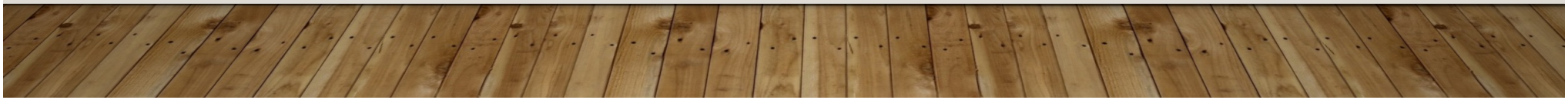
ЛЕКЦИЯ 16

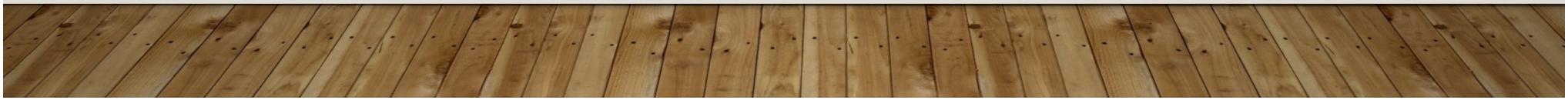
ЗАПИСЬ ФАЙЛОВ В RUBY



ПЛАН ЗАНЯТИЯ

1. Как работать со временем в Ruby
2. Запись в файлы, пишем дневник

- Мы научимся писать данные в файлы, узнаем как работать со временем в Ruby, напишем программу-дневник.



РАБОТАЕМ СО ВРЕМЕНЕМ – КЛАСС

- В Ruby есть класс для удобной работы со временем. Он называется Time. Например, чтобы понять, который сейчас час гики-программисты не смотрят на часы, а просто быстренько пишут программу на Ruby, в которой пишут одну строчку:

```
puts Time.now
```

- Дело в том, что метод now (англ. «сейчас») класса Time возвращает текущий момент времени:

```
2014-11-24 17:52:13 +0300
```

РАБОТАЕМ СО ВРЕМЕНЕМ – КЛАСС

- Любой экземпляр класса Time — объект «момента времени». Именно такой объект возвращает метод Time.now. У экземпляра класса Time есть много полезных методов. Но самый важный из них — метод strftime, который возвращает время в виде строки по специальному шаблону, например:

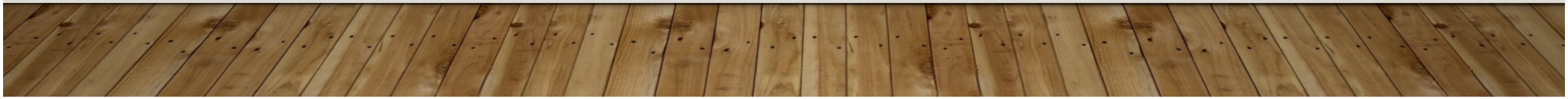
```
time = Time.now
puts time.strftime("%H:%M")
```

- Выведет на экран текущее время в 24-часовом формате с точностью до минут:

```
17:55
```

РАБОТАЕМ СО ВРЕМЕНЕМ – КЛАСС

- Чтобы попросить метод strftime вывести время именно в таком формате, мы передаём ему в качестве параметра так называемый «формат времени», строчку "%H:%M".
- Вот эти вот конструкции %H и %M (специальные ключи) метод заменяет на соответствующие данные из объекта класса Time, у которого он был вызван.
  - %H — часы в 24-часовом формате
  - %M — минуты с нулём, если меньше 10
- А остальные символы (всё, что не начинается с %, в нашем случае двоеточие), остаются как и были.



РАБОТАЕМ СО ВРЕМЕНЕМ – КЛАСС

- Чтобы вывести, например, текущую дату, нужно написать:

```
puts time.strftime("%d.%m.%Y")
```

- Это выведет что-то типа:

```
16.12.2021
```

РАБОТАЕМ СО ВРЕМЕНЕМ – КЛАСС

- Это, кстати, не только в Ruby. Такой способ форматирования времени с помощью специальных шаблонов-строк — некий общепринятый в разных языках стандарт. Отличие может быть только в деталях, в названиях некоторых ключей. О том, какие ещё бывают ключи в Ruby можно посмотреть по ссылке.

