Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ** **РАБОТЕ**

Язык программирования Ruby

Тема

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Преподаватель |  |  |  |  |  | Д.А.Евдокимов |
|  |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент |  | КИ19-16/2б, 031940418 |  |  |  | С.Е.Шпаков |
|  |  | номер группы, зачетной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2020

# СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc50479296)

[2 О языке Ruby 3](#_Toc50479297)

[2.1 История языка 3](#_Toc50479298)

[2.2 Введение в язык 3](#_Toc50479299)

[2.3 Ruby в современном мире 4](#_Toc50479300)

[3 Сравнение с другими языками программирования 5](#_Toc50479301)

[3.1 Сравнение с языком Python 5](#_Toc50479302)

[3.2 Сравнение с языком PHP 6](#_Toc50479303)

[4 Выполнение практического задания 7](#_Toc50479304)

[4.1 Условия задачи 7](#_Toc50479305)

[4.2 Реализация программы на языке Ruby 8](#_Toc50479306)

[5 Вывод 15](#_Toc50479307)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 16](#_Toc50479308)

# 1 Цель работы

Для выполнения учебной практики поставлены следующие задачи:

* изучить основы языка программирования (ЯП) Ruby;
* сравнить его с другими языками программирования;
* выполнить последнее задание по основам программирования на Ruby.

# 2 О языке Ruby

## 2.1 История языка

Язык программирования Ruby разработан японским программистом Юкихиро Мацумото в 1995 году. Целью Юкихиро было создать язык, который будет иметь самые лучшие подходы других языков, тем самым облегчая труд программиста. Как вторая мотивация выступало желание создать истинно объектно-ориентированный, простой в использовании высокоуровневый язык.

Ruby следует принципу «наименьшей неожиданности». Это значит, кто программа будет вести себя в соответствии с ожиданиями программиста

Вдохновением послужил язык Perl. Многие особенности синтаксиса и семантики перешли в Ruby. Собственно, и само название весьма близко по значению.

В Японии язык стал популярен очень быстро, с 1995 по 2002 год вышло около двадцати книг о Ruby. Выход на международный уровень начался после перевода документации на английский язык и создании первых рассылок в 1998 году. Настоящий всплеск интереса произошел, однако, при появлении фреймворка Ruby-on-Rails (RoR), направленного на разработку web-приложений.

## 2.2 Введение в язык

Для определения языка Ruby, можно отметить следующие особенности:

* это высокоуровневый ЯП. Имеет лаконичный и простой синтаксис, рассчитанный на простоту чтения;
* полностью объектно-ориентированный (все данные являются объектами);
* интерпретируемый, рефлективный;
* имеет сильную динамическую типизацию;
* содержит автоматический сборщик мусора, который работает для всех объектов языка;
* сравнительно низкая длина кода;
* и др.

## 2.3 Ruby в современном мире

Уже не первый год по интернету ходит мем «Ruby мертв» и вопрос это довольно спорный. Сейчас на российском рынке действительно определенное количество компаний уходят с языка Ruby, преимущественно на Go. Обусловлено это в основном тем, что Ruby достаточно требователен к ресурсам, в основном к памяти. Однако, сам язык программирования Ruby развивался на протяжении многих лет и продолжает развиваться по сей день, что просто не позволяет считать язык «мертвым». Он точно продолжит становиться лучше и лучше, ведь как шутил сам создатель на одной из конференций: «Вы же понимаете, что мы не можем позволить Ruby умереть? Вы-то ладно, вы другие языки выучите. А я-то вообще без работы останусь».

Хотя язык и считается универсальным по применению, но основную популярность ему дает фреймворк Ruby-on-rails для разработки web-приложений и естественно большая часть вакансий будут связаны именно с «рельсами», а не чистым Ruby. Достаточно простой и очень функциональный фреймворк, на котором написано множество крупных проектов, живущих по сей день.

Одним из больших плюсов, который выделяют о языке – это его понятность и читаемость в долгосрочном значении. Считается большой проблемой, из-за которой программисты быстро уходят из компаний, где их просят продолжить развитие какого-то проекта десятилетней давности, ведь разобраться в нем целое дело. Тем временем код десятилетней давности на Ruby понятен не хуже, чем тогда.

Стоит отметить, что хоть на российском рынке востребованность Ruby и падает, за рубежом, а в частности в Америке, его спрос остается большим. В том числе он считается очень востребованным в стартапах в силу того, что на Ruby можно быстро делать MVP и прототипы.

