**Министерство образования и науки Украины**

**Национальный горный университет**

Институт электроэнергетики

Факультет информационных технологий

**Кафедра ПЗКС**

Лабораторная работа № 4

**по дисциплине**

“Професійна практика програмної інженерії”

**Название** **работы**: Одномерные и многомерные массивы.

**Тема** Использование одномерных и многомерных массивов в решении математических задач.

**Цель**: Использование языковых конструкций языка Ruby для написания и отладки программ.

**Задание**: Математическое обоснование заданий для программирования берется из файла с заданиями к Лабораторной работе № 4

**ВЫПОЛНИЛ:**

студент группы Пиит-15-1

Сенькин Д. Р.

**ПРОВЕРИЛ:**

Коротенко Л.М.

**Днепр**

**2017**

**Условие задачи:**

5. Дан целочисленный массив А(13), содержащий отрицательные, нулевые и положительные элементы. Построить новый массив В, который содержит в начале все нулевые элементы, затем все отрицательные, а затем все положительные элементы.

**1.**

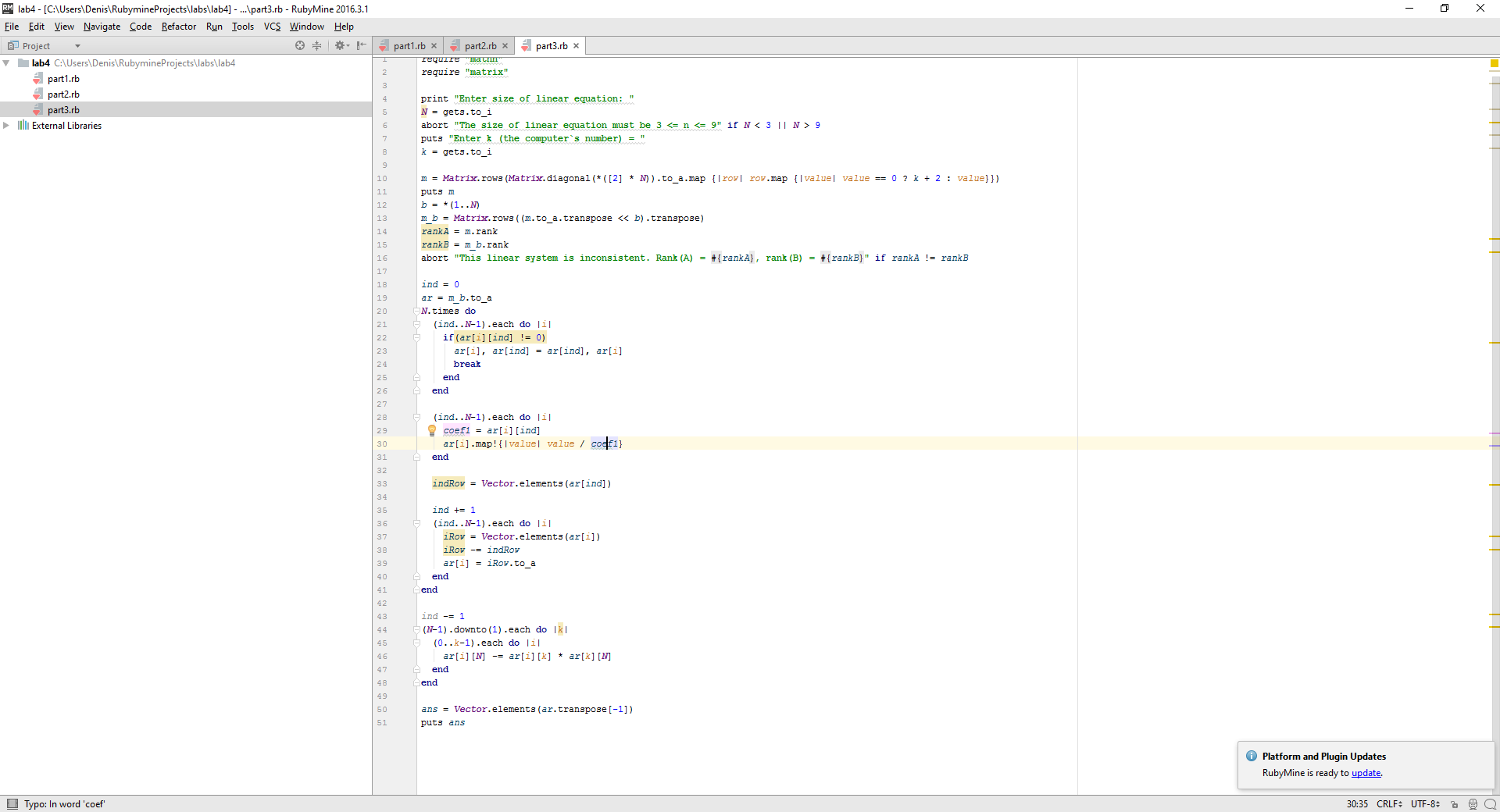
a = ***Array***.new(11) {rand(-100.0..100.0)}  
a += ***Array***.new(2, 0)  
puts **"a =** #{a}**"***b* = a.sort  
*tmp* = *b*.*count*(0.0)  
*b*.delete(0.0)  
*b* = ***Array***.new(*tmp*, 0.0) + *b*puts **"b =** #{*b*}**"**