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

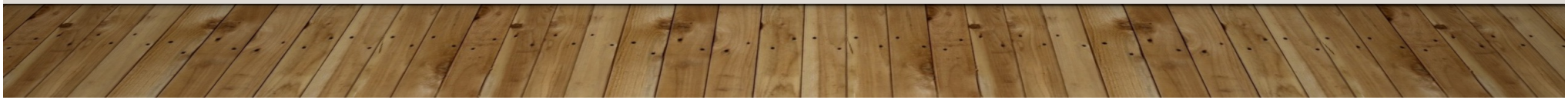
- Для того, чтобы записать что-то в файл, нам этот файл нужно сперва открыть, практически точно также, как мы это делали в 13-м уроке. С той лишь разницей, что мы используем другой ключ: вместо "r:UTF-8" мы напишем "a:UTF-8", потому что файл нам надо открыть для «добавления» (англ. append) строк.
- Если такой файл есть, то всё, что мы запишем в файл, будет дописано в конец файла, если же такого файла ещё нет, то будет создан новый файл и всё, что мы запишем в него будет сохранено в этом новом файле.

```
file = File.new("./file.txt", "a:UTF-8")
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Теперь давайте добавим что-нибудь в файл, это делается с помощью метода print у экземпляра класса File:

```
file.print("Строка для записи в файл\n\r")
```



ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Обратите внимание на символы \n\r\ в конце — это символы переноса на следующую строку. Если мы что-то захотим писать в этот файл снова, всё, что мы будем дописывать, начнётся с новой строчки. Так просто красивее и удобнее.
- Всегда заканчивайте ваши строки символом переноса строки \n (\r добавляется для пользователей Windows). Ну и как обычно, закроем файл:

```
file.close
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Всё, теперь в новом (если у вас ещё не было файла file.txt) файле записана строка:

```
Строка для записи в файл
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Если мы повторно сделаем открытие файла с ключом "a:UTF-8" и проделаем всё заново, но запишем уже другую строку:

```
file = File.new("./file.txt", "a:UTF-8")
file.print("Ещё одна строка\n\r")
file.close
```

- То в файле будет записано уже две строки:

```
Строка для записи в файл
Ещё одна строка
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Итак, когда мы умеем создавать в наших программах новые файлы и дописывать текст в уже существующие, мы можем начать писать нашу программу-дневник. Пишем программу «Дневник». Для начала, как обычно, ставим задачу:
- **Написать программу, которая предлагает пользователю сделать дневниковую запись в консоли, ждёт, пока пользователь напишет текст, а потом сохраняет его, добавив текущее время, в файл с именем в виде текущей даты. Записи в разные дни кладутся в разные файлы, все записи одного и того же дня лежат в одном файле.**

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Как обычно, создаём отдельную папку для нашей программы: c:\%username\lesson16 и в ней создаём файл my_diary.rb (не забудьте сохранить файл в кодировке UTF-8).
- Начнём с того, что выведем на экран инструкцию для пользователя с помощью команды puts:

```
puts "Привет, я твой дневник. Скажи мне что у тебя на уме и в душе?"
puts "Я сохраню всё, что ты напишешь до строчки \"end\" в файл."

current_path = File.dirname(__FILE__)
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Обратите внимание — мы снова сохранили путь к программе в переменную current_path. Теперь давайте спросим у пользователя, что он хочет написать. Как обычно, делаем это с помощью команды STDIN.gets:

```
line = nil
all_lines = []

while line != "end" do
     line = STDIN.gets.encode("UTF-8").chomp
     all_lines << line
end

all_lines.pop
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Мы будем записывать всё, что введёт пользователь в массив all_lines, пока пользователь не введёт end (по-английски маленькими буквами). После выполнения этого блока у нас в массиве all_lines вся нужная нам информация от пользователя (строчка all_lines.pop убирает из массива последний элемент, нам ведь не нужна там строчка с end), осталось только записать её в файл. Новый файл, если это первая запись за сегодня и в уже существующий, если пользователь делает уже не первую запись за день.

