МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра програмного забезпечення комп’ютерних систем

**ЗВІТ**

**з виробничої практики у підприємсті Vallsoft**

студента \_\_343\_\_ групи

\_\_\_\_\_\_\_\_\_  \_Ротар Я. О.\_\_

      (підпис)              (прізвище, ініціали)

Керівник практики від ЗВО

\_\_\_\_\_\_\_\_\_  Тарновецька О. Ю.

      (підпис)                (прізвище, ініціали)

Екзаменаційна оцінка з виробничої практики:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № пп | Прізвище, ініціали студента | Підсумкова оц.за шкалою ВНЗ | Підсумкова оц.за шкалою ECTS | Підсумкова оц.за націонал. Шкалою | Підпис керівника практики |
| 1. | Ротар Я.О | 60 | D | Задовільно |  |

Чернівці, 2020

**ЗМІСТ**

1. МЕТА І ЗАВДАННЯ ПРАКТИКИ…………………………………………….3

2. ЗМІСТ ПРАКТИКИ…………………………………………………………….4

3. ПРОГРАМА ПРАКТИКИ……………………………………………………...5

4. КАЛЕНДАРНИЙ ГРАФІК ПРОХОДЖЕННЯ ПРАКТИКИ………………..6

5. ІНСТРУКТАЖ ПО ТЕХНІЦІ БЕЗПЕКИ ТА ПОЖЕЖНІЙ БЕЗПЕЦІ……..7

6. ФУНКЦІОНАЛЬНІ ОБОВ’ЯЗКИ ПІД ЧАС ПРАКТИКИ………………….8

7. ОХОРОНА ПРАЦІ…………………………………………………………….9

8. ВИСНОВОК…………………………………………………………………...10

9. ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ……………………………………….11

10. ДОДАТКИ. КОД ПРОГРАМИ……………………………………………...12

1. **МЕТА І ЗАВДАННЯ ПРАКТИКИ**

Метою практики є розширення технічного кругозору за допомоги вивчення нової мови Ruby, її особливостей та можливостей. Використання цієї мови для написання якісних та сучасних програм.

Виробнича практика покликана сформувати у майбутнього випускника вищого навчального закладу професійні вміння та навички, розвивати самостійність при прийнятті рішень під час виконання прикладних завдань, регламентованих посадовими обов'язками фахівця.

Задачі практики:

* вивчення мови Ruby, її синтаксису;
* набуття навичок творчого підходу до вирішення теоретичних та практичних задач, що виникають при проектуванні, розробці і експлуатації програмного забезпечення;
* узагальнення, систематизація, закріплення і поглиблення знань по дисциплінах спеціальності «Інженерія програмного забезпечення»;
* вивчення заходів щодо техніки безпеки, охорони праці, протипожежної безпеки, охорони навколишнього середовища і цивільної оборони;

1. **ЗМІСТ ПРАКТИКИ**

Зміст виробничої практики полягає у вирішенні практичних проблем, використовуючи мову Ruby та її можливості. У ході практики я ознайомився зі стандартами написання програм мовою Ruby.

На практиці вдалось:

* вивчити основи нової мови;
* закріпити знання, отримані в університеті
* закріпити теоретичні знання по якості програмного забезпечення;

1. **ПРОГРАМА ПРАКТИКИ**

|  |  |
| --- | --- |
| **№** | **Назва теми** |
| 1 | Ознайомлення з Ruby та відповідними середовищами розробки |
| 2 | Введення у Ruby. Типи даних та регулярні вирази |
| 3 | Масиви даних |
| 4 | Хеш-функції |
| 5 | Умовні оператори |
| 6 | Методи |
| 7 | Класи |
| 8 | Модулі |
| 9 | Виключення |
| 10 | Введення у RubyOnRails |

*Тема 1. Ознайомлення з Ruby та відповідними середовищами розробки*

Мета, завдання та вимоги до проходження практики. Огляд програми практики.

Обговорення календарного плану проходження практики. Правила оформлення щоденника практики. Правила оформлення письмового звіту.

Умови захисту практики.

Ознайомлення з мовою та її синтаксисом. Встановлення IDE.

*Тема 2. Введення у Ruby. Типи даних та регулярні вирази*

Ознайомлення з типами даних та регулярними виразами. Виконання завдань.

*Тема 3. Масиви даних*

Ознайомлення з масивами даних та їхнім застосуванням. Виконання завдань.