**2.**

*require* **"mathn"***require* **"matrix"  
*X***, ***Y*** = ***Vector***.elements(***Array***.new(8) {rand(10)}), ***Vector***.elements(***Array***.new(8) {rand(10)})  
  
**class *Matrix* def** *[]=*(*i*, *j*, *value*)  
 **@rows**[*i*][*j*] = *value* **end  
end***res* = ***Matrix***.build(***X***.size, ***Y***.size) {0}  
**for** *i* **in** 0..***X***.size-1 **do  
 for** *j* **in** 0..***Y***.size-1 **do** *res*[*i*, *j*] = ***X***[*i*] \* ***Y***[*j*]  
 **end  
end**puts *res*

**3.**

*require* **"mathn"***require* **"matrix"**print **"Enter size of linear equation: "  
*N*** = gets.to\_i  
abort **"The size of linear equation must be 3 <= n <= 9" if *N*** < 3 || ***N*** > 9  
puts **"Enter k (the computer`s number) = "***k* = gets.to\_i  
  
*m* = ***Matrix***.rows(***Matrix***.diagonal(\*([2] \* ***N***)).to\_a.map {|*row*| *row*.map {|*value*| *value* == 0 ? *k* + 2 : *value*}})  
puts *m  
b* = \*(1..***N***)  
*m\_b* = ***Matrix***.rows((*m*.to\_a.transpose << *b*).transpose)  
*rankA* = *m*.rank  
*rankB* = *m\_b*.rank  
abort **"This linear system is inconsistent. Rank(A) =** #{*rankA*}**, rank(B) =** #{*rankB*}**" if** *rankA* != *rankB  
  
ind* = 0  
*ar* = *m\_b*.to\_a  
***N***.times **do** (*ind*..***N***-1).each **do** |*i*|  
 **if**(*ar*[*i*][*ind*] != 0)  
 *ar*[*i*], *ar*[*ind*] = *ar*[*ind*], *ar*[*i*]  
 **break  
 end  
 end** (*ind*..***N***-1).each **do** |*i*|  
 *coef1* = *ar*[*i*][*ind*]  
 *ar*[*i*].map!{|*value*| *value* / *coef1*}  
 **end** *indRow* = ***Vector***.elements(*ar*[*ind*])  
  
 *ind* += 1  
 (*ind*..***N***-1).each **do** |*i*|  
 *iRow* = ***Vector***.elements(*ar*[*i*])  
 *iRow* -= *indRow  
 ar*[*i*] = *iRow*.to\_a  
 **end  
end***ind* -= 1  
(***N***-1).downto(1).each **do** |*k*|  
 (0..*k*-1).each **do** |*i*|  
 *ar*[*i*][***N***] -= *ar*[*i*][*k*] \* *ar*[*k*][***N***]  
 **end  
end***ans* = ***Vector***.elements(*ar*.transpose[-1])  
puts *ans*



**5. Результат работы программы**

**1.**



**2.**

****

**3.**