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Для этого мы заведём переменную time и запишем в неё текущее время:

```
time = Time.now
```

- И с помощью метода strftime в переменную file_name мы запишем строку в формате «2021-12-16» (для удобства сортировки год выводим перед месяцем, а месяц перед днём, соединяем всё дефисами для наглядности):

```
file_name = time.strftime("%Y-%m-%d")
```

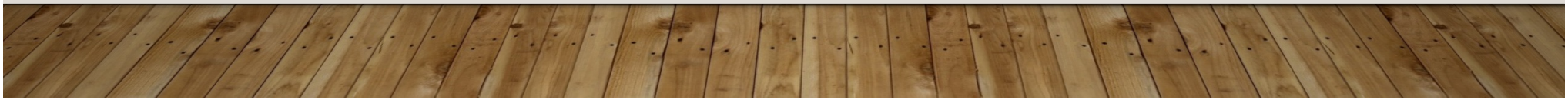
ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- В переменную time_string запишем текущее время, как мы это делали в начале нашего урока:

```
time_string = time.strftime("%H:%M")
```

- Сделаем также строчку с разделителем, которую мы будем писать каждый раз в конце записи, чтобы отделять записи друг от друга внутри одного файла:

```
separator = "------------------------------"
```



ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Почти все готово. Осталось только записать в наш файл все строчки этого массива. Мы будем это делать в цикле for. И на этот раз будем пользоваться не методом print, а методом puts, который также есть у каждого экземпляра класса File. Единственное отличие метода puts от метода print в том, что первый после записи в файл строчки, которую ему передали в параметрах, добавляет ещё перенос строки:

```
file = File.new("./file.txt", "a:UTF-8")
file.puts("Строка с переносом")
file.close
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- В файле file.txt будет две строки — первая с текстом «Строка с переносом» и вторая пустая. Итак, пора записать текст дневниковой записи в наш файл:

```
file = File.new(current_path + "/" + file_name + ".txt", "a:UTF-8")
file.print("\n\r" + time_string + "\n\r")

for item in all_lines do
  file.puts(item)
end

file.puts(separator)
file.close
```

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Обратите внимание, что путь к файлу мы как обычно собрали из нескольких частей:
  - переменной с текущей папкой программы current_path
  - слеша /, чтобы показать, что ищем и создаём файлы в этой папке
  - имени файла, которое храниться в переменной file_name
  - расширения файла .txt

ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Заканчиваем мы работу программы выводом пользователю сообщения о том, что всё записано:

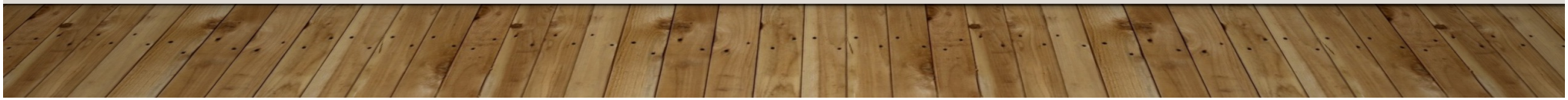
```
puts "Ваша запись сохранена в файле #{file_name}.txt"
puts "Запись сделана в #{time_string}"
```

- Пора запускать программу:

