

# 1INF06 Data Structures and Methodical Programming

Loaiza Vasquez, Manuel Alejandro  
Oyarce Tocto, Elizabeth Patricia  
Saras Rivera, André Edgardo

Octubre 8, 2020

Pontificia Universidad Católica del Perú  
Lima, Perú

[manuel.loaiza@pucp.edu.pe](mailto:manuel.loaiza@pucp.edu.pe)  
[a20182778@pucp.edu.pe](mailto:a20182778@pucp.edu.pe)  
[andre.saras@pucp.edu.pe](mailto:andre.saras@pucp.edu.pe)

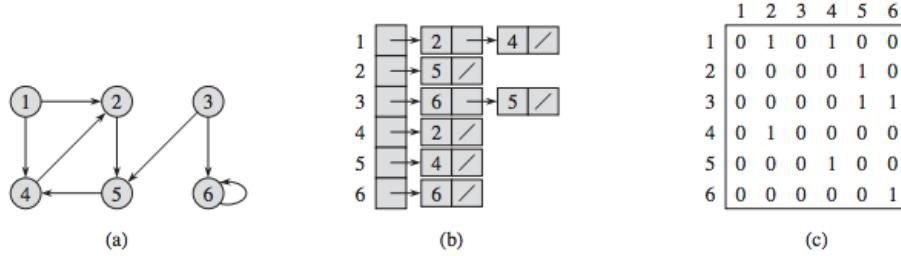
Third report of the course Data Structures and Methodical Programming taught at the Faculty of Science and Engineering at Pontificia Universidad Católica del Perú (PUCP) by Viktor Khlebnikov.

## Contents

<b>1</b>	<b>Linked data structure research: Double Linked List</b>	<b>2</b>
1.1	Implementation . . . . .	3
1.2	Testing . . . . .	6
<b>2</b>	<b>Oyarce Tocto, Elizabeth Patricia</b>	<b>9</b>
2.1	Resources and Oracle VM Virtual Box parameters . . . . .	9
2.2	Language research: Object Oriented Programming in Ruby . . . . .	10
2.3	Example problem: Add Two Numbers . . . . .	15
<b>3</b>	<b>Saras Rivera, André Edgardo</b>	<b>18</b>
3.1	Resources and Oracle VM Virtual Box parameters . . . . .	18
3.2	Language research: Ruby Basic Syntax and Commands . . . . .	19
3.3	Example problem: Traffic Recreation . . . . .	20
<b>4</b>	<b>Loaiza Vasquez, Manuel Alejandro</b>	<b>23</b>
4.1	Resources and Oracle VM Virtual Box parameters . . . . .	23
4.2	Language research: Arrays, strings, exceptions and files . . . . .	24
4.3	Example problem: Merged k Sorted Lists . . . . .	29

## 1 Linked data structure research: Double Linked List

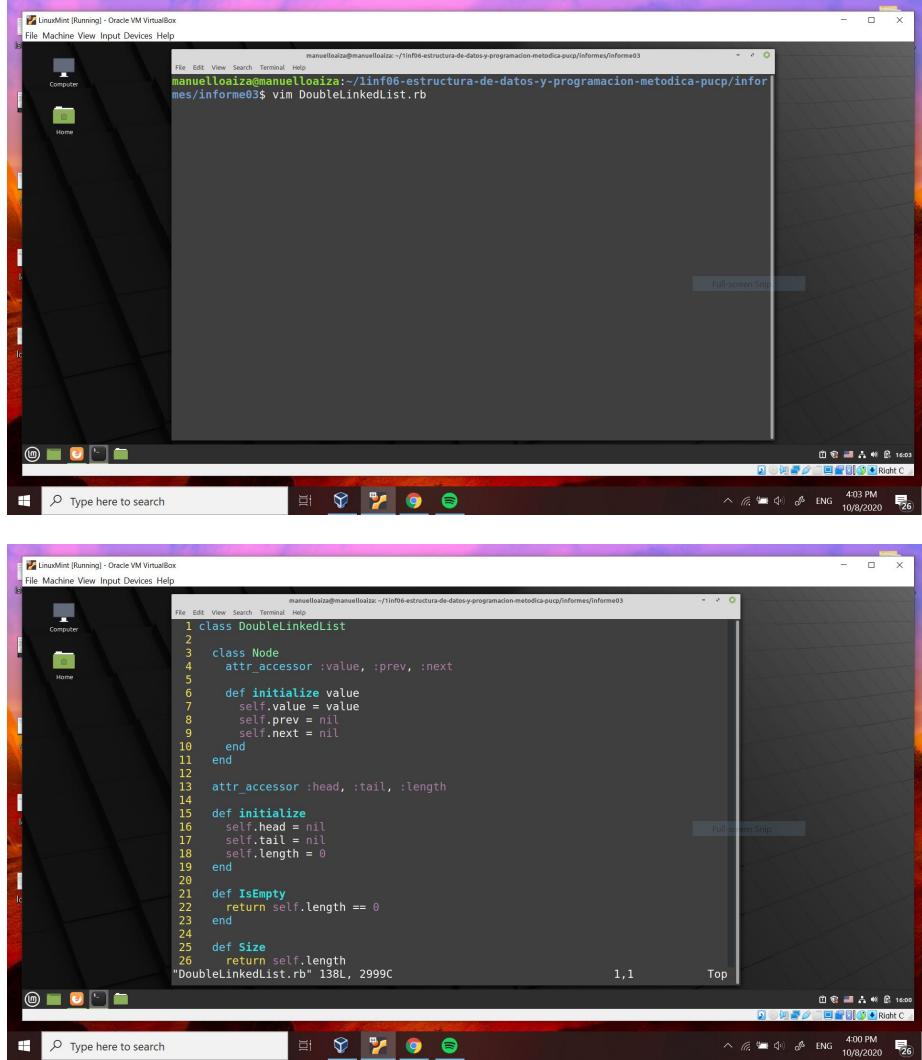
The structures we have to research and implement the following reports are graphs. There are two common ways to represent a graph: using an adjacency matrix or using an adjacency list. Given a graph  $G = (V, E)$ , the adjacency matrix representation is easier because we are going to create an array of  $|V|^2$  units of memory and  $a_{ij}$  will represent if an edge between the nodes  $i$  and  $j$  exists. However, we need a lot of memory to represent the graph and, also, there will be too much unused space. As a consequence, the adjacency list is the most efficient, in terms of space complexity, way to represent a graph because we are going to create nodes on each list that represent each existing edge if necessary.