*Тема 4. Хеш-функції*

Ознайомлення з хеш-функціями та їхнім застосуванням. Виконання завдань.

*Тема 5. Умовні оператори*

Ознайомлення з умовними операторами та їхнім застосуванням. Виконання завдань.

*Тема 6. Методи*

Ознайомлення з методами та їхнім застосуванням. Виконання завдань.

*Тема 7. Класи*

Ознайомлення з класами та їхнім застосуванням. Виконання завдань.

*Тема 8. Модулі*

Ознайомлення з модулями та їхнім застосуванням. Виконання завдань.

*Тема 9. Виключення*

Ознайомлення з виключеннями та їхнім застосуванням. Виконання завдань.

*Тема 10. Введення у RubyOnRails*

Ознайомлення з RubyOnRails та вивчення базового матеріалу про неї.

1. **КАЛЕНДАРНИЙ ГРАФІК ПРОХОДЖЕННЯ ВИРОБНИЧОЇ ПРАКТИКИ**

|  |  |  |
| --- | --- | --- |
| **Дата виконання** | **Зміст виконаної роботи** | **Відмітки про виконання** |
| 27.06 - 28.06 | Ознайомлення з Ruby та відповідними середовищами розробки | Виконано |
| 29.06 - 30.06 | Введення у Ruby. Типи даних та регулярні вирази | Виконано |
| 01.07 | Масиви даних | Виконано |
| 04.07 - 06.07 | Хеш-функції | Виконано |
| 07.07 – 08.07 | Умовні оператори | Виконано |
| 11.07 – 13.07 | Методи | Виконано |
| 14.07 – 18.07 | Класи | Виконано |
| 19.07 – 20.07 | Модулі | Виконано |
| 21.07 – 22.07 | Виключення | Виконано |
| 25.07 – 28.07 | Введення у RubyOnRails | Виконано |

**5. ІНСТРУКТАЖ ПО ТЕХНІЦІ БЕЗПЕКИ ТА ПОЖЕЖНІЙ БЕЗПЕЦІ**

Перед початком роботи в організації керівником практики проведено інструктаж з техніки безпеки. Основні положення інструктажу:

* заборонено сідати за ПК у верхньому одязі, знімати його необхідно у спеціально відведеному місці;
* починати роботу можна лише за вказівкою керівника практики;
* виконувати слід тільки зазначене керівником практики завдання, категорично забороняється виконувати інші роботи;
* у випадку виникнення несправностей засобів праці на робочому місці, необхідно повідомити керівника практики;
* на робочому місці не потрібно залишати зайвих предметів, всі особисті речі необхідно залишати у спеціально відведеному місці;
* користування принтерами, сканерами, ксероксами дозволяється тільки у присутності керівника практики;
* в разі ураження людини електричним струмом, необхідно знеструмити місце де відбувся витік електроенергії, надати першу лікарську допомогу і викликати «швидку медичну допомогу»;
* у разі недотримання вимог із охорони праці та пожежної безпеки можливе притягнення до дисциплінарної або адміністративної відповідальності.
* суворо забороняється чіпати роз’єми з’єднувальних кабелів, торкатися до проводів живлення і пристроїв заземлення; доторкатися до екрана з тильного боку монітора;
* дотримувати безпечну відстань від екрана монітора до очей, зберігати правильну осанку;
* не можна працювати при недостатньому освітленні та при поганому самопочутті.

**6. ФУНКЦІОНАЛЬНІ ОБОВ’ЯЗКИ ПІД ЧАС ПРАКТИКИ**

Під час проходження практики моє завдання полягало у вивченні мови, синтаксису та особливостей. Використання системи Git для надання доступу до результатів моєї праці для керівника практикою. Ознайомлення з IDE RubyMine.

Опис виконаної роботи:

1. Виконано всі початкові завдання: Написаний “Hello World”, встановлення IDE, приєднання до репозиторію зі створенням своєї гілки
2. Вивчені типи даних та виконано два завдання по регулярних виразах
3. Вивчені масиви даних та виконано сім завдань по них
4. Вивчені хеш-функції та виконано два завдання по них
5. Вивчені умовні оператори та виконано чотири завдання по них
6. Вивчено методи та виконано два завдання по них
7. Вивчено класи та виконано чотири завдання по них
8. Вивчено модулі та виконано одне завдання по них
9. Вивчено виключення та виконано три завдання по них
10. Пройдений вступ у RubyOnRails