# 3 Сравнение с другими языками программирования

## 3.1 Сравнение с языком Python

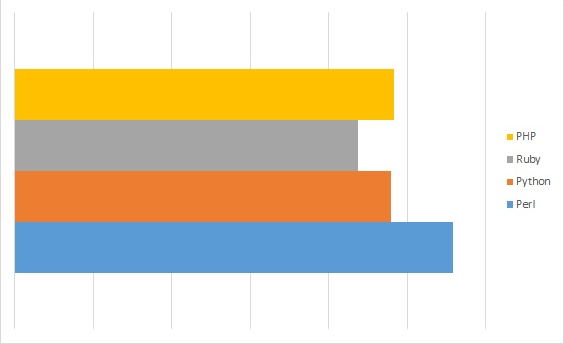
Язык Python активно применяется во многих сферах разработки, таких как: веб, мобильная разработка, игры, большие данные, искусственный интеллект, сетевое администрирование. Тем временем Ruby может соревноваться только в веб-разработке.

Считается, что Python очень прост в написании по скорости и понятности. Если оценивать объем кода, то Python действительно останется победителем, однако понятность кода выше в Ruby из-за «человечности» языка, что делает работу с ним более удобной.

Если оценивать перспективы, то у Ruby они значительно круче относительно веб-разработки, ведь уже существующий RoR обладает большими возможностями. Однако, Python всё больше уходит от веба в другие дисциплины, что делает сравнение этих языков не просто сложным, а неправильным, ведь векторы их развития совершенно разные.

И Ruby, и Python обвиняют в достаточно низкой производительности. Однако, согласно специальным тестам, оценка времени выполнения кода на Ruby обходит своих «основателей», в том числе Python (рисунок 1).

Рисунок 1 – Время выполнения тестов на разных языках



## 3.2 Сравнение с языком PHP

Язык PHP является, наверное, самым популярным в интернете в качестве создания динамических и интерактивных веб-сайтов. Php довольно простой язык, особенно тем, кто переходит с C. Вакансий на него много, но разработчиков во много раз больше, что создает большую конкуренцию. Естественно это всё относительно веб разработки.

Ruby рекомендуется тем, кто уже знает несколько языков программирования. Хоть он и сделан с расчетом на простое написание и легко читаемость, тем не менее новичкам будет тяжелее.

За большим количеством разработчиков следует и большое количество документации. Какой бы у вас не было проблемы на PHP – скорее всего кто-то с ней уже столкнулся и написал свой ответ на вашем же языке. В то же время на языке Ruby даже на англоязычных источниках не всегда можно найти то, что нужно.

Что касается функциональности – естественный победитель Ruby с знаменитым фреймворком Ruby on Rails.

# 4 Выполнение практического задания

## 4.1 Условия задачи

Сущности А и Б имеют отношение многие ко многим. Написать программу, моделирующую это отношение.

Исполнители и альбомы.

А: исполнитель (название коллектива или ФИО, год начала концертной деятельности, страна происхождения).

Б: альбом (наименование, год выпуска, тираж).

У одного исполнителя может быть несколько альбомов, один альбом может быть написан несколькими исполнителями.

На оценку 3 балла:

а) выполнить работу в соответствии с заданием;

б) добавление элементов типа А и Б;

в) создавать связь между элементами типа А и Б.

На оценку 4 балла:

а) выполнить требования предыдущих пунктов;

б) добавить возможность сохранять и загружать данные из файла (в

т.ч. связи);

в) выводить на экран все элементы А или Б (на усмотрение

пользователя);

г) выводить на экран все элементы А, связанные с выбранным

элементом Б и наоборот;

На оценку 5 баллов:

а) выполнить требования предыдущих пунктов;

б) добавить возможность удаления выбранного элемента типа А или

типа Б;

в) добавить сортировку по одному из выбранных полей.

## 4.2 Реализация программы на языке Ruby

Ниже представлена программа на языке программирования Ruby, в соответствии с заданием (листинг 1).