As we can see, we are going to represent a graph like the one in (a) using an array of double linked lists as shown in (b). First, we have to implement a double linked list in Ruby and then, each member is going to solve a Google coding interview problem using our data structure in our programs. In our next report, we are going to use this data structure to represent a graph and traverse it with algorithms like Depth First Search and Breadth First Search.

In the following pages, we are going to show the implementation of the double linked list data structure and the explanation is inside the code as comments. In `main.rb`, we have prepared some tests with assertions, which are called with the statement `raise` in Ruby, to check if our data structure has bugs. Also, the method `Print` that we have define is going to help to see how our list is growing or shrinking after each operation.

## 1.1 Implementation



```
manuelloaiza@manuelloaiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ vim DoubleLinkedList.rb
```

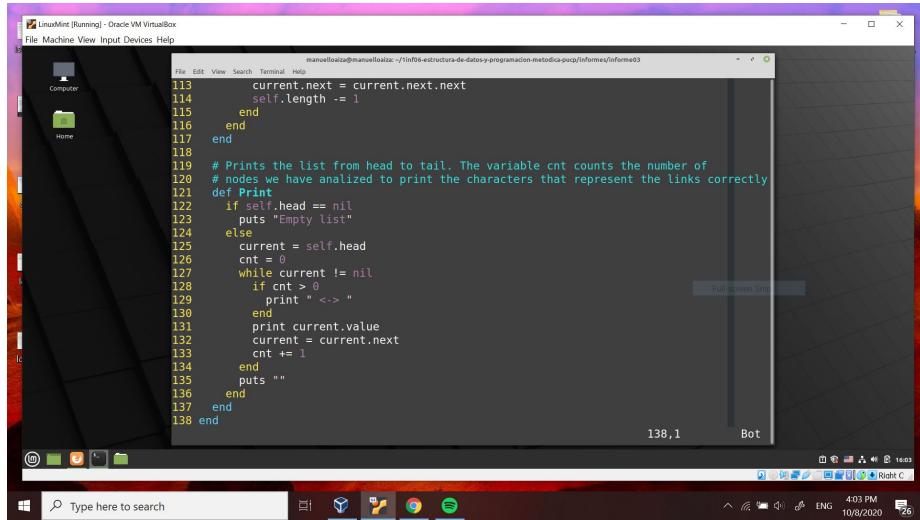
```
1 class DoubleLinkedList
2
3   class Node
4     attr_accessor :value, :prev, :next
5
6     def initialize value
7       self.value = value
8       self.prev = nil
9       self.next = nil
10    end
11  end
12
13  attr_accessor :head, :tail, :length
14
15  def initialize
16    self.head = nil
17    self.tail = nil
18    self.length = 0
19  end
20
21  def isEmpty
22    return self.length == 0
23  end
24
25  def Size
26    return self.length
"DoubleLinkedList.rb" 138L, 2999C
```

```
24
25     def Size
26         return self.length
27     end
28
29     def Head
30         return self.head.value
31     end
32
33     def Tail
34         return self.tail.value
35     end
36
37     # Push an element after the tail
38     def AddRight value
39         node = Node.new value
40         if self.head == nil
41             self.head = node
42             self.tail = node
43         else
44             self.tail.next = node
45             node.prev = self.tail
46             self.tail = node
47         end
48         self.length += 1
49     end
```

```
51     # Push an element before the head
52     def AddLeft value
53         node = Node.new value
54         if self.head == nil
55             self.head = node
56             self.tail = node
57         else
58             self.head.prev = node
59             node.next = self.head
60             self.head = node
61         end
62         self.length += 1
63     end
64
65     # Removes the tail node
66     def RemoveRight
67         raise "Empty list" unless self.length > 0
68         if self.length == 1
69             self.head = self.tail = nil
70         else
71             self.tail = self.tail.prev
72             self.tail.next = nil
73         end
74         self.length -= 1
75     end
```

```
manu@manu-OptiPlex-5090:~/Documentos/inf06-estructura-de-datos-y-programacion-metodica-pwp/informes/informe3
```

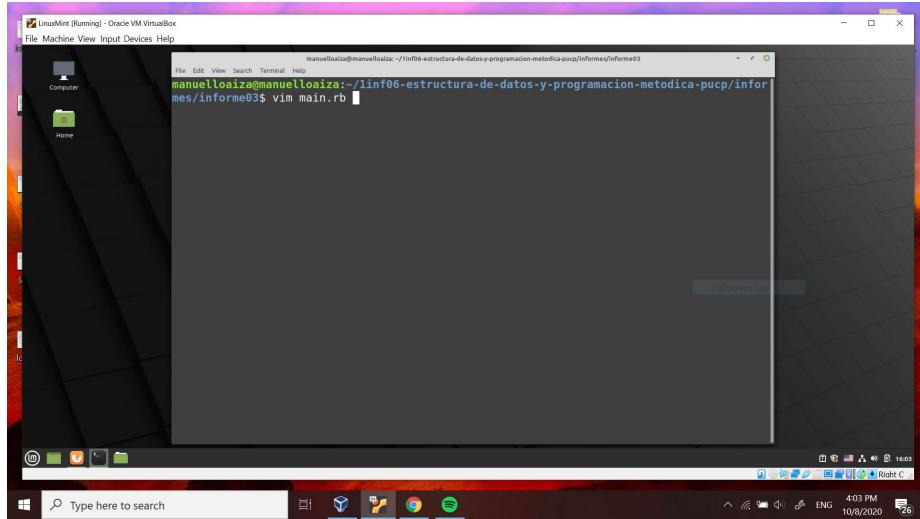
```
77 # Removes the head node
78 def RemoveLeft
79   raise "Empty list" unless self.length > 0
80   if self.length == 1
81     self.head = self.tail = nil
82   else
83     self.head = self.head.next
84     self.head.prev = nil
85   end
86   self.length -= 1
87 end
88
89 # Removes a node with value target in the list. If our target is in our head,
90 # then we are going to remove it first being aware of not having a unitary list.
91 # Otherwise, we are going to look for the node that is immediately before of our
92 # target and then, we are going to link this one and the one that is immediately
93 # after our target.
94 def Remove target
95   raise "Empty list" unless self.length > 0
96   if target == self.head.value
97     if self.head.next == nil
98       self.head = self.tail = nil
99     else
100      self.head.next.prev = nil
101      self.head = self.head.next
102    end
103    self.length -= 1
104  else
105    current = self.head
106    while current != nil && current.next.value != target
107      current = current.next
108    end
109    if current != nil
110      if current.next.next != nil
111        current.next.next.prev = current
112      end
113      current.next = current.next.next
114    end
115  end
116 end
117
118
```



