CRIMEA DIGITAL GROUP ACADEMY

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

ЛЗ #3 - Работа с файлами

Теория

Задание

Теория

Задание

Как читать файлы

Файлы в руби читают в такой последовательности:

1. **Открывают файл**, с помощью метода `open`.
2. **Читают файл,** целиком, построчно, или лишь небольшую часть байт.
3. **Закрывают файл,** используя метод `close`.

Как читать файлы

Используя класс File и путь к файлу, можно получить объект класса File, но не сам контент файла

```ruby
file = File.open("students.txt")
```

Дальше с этим файлом уже можно работать.

Как читать файлы

Чтение файла целиком происходит с помощью метода `read`

```ruby
file = File.open("students.txt")
file_data = file.read
# "Ivan Ivanov\nPetr Petrov\n"
```

Как читать файлы

Если в файле много строк, можно предыдущий результат разбить с помощью метода split и получить массив строк

```ruby
file = File.open("students.txt")
file_data = file.read
# "Ivan Ivanov\nPetr Petrov\n"
file_data.split("\n")
# ["Ivan Ivanov", "Petr Petrov"]
```

Как читать файлы

Или прочитать файл **построчно** убрав `\n` из конца строки с помощью метода `chomp`

```ruby
file = File.open("students.txt")
file_data = file.readlines.map { |line| line.chomp }
# или с помощью эквивалентной записи
file_data = file.readlines.map(&:chomp)
# ["Ivan Ivanov", "Petr Petrov"]
```

Как читать файлы

Когда вы закончили работу с файлом, его необходимо закрыть, чтобы освободить память и системные ресурсы. Для этого используйте метод close

```ruby
file = File.open("students.txt")

file_data = file.readlines.map(&:chomp)
# ["Ivan Ivanov", "Petr Petrov"]

file.close
```

Как читать файлы

Альтернативным способом открытию/закрытию файла можно использовать метод класса File read чтобы получить контент сразу

```
file_data = File.read("students.txt").split("\n")
# ["Ivan Ivanov", "Petr Petrov"]
```

Как читать файлы

Иногда приходится работать с очень большими файлами, которые могут не поместиться в RAM, в таких случаях обычно используют метод

```
students = []
File.foreach("a.rb") { |line| students.push(line.chomp) }

students
# ["Ivan Ivanov", "Petr Petrov"]
```

Как писать в файл

Иногда приходится работать с очень большими файлами, которые могут не поместиться в RAM, в таких случаях обычно используют метод

```
students = []
File.foreach("a.rb") { |line| students.push(line.chomp) }

students
# ["Ivan Ivanov", "Petr Petrov"]
```

Как писать в файл

Для записи в файл в Ruby нужно

1. **Открывают файл**, с помощью метода `open` в режиме записи с помощью флага `w`.
2. Используют метод `write`**, для добавления данных в файл**
3. **Закрывают файл,** используя метод `close`**.**

Как писать в файл

```ruby
File.open("log.txt", "w") { |f| f.write "#{Time.now} - User logged in\n" }
```

Важно:

При этом будет перезаписано содержимое предыдущего файла!

Если вы хотите дополнить файл, используйте флаг "a" (append) вместо "w" (write).

Как писать в файл

Шорткат для записи данных в файл

```ruby
File.write("students.txt", "data...")
# it will use "w" mode by default
```

Или для дополнения файла

```ruby
File.write("students.txt", "data...", mode: "a")
```

Работа с файлами и папками

```ruby
# Переименование файла
File.rename("old-name.txt", "new-name.txt")
# Размер файла в байтах
File.size("users.txt")
# Проверка на существование файла
File.exists?("log.txt")
# Способ получения расширения файла
File.extname("users.txt")
# => ".txt"
```

Работа с файлами и папками

```ruby
# Получить имя файла, без пути к нему
File.basename("/tmp/ebook.pdf")
# => "ebook.pdf"
# Метод для получения пути к файлу
File.dirname("/tmp/ebook.pdf")
# => "/tmp"
# Проверка, является ли файл по указанному маршруту директорией
File.directory?("cats")
```

Работа с файлами и папками

```ruby
# Все файлы в текущей директории
Dir.glob("*")
# Все файлы, которые содержат spec в имени
Dir.glob("*spec*")
# Все csv файлы
Dir.glob("*.csv")
```

Работа с файлами и папками

```ruby
# Путь к текущей директории
Dir.pwd
# Проверка, существует ли папка
Dir.exists?("desire")
# Все csv файлы
Dir.glob("*.csv")
```

Работа с файлами и папками

Для лабораторной работы также понадобится набор утилит, позволяющих сравнивать файлы. Для этого необходимо в начале файла подключить `gem "fileutils"` и использовать метод `compare_file` из класса `FileUtils`

```
require 'fileutils'

FileUtils.compare_file("a.txt", "b.txt")
```

Теория

Задание

SOFTWARE ENGINEERING

Задание #1

У вас есть файл, со следующей структурой и содержащий не менее 10 строк: Имя, фамилия и возраст

Прим.

Иван Иванов 20

Петр Петров 21

Необходимо прочитать файл, затем запросив у пользователя ввод возраста, найти в результатах чтения файла, студента чей возраст равен введенному числу и записать этого студента(тов) в другой файл с названием `results.txt.` После этого перезапросить ввод.

Программа завершается выводом на экран содержимого файла `results.txt` построчно, если все студенты из первого файла были записаны во второй или если пользователь ввел с клавиатуры `-1`

SOFTWARE ENGINEERING

Задание #2

Напишите программу, которая начинается с чтения банковского баланса клиента из файла с именем `balance.txt`. Этот файл содержит одну строку со стартовым балансом клиента. Если файл не существует, используйте стартовый баланс 100.0, который должен быть константой.

После этого программа повторно предложит клиенту внести деньги, вывести деньги, проверить баланс или выйти, используя буквы `D (deposit)`, `W (withdraw)`, `B (balance)` и `Q (quit)`. Программа должна принимать на вход значения в верхнем или нижнем регистре.

Задание #2

Для депозитов (D), программа подсказывает сумму. Сумма должна быть больше нуля. Если сумма действительна, программа добавляет сумму депозита к балансу клиента и отображает новый баланс.

При снятии средств (W) программа выдаст запрос на сумму. Сумма должна быть больше нуля и меньше или равна текущему балансу. Если сумма корректна, то программа вычитает сумму вывода из баланса клиента и отображает новый баланс.

Задание #2

Для проверки баланса (B) программа просто выводит текущий баланс.

Когда клиент решит выйти из программы (Q), программа запишет текущий баланс обратно в файл balance.txt.

В случае неправильного ввода (команда или сумма), ваша программа должна выдать соответствующее сообщение об ошибке, которое говорит клиенту, как ее исправить. Нельзя просто выводить "Error!" - это не поможет.

КОНТАКТЫ

Для быстрого входа
на сайт

courses@crimeadigital.ru

tel: 8 (800) 551 44 86

https://crimeadigital.ru