранение HTML. |
| CsvParser | Отвечает за чтение и разбор CSV-файла, проверку структуры и обработку ошибок. |
| CsvToHtml | Генерирует HTML-таблицу из обработанных данных. |
| ThemeManager | Обрабатывает выбор темы и шрифта, предоставляет стили оформления. |

3.2. Описание классов, их данных и методов

**В описании методов не приведены типы возвращаемых значений и аргументов. Где указаны, нет описания того, какую информацию они представляют**

**Не указаны в описании методов типы данных С#**

1. Program (главный класс, точка входа)

Класс содержит один метод:

* Main() – создает объект UserInterface и запускает процесс выполнения программы.

1. UserInterface (работа с пользователем)

Класс отвечает за ввод и вывод информации, а также за управление основными этапами работы программы.

Данные: нет постоянных данных, вся информация запрашивается и передаётся по мере выполнения.

Методы:

* Start() – основной метод, запускающий выполнение программы;
* GetFilePath() – запрашивает у пользователя путь к CSV-файлу и проверяет его существование;
* GetTheme() – запрашивает у пользователя выбор темы оформления HTML;
* GetFont() – запрашивает у пользователя выбор шрифта;
* GetOutputPath(filePath, theme, font) – запрашивает у пользователя путь для сохранения HTML-файла, формирует уникальное название файла;
* SaveHtmlFile(outputPath, html) – сохраняет сгенерированную HTML-таблицу в файл.

Этот класс необходим для организации взаимодействия с пользователем. Без него программа не смогла бы принимать вводимые данные и передавать их на обработку.

1. CsvParser (чтение и обработка CSV-файла)

Класс отвечает за загрузку данных из CSV и их валидацию.

Данные: нет постоянных данных, работа идёт с временными списками строк.

Методы:

* Parse(filePath) – читает и разбирает CSV-файл, возвращает данные в виде списка списков строк;
* ParseCsvLine(line) – разбирает одну строку CSV, учитывая кавычки.

Класс необходим для структурированной обработки CSV-файлов. Он содержит всю логику валидации, предотвращая появление ошибок на этапе генерации HTML.

Выбор типов данных:

List<List<string>> – используется для представления данных, где каждая вложенная коллекция представляет строку CSV. Это позволяет легко перебирать данные при генерации HTML.

1. CsvToHtml (генерация HTML-таблицы)

Этот класс преобразует полученные данные в HTML-код.

Данные:

* data\_ (List<List<string>>) – хранит разобранные данные CSV-файла.

Методы:

* GenerateHtml(theme, font) – создаёт HTML-таблицу с указанными стилями.

Этот класс изолирует логику генерации HTML от других частей программы.

1. ThemeManager (настройки оформления)

Класс отвечает за выбор темы и шрифта.

Методы:

* GetTableStyle(theme) – возвращает CSS-стиль для указанной темы.
* GetThemeName(choice) – преобразует ввод пользователя в название темы.
* GetFontName(choice) – преобразует ввод пользователя в название шрифта.

Этот класс позволяет легко расширять и модифицировать систему оформления HTML-файла без изменения других частей программы.

3.3. Взаимодействие классов

Последовательность работы программы:

1. UserInterface запрашивает у пользователя путь к CSV-файлу;
2. CsvParser читает файл, проверяет структуру и возвращает данные;
3. UserInterface запрашивает тему и шрифт у пользователя.
4. CsvToHtml создаёт HTML-таблицу;
5. UserInterface сохраняет HTML-файл в указанную папку.

3.4. Порядок использования классов и методов

1. UserInterface.Start():

* вызывает GetFilePath(), получает путь к CSV-файлу;
* передаёт путь в CsvParser.Parse(), получает List<List<string>>;
* запрашивает у пользователя тему (GetTheme()) и шрифт (GetFont());
* запрашивает путь для сохранения (GetOutputPath()).

1. CsvToHtml.GenerateHtml(theme, font): преобразует данные в HTML-таблицу.
2. UserInterface.SaveHtmlFile(outputPath, html): записывает HTML-файл в указанное пользователем место.

Программа структурирована по принципу разделения ответственности. Каждый класс выполняет конкретную задачу, что делает код удобным для поддержки и расширения.