A screenshot of a Linux Mint desktop environment. A terminal window titled "manuelloiza@manuelloiza: ~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03" is open, displaying the following Ruby code:

```
113     current.next = current.next.next
114     self.length -= 1
115   end
116 end
117 end
118
119 # Prints the list from head to tail. The variable cnt counts the number of
120 # nodes we have analized to print the characters that represent the links correctly
121 def Print
122   if self.head == nil
123     puts "Empty list"
124   else
125     current = self.head
126     cnt = 0
127     while current != nil
128       if cnt > 0
129         print " -> "
130       end
131       print current.value
132       current = current.next
133       cnt += 1
134     end
135     puts ""
136   end
137 end
138 end
```

## 1.2 Testing



A screenshot of a Linux Mint desktop environment. A terminal window titled "manuelloiza@manuelloiza: ~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03\$ vim main.rb" is open, showing the command "vim main.rb".

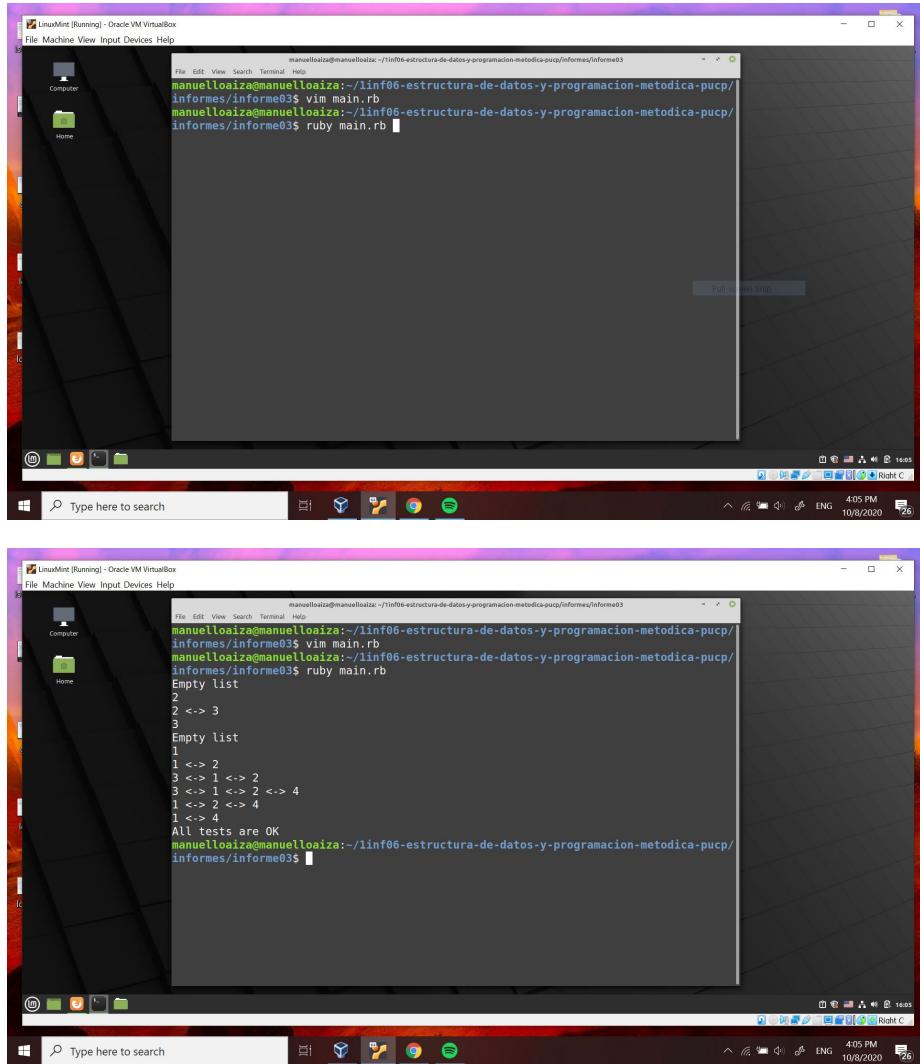
The image displays two terminal windows side-by-side on a Linux Mint desktop. Both windows have a dark theme and show the same command-line interface.

**Left Terminal Window:**

```
manuel@manuel-OptiPlex-5090:~/Desktop/informe03$ require_relative "./DoubleLinkedList.rb"
2
3 list1 = DoubleLinkedList.new
4 raise "Error in IsEmpty" unless list1.isEmpty
5
6 list1.Print
7 list1.AddRight 2
8 list1.Print
9 list1.AddRight 3
10 list1.Print
11 raise "Error in AddRight" unless list1.size == 2
12
13 raise "Error in Head" unless list1.head == 2
14 list1.RemoveLeft
15 list1.Print
16 raise "Error in RemoveLeft" unless list1.size == 1
17 raise "Error in Head" unless list1.head == 3
18 list1.RemoveRight
19 list1.Print
20 raise "Error in RemoveRight" unless list1.isEmpty
21
22 list1.AddLeft 1
23 list1.Print
24 list1.AddRight 2
25 list1.Print
26 list1.AddLeft 3
```

**Right Terminal Window:**

```
manuel@manuel-OptiPlex-5090:~/Desktop/informe03$ require_relative "./DoubleLinkedList.rb"
13 raise "Error in Head" unless list1.head == 2
14 list1.RemoveLeft
15 list1.Print
16 raise "Error in RemoveLeft" unless list1.size == 1
17 raise "Error in Head" unless list1.head == 3
18 list1.RemoveRight
19 list1.Print
20 raise "Error in RemoveRight" unless list1.isEmpty
21
22 list1.AddLeft 1
23 list1.Print
24 list1.AddRight 2
25 list1.Print
26 list1.AddLeft 3
27 list1.Print
28 list1.AddRight 4
29 list1.Print
30 list1.Remove 3
31 list1.Print
32 raise "Error in Remove" unless list1.size == 3
33
34 list1.Remove 2
35 list1.Print
36 raise "Error in Remove" unless list1.size == 2
37
38 puts "All tests are OK"
```



The image shows two screenshots of a Linux Mint desktop environment running in Oracle VM VirtualBox. Both screenshots feature a dark-themed desktop with a window manager interface.