Листинг 1 – Реализация программы

class Producer

@@pro\_list = []

attr\_reader :name, :date, :country

attr\_accessor :albums, :pro\_list

def initialize name, date, country, \*albs

@@pro\_list << name

@name = name

@date = date

@country = country

@albums = albs

end

def self.get\_list

@@pro\_list

end

def self.set\_list list

@@pro\_list = list

end

def to\_s

temp = "Name's #{@name} from #{@country}. Started career back in #{@date}."

if self.albums.size > 0

albs = ""

self.albums.each do |n|

albs += " #{n},"

end

return temp + " Their albums:#{albs}"

else

return temp + " Has no albums :c"

end

end

def output

"#{@name}%#{@date}%#{@country}%#{@albums};"

end

end

class Album

@@album\_list = []

attr\_reader :name, :date, :circulation

attr\_accessor :producers, :album\_list

def initialize name, date, circ, \*prods

@@album\_list << name

@name = name

@date = date

@circulation = circ

@producers = prods

end

Продолжение листинга 1

def self.get\_list

@@album\_list

end

def self.set\_list list

@@album\_list = list

end

def to\_s

temp = "Album '#{@name}' from #{@date}. Has a circulation of #{@circulation}."

if self.producers.length > 0

prods = ""

self.producers.each do |n|

prods += "#{n},"

end

return temp + " Produced by:#{prods}"

else

return temp + " Artist's unknown."

end

end

def output

"#{@name}%#{@date}%#{@circulation}%#{@producers};"

end

end

def main\_menu

puts "Please choose one of the following:\n" \

"1. Show Artists\n" \

"2. Show Albums\n" \

"3. Search by Artist\n" \

"4. Search by Album\n" \

"5. Sorting settings\n" \

"6. Add new Artist\n" \

"7. Add new Album\n" \

"8. Load from file\n" \

"9. Save to file\n" \

"10. Exit\n"

end

module ManageInfo

def self.add\_artist

puts "> Please fill in the information. Type 'back' to return."

artistname = ""

date = ""

country = ""

loop do

puts "> What's the name of the artist"

answer = gets.chomp

puts answer

if answer == ""

puts "> They should have a name"

elsif answer.downcase.eql?("back")

return nil

elsif Producer.get\_list.include?(answer)

puts "> Artist with this name is already in the list."

else

artistname = answer

break

end

end

Продолжение листинга 1

loop do

puts "> When did they start their career? YYYY/MM/DD preferrably"

answer = gets.chomp

if answer == ""

puts "> But they started at least somewhen, right?"

elsif answer.downcase == "back"

return nil

else

temp = answer.split('/')

if temp.size == 3 and temp.all?{Numeric}

date = answer

break

else

puts "> Invalid date."

end

end

end

loop do

puts "Where are they from?"

answer = gets.chomp

if answer == ""

puts "But we are all originated somewhere!"

elsif answer.downcase == "back"

return nil

else

country = answer

break

end

end

return artistname, date, country

end

def self.assign\_albums

puts "What albums did they take part in? Type in one at the time. " \

"Click enter if there's none (no more)"

albums = []

loop do

answer = gets.chomp

if answer == ""

break

else

albums << answer

unless Album.get\_list.include?(answer)

puts "This album is not listed yet. Make sure to do it afterwards!"

end

end

end

return albums

end

def self.add\_album

puts "Please fill in the information. " \

"Type 'back' to return"

name = ''

year = ''

circ = ''

loop do

puts "What's the name of an album?"

answer = gets.chomp

if answer == ''

puts "It's got to have a name."

elsif answer.downcase == "back"

return nil

Продолжение листинга 1

else

if not Album.get\_list.include?(answer)

name = answer

break

else

puts "Album with this name exists."

end

end

end

loop do

puts "When did the album release? (year)"

answer = gets.chomp

if answer == ''

puts "It has to appear somewhen in the timeline."

elsif answer.downcase == "back"

return nil

elsif [answer].all?{Numeric}

year = answer.to\_i

break

else

puts "It doesn't look like a 'year'."

end

end

loop do

puts "What's the circulation of the album? In numbers"

answer = gets.chomp

if answer == ''

puts "Does it really exist?"

elsif [answer].all?{Numeric}

circ = answer.to\_i

break

else

puts "Weird amount."

end

end

return name, year, circ

end

def self.assign\_prods

puts "What artists took part in it? Type in one at the time." \

"Click enter if there's none (no more)"

prods = []

loop do

answer = gets.chomp

if answer == ""

break

else

prods << answer

unless Producer.get\_list.include?(answer)

puts "This artist is not listed yet. Make sure to do it afterwards!"

end

end

end

return prods

end

def self.loadf

arttext = File.read("artists.txt")

albtext = File.read("albums.txt")

return arttext, albtext

end

def self.savef outart, outalb

file = File.new("artists.txt", "w+")