**7. ОХОРОНА ПРАЦІ**

При роботі за комп’ютером необхідно знати та виконувати наступні інструкції:

1. Виконувати умови інструкції з експлуатації ПК.
2. При експлуатації ПК необхідно пам’ятати, що первинні мережі електроспоживання під час роботи знаходяться під напругою, яка є небезпечною для життя людини, тому необхідно користуватися справними розетками, відгалужувальними та з’єднувальними коробками, вимикачами та іншими електроприладами.
3. До роботи з ПК допускаються працівники, з якими проведений вступний інструктаж та первинний інструктаж (на робочому місці) з питань охорони праці, техніки безпеки, пожежної безпеки та зроблений запис про їх проведення у спеціальному журналі інструктажів.
4. Працівники при роботі з ПК повинні дотримуватися вимог техніки безпеки, пожежної безпеки.
5. При виявленні в обладнанні ПК ознак несправності (іскріння, пробоїв, підвищення температури, запаху гару, ознак горіння) необхідно негайно припинити роботи, відключити усе обладнання від електромережі і терміново повідомити про це відповідних посадових осіб, спеціалістів.
6. Вміти діяти в разі ураження інших працівників електричним струмом або виникнення пожежі.
7. Знати місця розташування первинних засобів пожежогасіння, план евакуації працівників, матеріальних цінностей в приміщенні в разі виникнення пожежі.

**8. ВИСНОВОК**

Під час проходження практики я скористався чудовою можливістю вивчити нову мову програмування та ознайомитись із новими середовищами розробки. Я сконцентрував свої зусилля на вдосконаленні своїх навичок та набутті нових.

**9. ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ**

1. Документація Ruby: <https://ruby-doc.com/>
2. Гілка проєкту з практичними завданнями: <https://github.com/nromanen/ruby-school_summer-2022/tree/M_Bidiuk>

**10. ДОДАТКИ**

**Код програм**

## README.md

Ruby school 2022

**intro** - hello world

**lec\_1** - regex

**lec\_2** - arrays

**lec\_3-5** - hashes

**lec\_6** - conditionals

**lec\_7-8** - methods

**lec\_9** - classes

**lec\_10** - modules

**lec\_11** – exceptions

**intro/hello\_world.rb**

puts 'Hello world'

**lec\_1/task\_1.rb**

def is\_uppercase(str)

str =~ /[A-Z]/

end

def is\_lowercase(str)

str =~ /[a-z]/

end

def is\_number(str)

str =~ /\d/

end

def is\_symbol(str)

str =~ /[^\p{L}\p{N}]/

end

def interpolate(str)

puts "#{is\_uppercase(str)}\_#{is\_lowercase(str)}\_#{is\_number(str)}\_#{is\_symbol(str)}"

end

interpolate('#####')

interpolate('abc1')

interpolate('abcЯ1')

**lec\_1/task\_2.rb**

def match(str)

puts "#{str} - #{'not ' unless str =~ /^\p{Lu}\p{Ll}{0,14}(-\p{Lu}\p{Ll}{0,14})?$/}match"

end

match('Ффффффффффффффф')

match('І-Ф')

match('Анна1')

match('Hello')

match('Anna-Henrietta')

match('Anna-henrietta')

**lec\_2/arr\_1.rb**

def to\_power(n)

i = 0

arr = Array.new

while i <= n do

arr.push(2\*\*i)

i += 1

end

puts arr

end

to\_power(2)

**lec\_2/arr\_2.rb**

def sq\_sum(arr)

print("#{arr.map { |e| e \*\* 2 }}\n#{arr.sum}")

end

sq\_sum([1, 2, 2])

**lec\_2/arr\_3.rb**

def city\_hello(name, city, state)

puts "Hello, #{name.join(" ")}! Welcome to #{city}, #{state}!"

end

city\_hello(["John", "Smith"], "Phoenix", "Arizona")

**lec\_2/arr\_4.rb**

def gps(s, x)

if x.length <= 1

return 0

end

delta = 0

max = 0

i = 0

while i < x.length - 1

delta = (x[i + 1].to\_f - x[i].to\_f).abs \* (60 / s) \* 60

if delta > max

max = delta

end

i += 1

end

puts max.floor

end

gps(15, [0.0, 0.19, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25])

**lec\_2/arr\_5.rb**

def number\_elements(arr)

i = 0

while i < arr.size do

arr.map { |e| puts "#{i+=1}: #{e}" }

i += 1

end

end

number\_elements(["a", "b", "c"])

number\_elements([])