In the top screenshot, a terminal window is open with the following command:

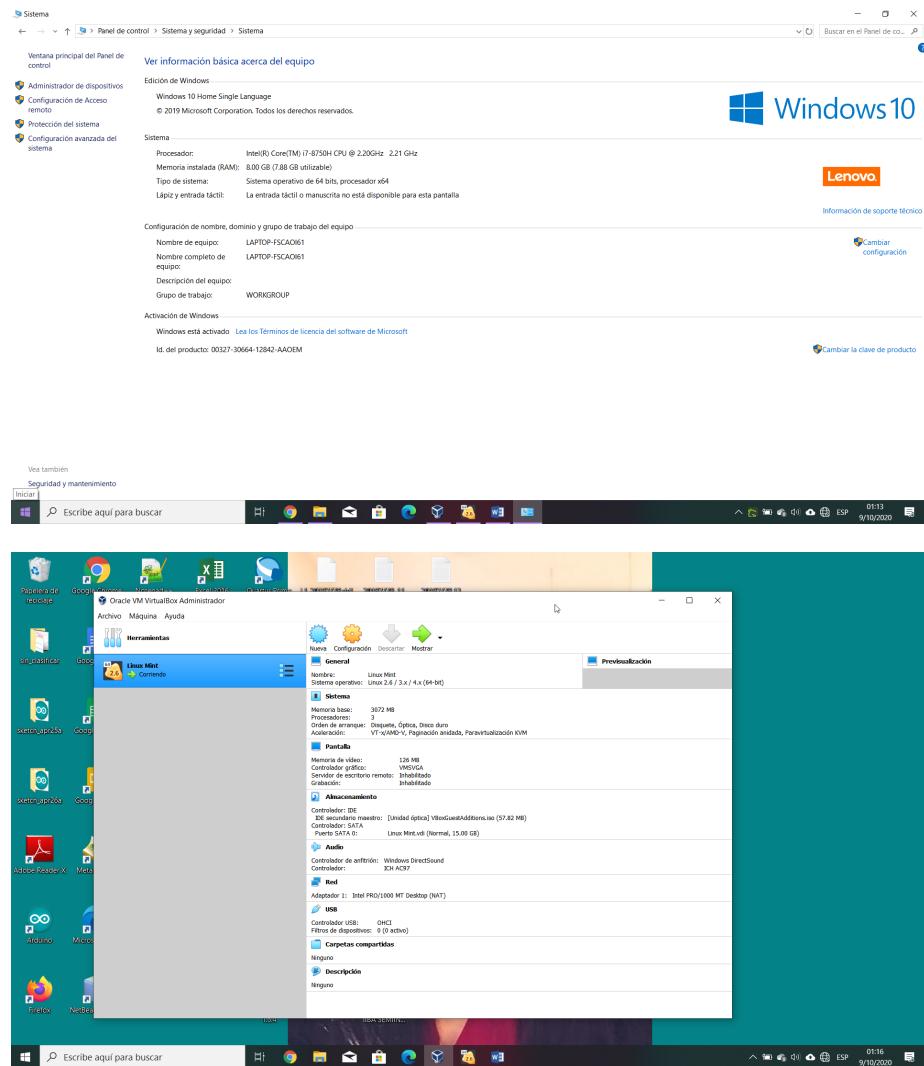
```
manuelloiza@manuelloiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ vim main.rb
```

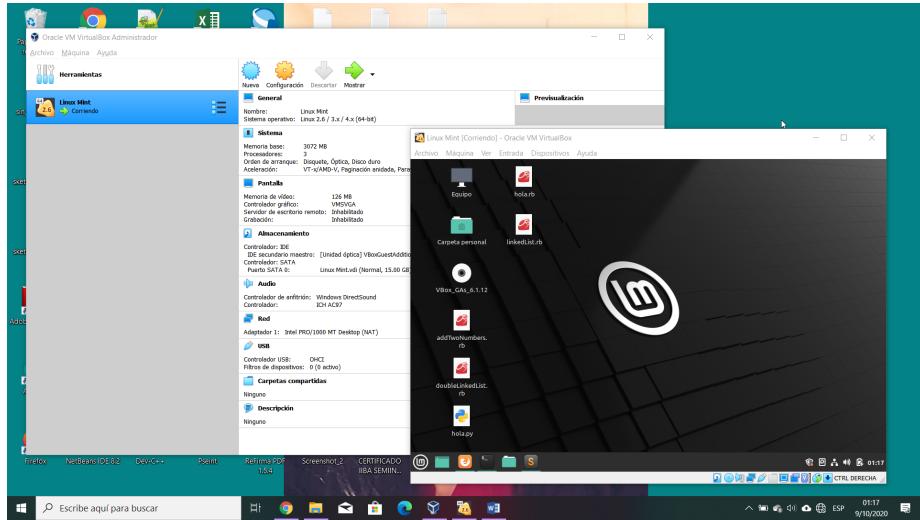
In the bottom screenshot, the terminal window displays the output of the Ruby script:

```
manuelloiza@manuelloiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ ruby main.rb
Empty list
2
2 <-> 3
3
Empty list
1
1 <-> 2
3 <-> 1 <-> 2
3 <-> 1 <-> 2 <-> 4
1 <-> 2 <-> 4
1 <-> 4
All tests are OK
manuelloiza@manuelloiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$
```

## 2 Oyarce Tocco, Elizabeth Patricia

### 2.1 Resources and Oracle VM Virtual Box parameters





## 2.2 Language research: Object Oriented Programming in Ruby

Ruby is an object-oriented programming language, that is, it meets the features of an OOP language.

