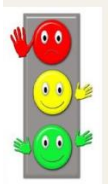
Школа:		
Дата:	**ФИО учителя:**	
Класс:	**Участвовали:**	**Не участволвали:**
Тема урока: Работа с файлами. Работа с системным окружением		
Цели обучения, которые достигаются на данном уроке	В этом уроке познакомятся работой с файлами языка Ruby	
Цели урока	**Все учащиеся смогут**: • Узнать работу с файлами *Ruby* **Большинство учащихся смогут**: • *научатся находить среду Ruby* **Некоторые учащиеся смогут:** *Написать программу*	
Критерии оценивания	**Грамотность -1балл** **Оригинальность -1балл** **Время -1 балл**	
Воспитаниеценностей	Воспитание уважения к мнению друг друга, ответственность, коммуникативные способности, критическое мышление	
Предварительныезнания	История Ruby, «Интерпретация» Компиляция и интерпретация, для программистов, Что умеет Ruby? Что не умеет Ruby?	
Межпредметныесвязи	Информатика, математика	

Запланированные этапы урока	**Запланированная деятельность на уроке**	**Ресурсы**
Начало урока _5__ мин	**Организационный момент.** Приветствие учащихся. Создание положительного эмоционального фона. Настрой на рабочий лад. Показываю тему урока и цель урока, озвучиваю критерии для цели урока **Деление на подгруппы.** Стратегия «Нумерация»	Карточки с цифры
Середина урока _30__ мин	**Организационный момент.** *Определите смысл этих слов, что они из себя представляют:* *CSS / HTML* *PHP* *SQL* *JavaScript* *Python* *Go* *Java* *Ruby* *C++* *C* • Детям раздаются карточки с вышеуказанными словами, дети в группе определяют что это и где они используются, они могут воспользоватся мобильными средствами или классным	Карточки для стратегии светофор 

компьютеров на его исполнение некоторые ограничения…

Файлы в программах играют роль хранилищ, в которые можно записать любые объекты. В отличие от привычных нам объектов, файлы позволяют хранить данные даже тогда, когда программа завершила свою работу. Именно поэтому они могут использоваться для передачи данных между разными программами или разными запусками одной и той же программы.

Как организована работа с файлами? В самом общем случае работа с файлами состоит из следующих этапов:

1. Открытие файла. Сущность этого этапа состоит в создании объекта класса `File`.
2. Запись или чтение. Вызываются привычные нам методы вывода на экран и не совсем привычные — ввода-вывода.
3. Закрытие файла. Во время закрытия файла происходят действия с файловой системой. С объектом, который создаётся при открытии файла, ничего не происходит, но после этого он указывает на закрытый файл, и производить операции чтения/записи при помощи него уже нельзя.

Существует масса способов реализации работы с файлами:

1. Чтение при помощи класса `IO`. Класс `IO` имеет множество методов, которые позволяют производить чтение из текстовых файлов (с «двоичными файлами» лучше так не работать). Если нужно считать весь текстовый файл, то лучше пользоваться методами класса `IO`.
2. Перенаправление потоков. Существует три предопределённые переменные: `$stdout`, `$stdin` и `$stderr`. Если им присвоить объект класса `File` (создаваемый во время открытия файла), то весь вывод пойдёт в файл, который присвоили переменной `$stdout`. Весь ввод будет браться из файла, который присвоили переменной `$stdin`, а все ошибки будут сохраняться в файле, который присвоили переменной `$stderr`. Если нужно работать только с одним файлом на чтение и одним файлом на запись, то обычно используют этот способ. Также очень удобно использовать перенаправление потока ошибок (переменная `$stderr`) для программ, которые работают в составе пакетных файлов и используют только интерфейс командной строки.
3. Универсальный способ. Используется в ситуациях, когда нельзя использовать предыдущие два способа.

Подведём небольшой итог:

1. Если нужно считать весь файл целиком, то надо использовать методы класса `IO`.
2. Если нужно работать только с одним файлом на чтение и только одним файлом на запись, то надо использовать перенаправление потока.
3. Если нельзя применить два вышеперечисленных способа, то надо использовать универсальный способ работы с файлами.

Для чтения файла целиком используется метод `.read`. Он считывает

	весь файл в виде строки. Во время его использования не сто́ит задумываться об открытии/закрытии файла, так как эти операции скрыты внутри метода.
	
	`config = IO.read('config.yaml')`
	`config.class  #=> String`
	
	Имя файла — это строка.
	
	В примере можно увидеть, как считывается файл `config.yaml` в переменную `config`. Вторая строка показывает, что при использовании метода `.read` файл считывается в виде строки (класс `String`). Теперь к переменной `config` можно применять любые методы из богатого строкового арсенала.
	
	В качестве упражнения предлагаю вам считывать и выводить на экран любые файлы, которые вы только найдёте у себя на диске.	

Конец урока _5__ мин	**Рефлексия:** Метод «Рефлексивные карточки» Было интересно… Я понял, что… У меня получилось… Расскажу дома, что…	«Рефлексивные стикеры» 

Дифференциация – каким образом Вы планируете оказать больше поддержки? Какие задачи Вы планируете поставить перед более способными учащимися?	**Оценивание – как Вы планируете проверить уровень усвоения материала учащимися?**	**Охрана здоровья и соблюдение техники безопасности**