**lec\_2/arr\_6.rb**

def square\_or\_power(arr)

res = arr.map do |e|

if Math.sqrt(e)% 1 == 0

Math.sqrt(e).round

else

e\*\*2

end

end

print(res)

end

square\_or\_power([4,3,9,7,2,1])

**lec\_2/arr\_7.rb**

def fill\_by\_length (n, f, s)

arr = Array.new

i = 0

while i < n do

if i % 2 == 0

arr.push(f)

else

arr.push(s)

end

i += 1

end

print("#{arr}\n")

end

fill\_by\_length(5, true, false)

fill\_by\_length(10, "blue", "red")

fill\_by\_length(0, "sds", "sdsww")

**lec\_3-5/task\_1/inventory.rb**

module Inventory

def warehouse(array)

res = Hash.new

array.map{ |e| e.scan(/\w+/) }

.each { |e|

if res[e[0].to\_sym].nil?

res[e[0].to\_sym] = e[1].to\_i

else

res[e[0].to\_sym] += e[1].to\_i

end

}

res

end

end

**lec\_3-5/task\_1/inventory\_test.rb**

require 'test/unit'

require\_relative 'inventory'

class InventoryTest < Test::Unit::TestCase

include Inventory

def test\_empty

assert\_equal(Hash.new, warehouse([]), "Result should be empty")

end

def test\_unique

assert\_equal({:one=>18, :two=>3}, warehouse(['one -18', 'two - 3']))

end

def test\_duplicate

assert\_equal({:one=>20, :two=>3}, warehouse(['one -18', 'two - 3', 'one-2']))

end

end

**lec\_3-5/task\_2/student\_scores.rb**

def students\_by\_scores(max, low, up, students)

scores = Hash[top: [], middle: [], bottom: []]

min\_range = 0 .. max \* low / 100

max\_range = max - up .. max

split\_scores = students.split(/[\s,']/)

.each\_slice(2)

.to\_a

split\_scores.map { |student, score| [student, score.to\_i] }

.sort { |student, score| score[1] <=> student[1] }

.each { |e|

if min\_range.include?(e[1])

scores[:bottom].push(e[0])

elsif max\_range.include?(e[1])

scores[:top].push(e[0])

else

scores[:middle].push(e[0])

end

}

puts scores

end

students\_by\_scores(100, 15, 10,

"S1,5\nS2,18\nS3,45\nS4,90\nS5,100\nS6,94\nS7,1\nS8,13\nA,100\nB,13\nC,18\nS9,2")

**lec\_6/merge.rb**

def merge(arr)

start\_i = 0

end\_i = 1

while end\_i < arr.length do

if arr[start\_i] == arr[end\_i]

arr[start\_i] = arr[start\_i] + arr[end\_i]

arr[end\_i] = 0

start\_i+=1

end

end\_i+=1

end

print("#{arr.sort.reverse}\n")

end

merge([2,0,2,2]) # 4,2,0,0

merge([2,0,2,4]) # 4,4,0,0

merge([4,4,0,4]) # 8,4,0,0

merge([2,2,4,16]) # 16,4,4,0

**lec\_6/pangram.rb**

def pangram(str)

split\_str = str.chars.map(&:downcase)

alphabet = ('a'..'z').to\_a

res = true

alphabet.each{ |e|

unless split\_str.include? e.downcase

res = false

break

end

}

puts res

end

pangram("The quick brown fox jumps over the lazy dog")

pangram("The q123uick brown fox jump!!!!s over the lazy dog....")

pangram("The brown fox jumps over the lazy dog")

**lec\_6/rain\_fall.rb**

data =

"Rome:Jan 81.2,Feb 63.2,Mar 70.3,Apr 55.7,May 53.0,Jun 36.4,Jul 17.5,Aug 27.5,Sep 60.9,Oct 117.7,Nov 111.0,Dec 97.9" + "\n" +

"London:Jan 48.0,Feb 38.9,Mar 39.9,Apr 42.2,May 47.3,Jun 52.1,Jul 59.5,Aug 57.2,Sep 55.4,Oct 62.0,Nov 59.0,Dec 52.9" + "\n" +

"Paris:Jan 182.3,Feb 120.6,Mar 158.1,Apr 204.9,May 323.1,Jun 300.5,Jul 236.8,Aug 192.9,Sep 66.3,Oct 63.3,Nov 83.2,Dec 154.7" + "\n" +