1. Data Encapsulation
2. Data Abstraction
3. Polymorphism
4. Inheritance

Like all object-oriented programming languages, Ruby involves classes and objects. A class is the abstraction of an object, it defines the behavior that objects will have and for them, it defines attributes, methods, fields and events. The class allows individual objects to be instantiated, therefore a class can be defined as a combination of characteristics and functions.

A class in Ruby always starts with the `class` keyword followed by the class name and finished with the keyword `end`, it is recommended to use camel case. The name must always be in initial capital letters. To exemplify, the class Person can be displayed as:

```
1 class Person
2 end
```

Ruby provides four types of variables:

1. Local Variables, they are the variables that are defined in a method. Local variables are not available outside the method and begin with a lowercase letter.
2. Instance Variables, are available in all methods or anywhere in the class. That means instance variables change from one object to another. Instance variables are preceded by the at sign @ followed by the variable name.
3. Class Variables, are available in different objects. A class variable belongs to the class and to all the objects within it. They are preceded by the @@ sign and followed by the variable name.
4. Global Variables, if you want to have a single variable, which is available in all classes, you must define a global variable. Global variables are always preceded by the dollar sign \$.

Methods are defined within a class as follows.

```

1▼ class Person
2
3  def initialize word
4  |  @name = word
5  end
6
7  def age value
8  |  @age = value
9  end
10 end
11

```

The initialize method is a special type of method (constructor), which will be executed when the `new` method of the class is called with the parameters.

```
me = Person.new "Elizabeth"
```

It should be noted that instance variables cannot be modified or obtained from outside the object. Below, we have an example to access and print the name of a person within the class.

```
class Person
    def initialize word
        @name = word
        puts @name
    end

    def age value
        @age = value
    end
end

me = Person.new "Elizabeth"

elizabeth@Elizabeth-VirtualBox:~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
elizabeth@Elizabeth-VirtualBox:~$ cd Escritorio
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb

Traceback (most recent call last):
conceptos.rb:14:in `<main>': undefined local variable or method `person' for main:Object (NameError)
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb
Elizabeth
elizabeth@Elizabeth-VirtualBox:~/Escritorio$
```

Accessor methods (getters and setters) are used to access a property, getters allow obtaining properties, and setters are used to assign or modify a property.

```
1 class Person
2
3     def initialize word
4         @name = word
5     end
6
7     def age value
8         @age = value
9     end
10
11    def name
12        @name
13    end
14
15    def name=(name2)
16        @name = name2
17    end
18
19
20
21 me = Person.new "Elizabeth"
22 puts me.name
23 me.name="Paty"
24 puts me.name
25
```

elizabeth@Elizabeth-VirtualBox: ~/Escritorio

Archivo Editar Ver Buscar Terminal Ayuda

```
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb
Elizabeth
Paty
elizabeth@Elizabeth-VirtualBox:~/Escritorio$
```

An easier way to access these properties is by defining attributes.

1. `attr_accessor` : getter and setter
2. `attr_reader` : getter
3. `attr_writer` : setter

```

1  class Person
2
3      attr_accessor :name , :age
4
5      def initialize (word)
6          self.name = word
7      end
8
9
10 end
11
12 me= Person.new "Elizabeth"
13 puts me.name
14 me.name ="Paty"
15 puts me.name
16
17
elizabeth@Elizabeth-VirtualBox:~/Escritorio

```

Archivo Editar Ver Buscar Terminal Ayuda

```

elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb
Elizabeth
Paty
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ 

```

Finally, an application of the classes is shown, using the node and list classes, which allow me to concatenate the nodes together to form a simple linked list, some methods are also defined that allow manipulating the nodes. This list will help us to develop the exercise that follows.

```

class LinkedList
    attr_accessor :head, :tail, :length
    class Node
        attr_accessor :value, :next
        def initialize value
            self.value = value
            self.next = nil
        end
    end
    def initialize
        self.head = nil
        self.tail = nil
        self.length = 0
    end
    def isEmpty
        return self.length == 0
    end
    def size
        return self.length
    end
    def head
        return self.head.value
    end
    def tail
        return self.tail.value
    end
    #Push an element after the tail
    def appendRight value
        newNode = Node.new value
        if self.head == nil
            self.head = newNode
            self.tail = newNode
        else
            self.tail.next = newNode
            self.tail = newNode
        end
        self.length += 1
    end
    #Push an element before the head
    def appendLeft value
        newNode = Node.new value
        if self.head == nil
            self.head = newNode
            self.tail = newNode
        else
            newNode.next = self.head
            self.head = newNode
        end
        self.length += 1
    end
    #Push an element after the head
    def insertAfter head, value
        if head == nil
            self.appendRight value
        else
            newNode = Node.new value
            newNode.next = head.next
            head.next = newNode
            self.length += 1
        end
    end
    #Push an element before the head
    def insertBefore head, value
        if head == nil
            self.appendLeft value
        else
            newNode = Node.new value
            newNode.next = head
            head.previous = newNode
            self.length += 1
        end
    end
    #Push an element at the head
    def pushFront value
        self.appendLeft value
    end
    #Push an element at the tail
    def pushBack value
        self.appendRight value
    end
    #Delete an element from the head
    def deleteFront
        if self.length == 0
            return nil
        else
            deletedNode = self.head
            self.head = self.head.next
            self.length -= 1
            return deletedNode.value
        end
    end
    #Delete an element from the tail
    def deleteBack
        if self.length == 0
            return nil
        else
            deletedNode = self.tail
            self.tail = self.tail.previous
            self.length -= 1
            return deletedNode.value
        end
    end
    #Delete an element from the middle
    def deleteValue value
        if self.length == 0
            return nil
        else
            current = self.head
            while current != nil
                if current.value == value
                    if current == self.head
                        self.deleteFront
                    else
                        previous = current.previous
                        previous.next = current.next
                        self.length -= 1
                    end
                end
                current = current.next
            end
        end
    end
    #Print the list
    def printList
        current = self.head
        while current != nil
            puts current.value
            current = current.next
        end
    end
end