```
cd c:\%username\lesson16
ruby my_diary.rb
```

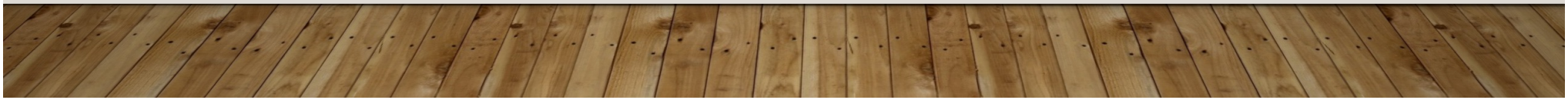
ДОБАВЛЕНИЕ ДАННЫХ В ФАЙЛЫ

- Чтобы найти ваши записи, вам нужно в проводнике перейти в эту самую папку lesson16.
- Как только наиграетесь с программой, можете проверить, что она создаёт файлы для каждого нового дня, делает в них отметку о текущем времени и сохраняет то, что вы написали.



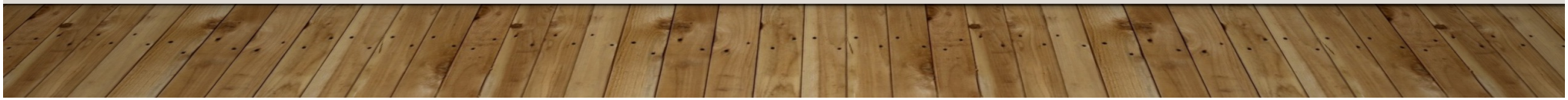
HELLO, FILE!

- Напишите программу, которая здоровается в файл hello.txt (пишет строку "Hello, file!" в него).
- Вместо метода puts откройте файл с помощью класса File и запишите строчку в файл с помощью метода file.puts. Не забудьте закрыть файл.



РОЖИ В ФАЙЛ!

- Сделайте генератор рожиц таким, чтобы он рисовал рожицы в файл. Каждый раз в новый, название файла должно состоять из строчки "face", даты и текущего времени.



РОЖИ В ФАЙЛ! ПОДСКАЗКА

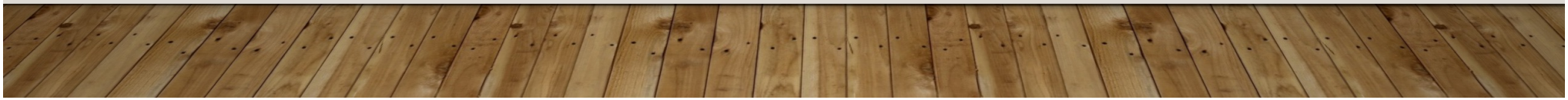
- Перед вызовом 4-х методов puts запишите в переменную название для файла. Оно будет состоять из трёх частей: постоянной части "face-", меняющейся с каждой секундой части с текущим временем, и ещё одной постоянной части, расширения файла ".txt":

```
time = Time.now
filename = "face " + time.strftime("%Y.%m.%d %H-%M-%S") + ".txt"
```

- Далее откройте файл с именем filename для записи и запишите в него 4 фрагмента лица, повторяя шаги записи, которые мы делали ранее. Не забудьте, что файл откроется там, откуда вы будете вызывать программу, в этом случае это нормально.

СЕГОДНЯ ВЫХОДНОЙ? (С ПРАЗДНИКАМИ)

- Улучшите программу из ранних уроков, которая говорит, выходной ли сегодня.
- Сделайте так, чтобы программа говорила, что сегодня выходной не только если сегодня суббота или воскресенье, но и если сегодня один из государственных праздников.
- Список праздничных дней на ближайший год нагуглите в интернете и запишите в файл, который будет использовать ваша программа.



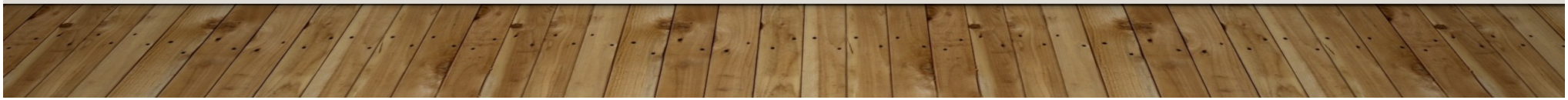
СЕГОДНЯ ВЫХОДНОЙ? (С ПРАЗДНИКАМИ). ПОДСКАЗКА

- Сохраните список праздничных дней в файл data/holidays.txt в формате День.Месяц, получится что-то вроде:

```
01.01
02.01
...
23.02
...
```

СЕГОДНЯ ВЫХОДНОЙ? (С ПРАЗДНИКАМИ). ПОДСКАЗКА

- В программе откройте этот файл и сохраните все дни в массив holidays с помощью метода readlines. Не забудьте в цикле обрезать у каждого элемента последний символ с помощью метода chomp! (с восклицательным знаком, потому что нам нужно менять элементы массива).
- Если сегодня не суббота и не воскресенье, попробуйте найти в этом массиве элемент, соответствующий текущему дню с помощью метода include? или просто пройдитесь по нему циклом, проверяя, совпадёт ли текущая дата с датой в массиве. Если совпал — можно отдыхать.



СПРАВОЧНАЯ ИНФОРМАЦИЯ

- Еще раз о файлах в Руби
- Когда читаешь хорошо написанную документацию :)
- Обзор возможностей Ruby

СПАСИБО ЗА ВНИМАНИЕ!

Запись файлов в Ruby