"NY:Jan 108.7,Feb 101.8,Mar 131.9,Apr 93.5,May 98.8,Jun 93.6,Jul 102.2,Aug 131.8,Sep 92.0,Oct 82.3,Nov 107.8,Dec 94.2" + "\n" +

"Vancouver:Jan 145.7,Feb 121.4,Mar 102.3,Apr 69.2,May 55.8,Jun 47.1,Jul 31.3,Aug 37.0,Sep 59.6,Oct 116.3,Nov 154.6,Dec 171.5" + "\n" +

"Sydney:Jan 103.4,Feb 111.0,Mar 131.3,Apr 129.7,May 123.0,Jun 129.2,Jul 102.8,Aug 80.3,Sep 69.3,Oct 82.6,Nov 81.4,Dec 78.2" + "\n" +

"Bangkok:Jan 10.6,Feb 28.2,Mar 30.7,Apr 71.8,May 189.4,Jun 151.7,Jul 158.2,Aug 187.0,Sep 319.9,Oct 230.8,Nov 57.3,Dec 9.4" + "\n" +

"Tokyo:Jan 49.9,Feb 71.5,Mar 106.4,Apr 129.2,May 144.0,Jun 176.0,Jul 135.6,Aug 148.5,Sep 216.4,Oct 194.1,Nov 95.6,Dec 54.4" + "\n" +

"Beijing:Jan 3.9,Feb 4.7,Mar 8.2,Apr 18.4,May 33.0,Jun 78.1,Jul 224.3,Aug 170.0,Sep 58.4,Oct 18.0,Nov 9.3,Dec 2.7" + "\n" +

"Lima:Jan 1.2,Feb 0.9,Mar 0.7,Apr 0.4,May 0.6,Jun 1.8,Jul 4.4,Aug 3.1,Sep 3.3,Oct 1.7,Nov 0.5,Dec 0.7"

data1 =

"Rome:Jan 90.2,Feb 73.2,Mar 80.3,Apr 55.7,May 53.0,Jun 36.4,Jul 17.5,Aug 27.5,Sep 60.9,Oct 147.7,Nov 121.0,Dec 97.9" + "\n" +

"London:Jan 58.0,Feb 38.9,Mar 49.9,Apr 42.2,May 67.3,Jun 52.1,Jul 59.5,Aug 77.2,Sep 55.4,Oct 62.0,Nov 69.0,Dec 52.9" + "\n" +

"Paris:Jan 182.3,Feb 120.6,Mar 188.1,Apr 204.9,May 323.1,Jun 350.5,Jul 336.8,Aug 192.9,Sep 66.3,Oct 63.3,Nov 83.2,Dec 154.7" + "\n" +

"NY:Jan 128.7,Feb 121.8,Mar 151.9,Apr 93.5,May 98.8,Jun 93.6,Jul 142.2,Aug 131.8,Sep 92.0,Oct 82.3,Nov 107.8,Dec 94.2" + "\n" +

"Vancouver:Jan 155.7,Feb 121.4,Mar 132.3,Apr 69.2,May 85.8,Jun 47.1,Jul 31.3,Aug 37.0,Sep 69.6,Oct 116.3,Nov 154.6,Dec 171.5" + "\n" +

"Sydney:Jan 123.4,Feb 111.0,Mar 151.3,Apr 129.7,May 123.0,Jun 159.2,Jul 102.8,Aug 90.3,Sep 69.3,Oct 82.6,Nov 81.4,Dec 78.2" + "\n" +

"Bangkok:Jan 20.6,Feb 28.2,Mar 40.7,Apr 81.8,May 189.4,Jun 151.7,Jul 198.2,Aug 197.0,Sep 319.9,Oct 230.8,Nov 57.3,Dec 9.4" + "\n" +

"Tokyo:Jan 59.9,Feb 81.5,Mar 106.4,Apr 139.2,May 144.0,Jun 186.0,Jul 155.6,Aug 148.5,Sep 216.4,Oct 194.1,Nov 95.6,Dec 54.4" + "\n" +

"Beijing:Jan 13.9,Feb 14.7,Mar 18.2,Apr 18.4,May 43.0,Jun 88.1,Jul 224.3,Aug 170.0,Sep 58.4,Oct 38.0,Nov 19.3,Dec 2.7" + "\n" +