```

```

Linux Mint [Corriendo] - Oracle VM VirtualBox
Archivo Maquina Ver Entrada Dispositivos Ayuda
File Edit Selection Find View Goto Tools Project Preferences Help
dosdeelencion.rb 1 linkedList.rb x escribeNumeros.rb
49 # Puts an element before the head
50 def insertBefore(head, node)
51   node.value
52   if self.head == node
53     self.head = node
54     self.tail = node
55   else
56     node.next = self.head
57     self.head = node
58   end
59   self.length += 1
60 end
61
62 # Puts an element after the tail
63 def insertAfter(tail, value)
64   if self.length > 0
65     if self.length == self.length
66       self.head = self.tail = nil
67     else
68       current = self.head
69       while current != nil do current.next != self.tail
70         current = current.next
71       end
72       if current == nil
73         self.tail = nil
74       else
75         self.tail = current
76       end
77     end
78   end
79   self.length += 1
80
81 # Removes the head node
82 def RemoveHead()
83   if self.length > 0
84     raise "Empty list" unless self.length > 0
85     if self.length == self.length
86       self.head = self.tail = nil
87     else
88       self.head = self.head.next
89     end
90   end
91
92 # Removes a node with value target in the list. If our target is in our head,
93 # then we are going to remove it first being aware of not having a unary list.
94 # Otherwise, we are going to look for the node that is immediately before our
95 # target and then, we are going to link this one and the one that is immediately
96 # after our target
97 def Remove(target)
98   if self.length > 0
99     raise "Empty list" unless self.length > 0
100    if target == self.head.value
101      self.head = self.tail = nil
102    else
103      current = self.head
104      while current != nil do
105        if current.value == target
106          if current == self.head
107            self.head = current.next
108          else
109            self.head = self.head.next
110          end
111        end
112      end
113    end
114  end
115
116  # Prints the list from head to tail. The variable cnt counts the number of
117  # characters we have analyzed to print the characters that represent the links correctly.
118  def Print()
119    if self.head == nil
120      raise "Empty list"
121    else
122      current = self.head
123      cnt = 0
124      while current != nil
125        if cnt < 8
126          print current.value
127        end
128        current = current.next
129      end
130    end
131  end
132
133  puts ""
134 end
135
136 end

```

1 -> 2 -> 3 -> 4 -> 5 -> 6

### 2.3 Example problem: Add Two Numbers

You are given two non-negative integers that you must represent in two non-empty linked lists. The digits are stored in reverse order and each of its nodes contains a single digit. Add the two numbers, return the sum and its linked list. You can assume that the two numbers do not contain any leading zeros, except for the number 0 itself.

**Code Block 1: Sublime Text Editor showing the first version of the program (addTwoNumbers.rb) for adding two numbers represented as linked lists.**

```

1  require_relative './linkedlist.rb'
2
3  puts "Enter two numbers non-negatives"
4  #To operate numbers we must transform the
5  #character string to integer values
6
7  x = gets.chomp.to_i
8  y = gets.chomp.to_i
9
10 #Create a copy of the numbers to reserve the
11 #original values
12
13 number1 = x
14
15 number2 = y
16
17 #Generate empty lists representing the integers.
18 #The digits are stored in reverse order, and
19 #each node must contain a single digit.
20 list1=LinkedList.new
21 list2=LinkedList.new
22
23 while number1>0
24   list1.AddDigit(number1%10)
25   number1=number1/10
26 end
27
28 while number2>0
29   list2.AddDigit(number2%10)
30   number2=number2/10
31 end
32
33 #Print lists
34
35 puts "First number list"
36 list1.Print
37 puts "Second number list"
38 list2.Print
39
40 #Get the biggest and shortest size of both lists
41 if list1.size > list2.size
42   biggerSize = list1.size
43   smallerSize = list2.size
44 else
45   biggerSize = list2.size
46   smallerSize = list1.size
47 end
48
49 #Define the variables "carry" and "sum", and the
50 #values of the nodes are added in order, taking into
51 #account that one list may be longer than the other
52 carry=0
53 cnt=0
54
55 list3=LinkedList.new
56 list3.head=nil
57 node1=list1.head
58 node2=list2.head
59
60 while cnt<biggerSize
61   sum=(node1.value+node2.value)+carry
62   sum+=sum%10
63   list3.AddDigit(sum)
64   node1=node1.next
65   node2=node2.next
66   carry=sum/10
67   sum-=node1.value+carry
68   sum+=sum%10
69   list3.AddDigit(sum)
70   node1=node1.next
71   node2=node2.next
72   carry=sum/10
73   sum-=node1.value+carry
74   sum+=sum%10
75   list3.AddDigit(sum)
76   node1=node1.next
77   node2=node2.next
78   carry=sum/10
79   sum-=node1.value+carry
80   sum+=sum%10
81   list3.AddDigit(sum)
82   node1=node1.next
83   node2=node2.next
84   carry=sum/10
85   sum-=node1.value+carry
86   sum+=sum%10
87   list3.AddDigit(sum)
88   node1=node1.next
89   node2=node2.next
90   carry=0
91
92 #Print the resulting list
93 puts "Sum list"

```

**Code Block 2: Sublime Text Editor showing the second version of the program (addTwoNumbers.rb) for adding two numbers represented as linked lists.**

```

1  require_relative './linkedlist.rb'
2
3  puts "Enter two numbers non-negatives"
4  #To operate numbers we must transform the
5  #character string to integer values
6
7  x = gets.chomp.to_i
8  y = gets.chomp.to_i
9
10 #Set the longest and shortest size of both lists
11 if list1.size > list2.size
12   biggerSize = list1.size
13   smallerSize = list2.size
14 else
15   biggerSize = list2.size
16   smallerSize = list1.size
17 end
18
19 #Define the variables "carry" and "sum", and the
20 #values of the nodes are added in order, taking into
21 #account that one list may be longer than the other
22 carry=0
23
24 list3=LinkedList.new
25 list3.head=nil
26 node1=list1.head
27 node2=list2.head
28
29 while node1!=nil || node2!=nil
30   if node1==nil
31     sum=(node2.value+carry)
32   elsif node2==nil
33     sum=(node1.value+carry)
34   else
35     sum=(node1.value+node2.value)+carry
36   end
37   sum+=sum%10
38   list3.AddDigit(sum)
39   node1=node1.next
40   node2=node2.next
41   carry=sum/10
42   sum-=node1.value+carry
43   sum+=sum%10
44   list3.AddDigit(sum)
45   node1=node1.next
46   node2=node2.next
47   carry=0
48
49 #Print the resulting list
50 puts "Sum list"