Продолжение листинга 1

file.write(outart)

file.close

nfile = File.new("albums.txt", "w+")

nfile.write(outalb)

nfile.close

end

end

def main

allArtists = []

allAlbums = []

sortby = ['name', 'name']

loop do

main\_menu

choice = gets.chomp.to\_i

case choice

when 1

if allArtists.length > 0

case sortby[0]

when "name"

allArtists.sort{|a,b| a.name <=> b.name}.each do |art|

puts art

end

when "date"

allArtists.sort{|a,b| a.date <=> b.date}.each do |art|

puts art

end

when "country"

allArtists.sort{|a,b| a.country <=> b.country}.each do |art|

puts art

end

end

else

puts "Nothing to show :c"

end

when 2

if allAlbums.length > 0

case sortby[1]

when "name"

allAlbums.sort{|a,b| a.name <=> b.name}.each do |alb|

puts alb

end

when "date"

allAlbums.sort{|a,b| a.date <=> b.date}.each do |alb|

puts alb

end

when "circulation"

allAlbums.sort{|a,b| a.circulation <=> b.circulation}.each do |alb|

puts alb

end

end

else

puts "Nothing to show :c"

end

when 3

puts "> Type in the name of the artist (case sensitive). Type 'back' to return."

loop do

answer = gets.chomp

if answer == ''

Продолжение листинга 1

puts "> Empty search received. Type something in."

elsif answer.downcase == 'back'

break

elsif not Producer.get\_list.include?(answer)

puts "Nothing found."

else

art = allArtists.detect{|x| answer == x.name}

puts "Information about the artist:\n#{art}"

if art.albums.length > 0

puts "Extra information about albums:"

art.albums.each do |alb|

if Album.get\_list.include?(alb)

puts allAlbums.detect{|x| x.name == alb}

end

end

end

break

end

end

when 4

puts "> Type in the name of the album (case sensitive). Type 'back' to return."

loop do

answer = gets.chomp

if answer == ''

puts "> Empty search received. Type something in."

elsif answer.downcase == 'back'

break

elsif not Album.get\_list.include?(answer)

puts "> Nothing found."

else

alb = allAlbums.detect{|x| answer == x.name}

puts "Information about the album:\n#{alb}"

if alb.producers.length > 0

puts "Extra information about artists:"

alb.producers.each do |art|

if Producer.get\_list.include?(art)

puts allArtists.detect{|x| x.name == art}

end

end

end

break

end

end

when 5

loop do

puts "Sort Artists by:\n" \

"1. Name\n" \

"2. Date\n" \

"3. Country\n" \

"4. <Skip>"

answer = gets.chomp.to\_i

case answer

when 1

sortby[0] = "name"

break

when 2

sortby[0] = "date"

break

when 3

sortby[0] = "country"

Продолжение листинга 1

break

when 4

break

else

puts "There's no such option"

end

end

loop do

puts "Sort Albums by:\n" \

"1. Name\n" \

"2. Date\n" \

"3. Circulation\n" \

"4. <Skip>"

answer = gets.chomp.to\_i

case answer

when 1

sortby[1] = "name"

break

when 2

sortby[1] = "date"

break

when 3

sortby[1] = "circulation"

break

when 4

break

else

puts "There's no such option"

end

end

when 6

name, date, country = ManageInfo.add\_artist

unless name == nil

album = ManageInfo.assign\_albums

allArtists << Producer.new(name, date, country, \*album)

end

when 7

name, year, circ = ManageInfo.add\_album

unless name == nil

prods = ManageInfo.assign\_prods

allAlbums << Album.new(name, year, circ, \*prods)

end

when 8

art, alb = ManageInfo.loadf

art = art.split(';')

art.each do |single|

name, date, country, album = \*single.split('%')

album = album[2...-2]

allArtists << Producer.new(name, date, country, \*album.split('", "'))

end

alb = alb.split(';')

alb.each do |single|

name, year, circ, prods = \*single.split('%')

prods = prods[2...-2]

allAlbums << Album.new(name, year, circ, \*prods.split('", "'))

end

when 9

outart = ""

allArtists.each do |x|

outart += x.output

Продолжение листинга 1

end

outalb = ""

allAlbums.each do |y|

outalb += y.output

end

ManageInfo.savef outart, outalb

when 10

break

else

puts "There is no such option."

end

end

end

main

# 5 Вывод

За период летней практики была изучена база языка программирования Ruby, его плюсы и минусы, актуальность и история. Оценена популярность среди языков конкурентов.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 Система менеджмента качества Общие требования к построению, изложению и оформлению документов учебной деятельности. – Взамен СТО 4.2–07–2012 ; введ. 09.01.2014, - Красноярск : ИПК СФУ, 2014 – 60 с.