"Lima:Jan 11.2,Feb 10.9,Mar 10.7,Apr 10.4,May 10.6,Jun 11.8,Jul 14.4,Aug 13.1,Sep 23.3,Oct 1.7,Nov 0.5,Dec 10.7"

towns = ["Rome", "London", "Paris", "NY", "Vancouver", "Sydney", "Bangkok", "Tokyo",

"Beijing", "Lima", "Montevideo", "Caracas", "Madrid", "Berlin"]

def mean(town, strng)

split\_str = strng.split("\n").select{ |e| e.include?("#{town}:") }

rain\_town = ""

rain\_data = ""

sum = 0.0

split\_str.each { |str|

temp = str.split(":")

if town == temp[0]

rain\_town = town

rain\_data = temp[1]

end

}

if rain\_town == ""

return -1

end

rain\_value = rain\_data.split(",")

rain\_value.each { |str|

temp = str.split(" ")

sum += temp[1].to\_f

}

sum / 12.0

end

def variance(town, strng)

split\_str = strng.split("\n").select{ |e| e.include?("#{town}:") }

rain\_town = ""

rain\_data = ""

sum = 0.0

split\_str.each { |str|

temp = str.split(":")

if town == temp[0]

rain\_town = town

rain\_data = temp[1]

end

}

if rain\_town == ""

return -1

end

avg = mean(town, strng)

rain\_value = rain\_data.split(",")

rain\_value.each { |str|

temp = str.split(" ")

sum += (temp[1].to\_f - avg.to\_f)\*\*2

}

sum / 12.0

end

puts mean("London", data)

puts mean("Sydney", data)

puts mean("sdshdsdsd", data)

puts variance("London", data)

puts variance("Sydney", data)

puts variance("asddsaasd", data)

**lec\_6/rectangle\_into\_squares.rb**

def rectangle\_into\_squares(length, width)

arr = Array.new

if length != width

while length > 0 && width > 0 do

arr.push(length > width ? width : length)

length > width ? length -= width : width -= length

end

else

nil

end

print("#{arr}\n")

end

rectangle\_into\_squares(5,5)

rectangle\_into\_squares(5,3)

rectangle\_into\_squares(5,4)

**lec\_7-8/task\_1.rb**

def sum(arr)

sum = 0

i = 0

while i < arr.size do

sum += yield(arr[i])

i+=1

end

puts sum

end

sum([4, 0, -12]) { |e| e < 0 ? e\*\*2 : e }

**lec\_7-8/task\_2.rb**

def check\_password(user, password)

Proc.new { |u, p| u == user && p == password }

end

admin = check\_password("admin", "123")

puts admin.call("admin", "123")

puts admin.call("admsdfsdfsdfin", "123")

puts admin.call("admin", "fhfghf")

puts admin.call("admin")

**lec\_9/task\_1-2/mark.rb**

class Mark

attr\_accessor :subject

attr\_accessor :mark

def initialize(subject, mark)

@subject = subject

@mark = mark

end

def to\_s

"#{@subject}: #{@mark}"

end

end

**lec\_9/task\_1-2/student.rb**

class Student

attr\_accessor :name

attr\_accessor :marks

def initialize(name)

@name = name

@marks = Array.new

end

def exists?(subject\_title)

@marks.select { |e|

e.subject == subject\_title

}.length != 0

end

def add\_mark(mark)

unless mark.instance\_of? Mark

return

end

if exists?(mark.subject)

nil

else

@marks.push(mark)

end

end

def to\_s

"#{@name} = #{@marks.map { |e| e.to\_s }}"

end

end

**lec\_9/task\_1-2/retake\_student.rb**

class RetakeStudent < Student

def initialize(name)

super

end

def exists?(subject\_title)

super

end

def add\_mark(mark)

unless mark.instance\_of? Mark

return

end

if exists?(mark.subject)