```

The screenshot shows a Linux desktop environment with a terminal window open in a window manager. The terminal window has tabs for 'readmeAndEditFile' and 'addTwoNumbers.py'. The code in the terminal is as follows:

```
91 #Print the resulting list
92
93 puts "Sum list"
94
95 lists.Print
96
97 #Transform the list to an integer
98
99 number3 = 0
100 current = list.head
101 cmt2 = 0
102 while cmt2 < list.size
103     number3 += current.value*(10**cmt2)
104     current = current.next
105     cmt2+=1
106
107 #Print the operation results
108
109 puts "The sum of #{x} and #{y} is #{number3}"
```

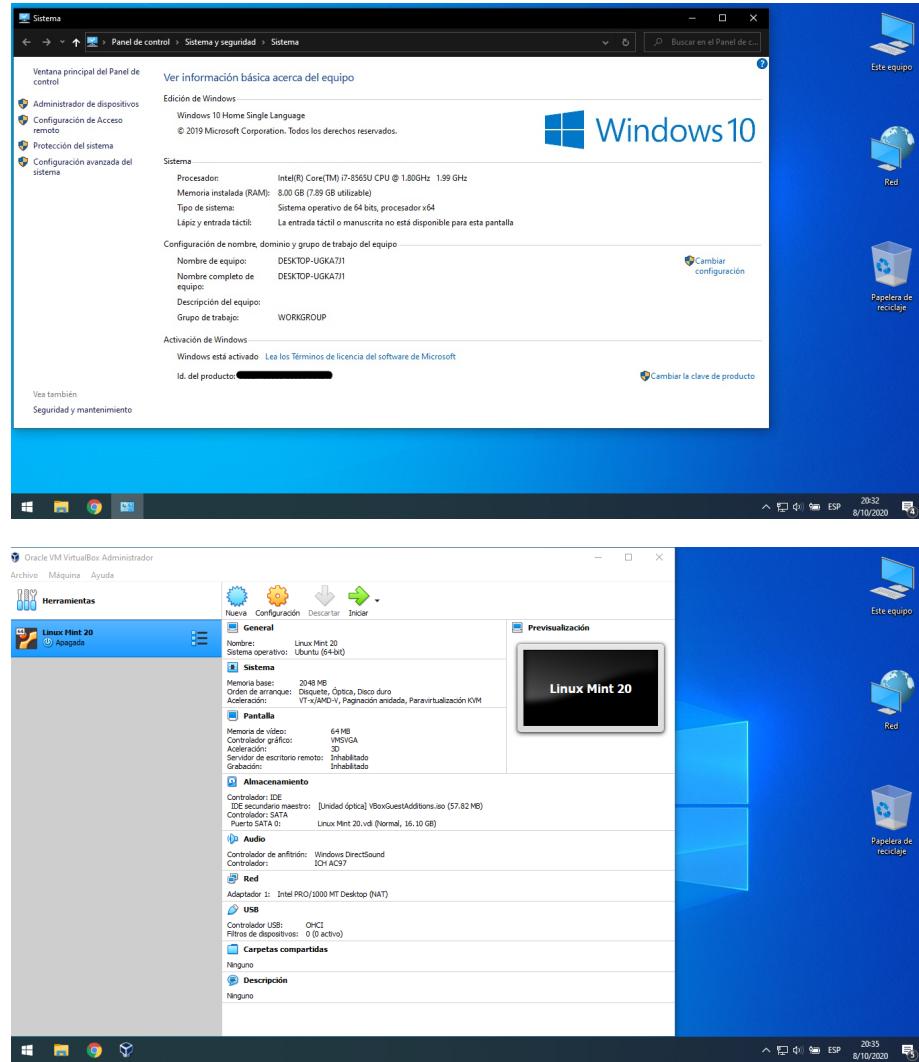
The terminal window also displays status information at the bottom: Line 111, Column 47, Tab Stop: 4, and a file icon.

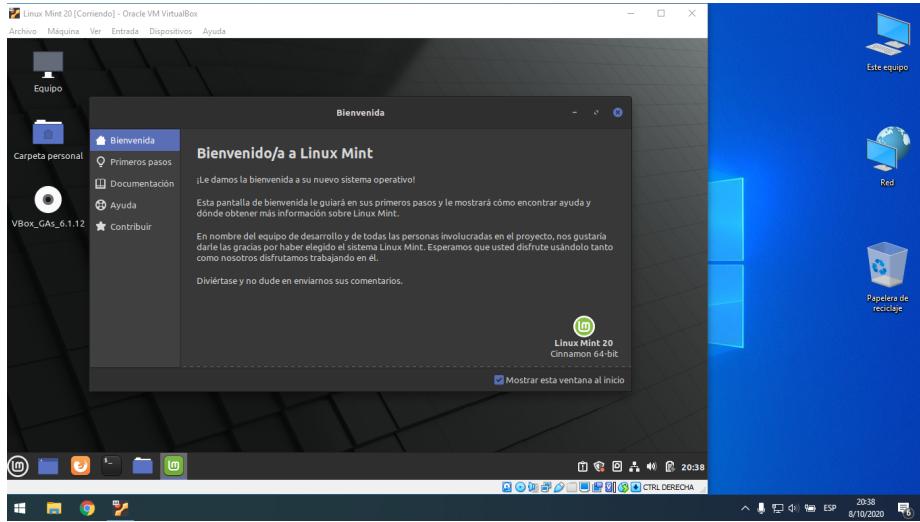
The following picture is from the program execution

```
ellizabetht@Elizabeth-VirtualBox:~/Escritorio$ ruby addTwoNumbers.rb
Enter two numbers non-negatives
> 123
> 456
123 + 456 = 589
```