@marks.reject! { |e|

e.subject == mark.subject

}

@marks.push(mark)

else

@marks.push(mark)

end

end

def to\_s

super

end

end

**lec\_9/task\_1-2/student\_marks.rb**

require\_relative 'mark.rb'

require\_relative 'student.rb'

require\_relative 'retake\_student.rb'

mark1 = Mark.new('C++', 5)

mark2 = Mark.new('C+++', 5)

mark3 = Mark.new('C', 4)

mark4 = Mark.new('Java', 1)

mark4\_2 = Mark.new('Java', 5)

puts "~~~~~~~~~~~~~1~~~~~~~~~~~~~~"

student = Student.new('Student')

student.add\_mark(mark1)

student.add\_mark(mark2)

student.add\_mark(mark3)

student.add\_mark(mark4)

puts student

student.add\_mark(mark4\_2)

puts student

puts "~~~~~~~~~~~~~~"

puts student.exists?('C++')

puts student.exists?('C+++')

puts student.exists?('asdasda')

puts "~~~~~~~~~~~~~2~~~~~~~~~~~~~~"

new\_student = RetakeStudent.new('New Student')

new\_student.add\_mark(mark1)

new\_student.add\_mark(mark2)

new\_student.add\_mark(mark3)

new\_student.add\_mark(mark4)

puts new\_student

new\_student.add\_mark(mark4\_2)

puts new\_student

puts "~~~~~~~~~~~~~~"

puts new\_student.exists?('C++')

puts new\_student.exists?('Javarr')

puts new\_student.exists?('asdasda')

**lec\_9/task\_3/pizza.rb**

class Pizza

attr\_accessor :ingredients

attr\_accessor :number

attr\_accessor :d

@@order\_count = 0

@@fail\_count = 0

def initialize(d, ingredients)

if d == 25 || d == 50

@d = d

@ingredients = ingredients

@@order\_count += 1

@number = @@order\_count

else

@number = 0

@d = 0

@ingredients = []

@@fail\_count += 1

end

end

def self.failInstance

@@fail\_count

end

def to\_s

"#{@number} - #{@d}, #{@ingredients}"

end

end

**lec\_9/task\_3/order\_pizza.rb**

require\_relative 'pizza.rb'

pizza1 = Pizza.new(10, ['mushroom', 'cheese'])

pizza2 = Pizza.new(25, ['mushroom', 'cheese', 'cherry'])

pizza3 = Pizza.new(50, ['mushroom'])

pizza4 = Pizza.new(100, ['mushroom', 'cheese', 'cherry'])

puts pizza1.to\_s

puts pizza2.to\_s

puts pizza3.to\_s

puts pizza4.to\_s

puts Pizza.failInstance

**lec\_9/task\_4/author.rb**

class Author

attr\_accessor :f\_name

attr\_accessor :l\_name

attr\_accessor :patronymic

@@Authors = []

def initialize(f\_name, l\_name, patronymic)

@f\_name = f\_name

@l\_name = l\_name

@patronymic = patronymic

end

def self.add\_author(f\_name, l\_name, patronymic = '')

if f\_name.match(/[^A-Za-z-]/).eql?(nil) && l\_name.match(/[^A-Za-z-]/).eql?(nil) &&

(patronymic == '' ? true : patronymic.match(/[^A-Za-z]/).eql?(nil))

if f\_name.length < 15 && l\_name.length < 15 && patronymic.length < 15

duplicate\_check = @@Authors.none? do |e|

e.eql?([f\_name, l\_name, patronymic])

end

if duplicate\_check

@@Authors.push([f\_name, l\_name, patronymic])

end

end

end

end

def self.find\_author(f\_name, l\_name, patronymic = '')

@@Authors.select do |e|

e.eql?([f\_name, l\_name, patronymic])

end

end

def self.print

i = 0

while i <= @@Authors.size

puts "#{@@Authors[i].join(" ")}"

i+=1

end

end

end

**lec\_9/task\_4/book.rb**

class Book

attr\_accessor :name

attr\_accessor :publisher

attr\_accessor :year

attr\_accessor :authors

def initialize(name, publisher, year)

year\_now = Time.now.year

if !name.match(/[\W.,-;] /).eql?(nil) ||

!publisher.match(/[\W.,-;] /).eql?(nil) ||

((year <= year\_now) && (year.to\_s.length == 4))

@name = name

@publisher = publisher

@authors = Array.new

@year = year

else

@name = nil

@publisher = publisher

@authors = nil

@year = nil

end

end

def add\_author(author)

@authors.push(author)

end

def to\_s

"Name: #{@name}\n" +

"Publisher: #{@publisher}\n" +

"Year: #{@year}\n" +

"Authors: #{@authors.join(" ")}\n\n"

end

end

**lec\_9/task\_4/create\_book.rb**

require\_relative 'author.rb'

require\_relative 'book.rb'

auth1 = Author.add\_author("John", "Tolkien", "RR")

auth1\_2 = Author.add\_author("John", "Tolkien", "RR")

auth2 = Author.add\_author("Author", "Authorsson")

auth3 = Author.add\_author("Author", "Authorsberg")

auth4 = Author.add\_author("Charles", "Author")

book1 = Book.new("LOTR", "КСД", 2009)

book2 = Book.new("The Hobbit", "КСД", 2010)

book3 = Book.new("The Hobbit", "КСД", 2210)

book1.add\_author(Author.find\_author("John", "Tolkien", "RR"))

book2.add\_author(Author.find\_author("Author", "Authorsson"))

puts book1.to\_s

puts book2.to\_s

**lec\_10/console\_input.rb**

module ConsoleInput

def string

print "Enter any string:\n"

gets.chomp.to\_s

end

def number

print "Enter any number:\n"

gets.chomp.gsub(/[.,]/, '.' => '.', ',' => '.').to\_f

end

def symbol

print "Enter any letters:\n"

gets.chomp.downcase.gsub(/\s+/, "").to\_sym

end

def new\_hash

print "Enter hash in format 'key: value' or 'key => value':\n"

str = gets.chomp

hash\_str = str.scan(/\w+/).each\_slice(2).to\_h

end

end

**lec\_10/module\_task.rb**

require\_relative 'console\_input.rb'

class Task

include ConsoleInput

end

task = Task.new

puts task.string

puts task.number

puts task.symbol

puts task.new\_hash

**lec\_11/task\_1/console\_input.rb**

module ValidType

def float?

true if Float self rescue false

end

def int?

true if Integer self rescue false

end

def valid\_int?

int? & float?

end

def valid\_float?

!int? & float?

end

end

module ConsoleInput

String.include ValidType

def string

print "Enter any string:\n"

gets.chomp.to\_s

end

def number

print "Enter any number:\n"

res = gets.chomp.gsub(/[.,]/, '.' => '.', ',' => '.')

begin

if res.valid\_float?

res.to\_f

elsif res.valid\_int?

res.to\_i

else

raise ArgumentError.new("Error when entering data; should be float (x.xx) or int (x)")

end

rescue Exception => e

print "#{e.message} -> #{res}"

end

end

def symbol

print "Enter any letters:\n"

gets.chomp.downcase.gsub(/\s+/, "").to\_sym

end

def new\_hash

print "Enter hash in format 'key: value' or 'key => value':\n"

str = gets.chomp

begin

unless str.include?('=>') || str.include?(':')

raise ArgumentError.new("Error when entering data; required format is 'key: value' or 'key => value'")

end

hash\_str = str.scan(/\w+/).each\_slice(2).to\_h

rescue Exception => e

print "#{e.message} -> #{str}"

end

end

end

**lec\_11/task\_1/module\_task.rb**

require\_relative 'console\_input.rb'

class Task

include ConsoleInput

end

task = Task.new

puts task.string

puts task.number

puts task.symbol

puts task.new\_hash

**lec\_11/task\_2/phone\_format\_exception.rb**

class PhoneFormatException < StandardError

attr\_reader :error

def initialize(phone)

@error = "Phone number #{phone} is incorrect. You should enter phone in format +38dddddddddd.\n"

end

def message

@error

end

end

**lec\_11/task\_2/validation.rb**

module Validation

def phone\_valid?(phone)

phone.match?(/^\+38\d{10}$/)

end

end

**lec\_11/task\_2/user.rb**

require\_relative 'phone\_format\_exception.rb'

require\_relative 'validation.rb'

class User

include Validation

attr\_reader :name

attr\_writer :phone

def initialize(name)

@name = name

end

def phone=(phone)

begin

unless phone\_valid?(phone)

raise PhoneFormatException.new(phone)

end

@phone = phone

rescue PhoneFormatException => e

print e.message

end

end

def to\_s

"Name: #{@name}, Phone: #{@phone}\n"

end

end

**lec\_11/task\_2/enter\_phone.rb**

require\_relative 'user.rb'

user = User.new('New User')

user.phone = '380999999999'

user.phone = '+380999999999'

puts user

**lec\_11/task\_3/retry\_task.rb**

def third\_order(max)

try = 1

begin

number = rand(max)

res = number\*\*3

puts "#{number} \* #{number} \* #{number} = "

answer = gets.chomp.to\_i

if answer.eql?(res)

puts "Good job!"

else

try += 1

end

rescue

if try > 3

puts "Fail"

else

retry

end

end

end

third\_order(6)