### 3 Saras Rivera, André Edgardo

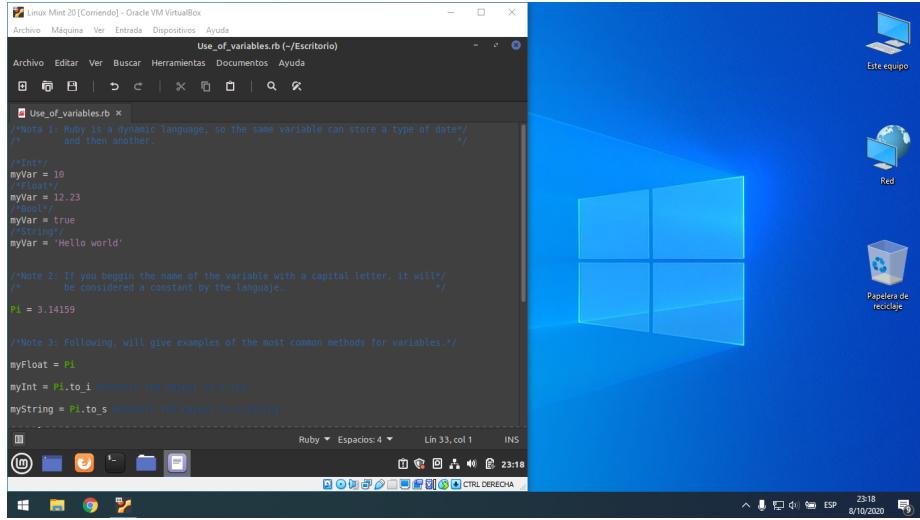
#### 3.1 Resources and Oracle VM Virtual Box parameters





### 3.2 Language research: Ruby Basic Syntax and Commands

First of all, it should be clarified Ruby is a dynamic language where all variables including constants are treated as objects, that is to say, operations performed at compilation time occur at runtime, it is not necessary to declare variables either. In this context, arithmetic, comparison and ternary operators will be considered methods. On the following pages, will be explained using examples below:



The image shows two screenshots of a Linux Mint 20 terminal window running on Oracle VM VirtualBox. The top screenshot displays the content of a file named `Use_of_variables.rb`. The code demonstrates various methods for working with variables in Ruby, such as `to_i`, `to_s`, `even?`, `abs`, and `upcase`. It also includes notes about object assignment and identity. The bottom screenshot shows the content of a file named `Output_Input_dates.rb`. This code illustrates how to use `gets` to read input from the terminal and `puts` to print output, including examples of string concatenation and interpolation.

```

Linux Mint 20 [Corriendo] - Oracle VM VirtualBox
Archivo Maquina Ver Entrada Dispositivos Ayuda
Use_of_variables.rb (~/Escritorio)
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Use_of_variables.rb x
r= -3.141592653589793
/*Note 3: Following, will give examples of the most common methods for variables.*/
myFloat = PI
myInt = PI.to_i #Convert the object in a int
myString = PI.to_s #Convert the object in a string
myBool = PI.even? #returns True if the object is a even number.
myPositive = -10.abs #returns the absolute value
myTitle = 'capital letter'.upcase #Capitalize all letters of the string
/*Note 4: In Ruby all are considered an object, therefore to differentiate them it assigns*/
/* an ID, which we can know with the method below.*/
puts myVar.object_id

Ruby Espacios: 4 Lin 33, col 1 INS
Windows G Suite Google Photos Google Sheets Google Slides CTRL DERECHA 23:18 8/10/2020

Linux Mint 20 [Corriendo] - Oracle VM VirtualBox
Archivo Maquina Ver Entrada Dispositivos Ayuda
Output_Input_dates.rb (~/Escritorio)
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Output_Input_dates.rb x
/*Note 1: In Ruby the standard data input is "gets", this function returns a string read from the terminal.*/
myName = gets
myName = gets.chomp #The 'chomp' method delete the carriage return characters at the end of the string
/*Note 1: Ruby has two standard data outputs.*/
print myName #Prints without adding the line change at the end.
puts myName #Prints adding a line break to the end of the string.
puts 'Hi, ' + myName + ' nice to meet you' #Print using concatenation
puts "Hi, #{myName} nice to meet you" #Print using interpolation (to use this way of printing, we have to put double quotes)
Ruby Espacios: 4 Lin 16, col 19 INS
Windows G Suite Google Photos Google Sheets Google Slides CTRL DERECHA 23:48 8/10/2020

```

Linked structures such as lists, stacks and queues are not created by default in Ruby, however, the arrays defined in this language are dynamic (the size can be varied). In that context, we can implement linked structures through classes and create methods to handle them as seen in the introductory example. The concepts of what a linked structure is already found in the Data Structures and Algorithms Using Python book provided by the teacher in the course bibliography, for that reason, it will not go into much depth on this topic.

### 3.3 Example problem: Traffic Recreation

The use of the concepts mentioned above will be explained by means of a simple and didactic example, in this case the traffic simulation using a queue.

A screenshot of a Linux Mint 20 desktop environment. On the left, there is a terminal window titled 'Traffic on Marina avenue.rb (~/Escritorio)' containing Ruby code. The code defines a class 'Cars\_in\_traffic' with methods for initializing a head car, adding new cars, and printing the traffic. It also defines a class 'New\_car' with a 'next' attribute. On the right, there is a desktop interface with icons for 'Este equipo', 'Red', and 'Papelera de reciclaje'. The taskbar at the bottom shows various application icons.

```

class Cars_in_traffic
  attr_accessor :car_ahead, :car_back
  def initialize()
    @car_ahead = @car_back = nil
  end

  def new_car arrives(plate)
    new_car = New_car.new(plate)
    @car_ahead = new_car if @car_ahead.nil?
    @car_back.next = new_car unless @car_back.nil?
    @car_back = new_car
  end

  def print_cars_in_traffic()
    current = @car_ahead
    while current
      print current.car_plate.to_s + ' - '
      current = current.next
    end
    puts 'How much traffic!'
  end
end

```

A screenshot of a Linux Mint 20 desktop environment. On the left, there is a terminal window titled 'Traffic on Marina avenue.rb (~/Escritorio)' containing Ruby code. The code defines a class 'Cars\_in\_traffic' with methods for printing traffic and adding new cars. It also defines a class 'New\_car' with a 'next' attribute. On the right, there is a desktop interface with icons for 'Este equipo', 'Red', and 'Papelera de reciclaje'. The taskbar at the bottom shows various application icons.

```

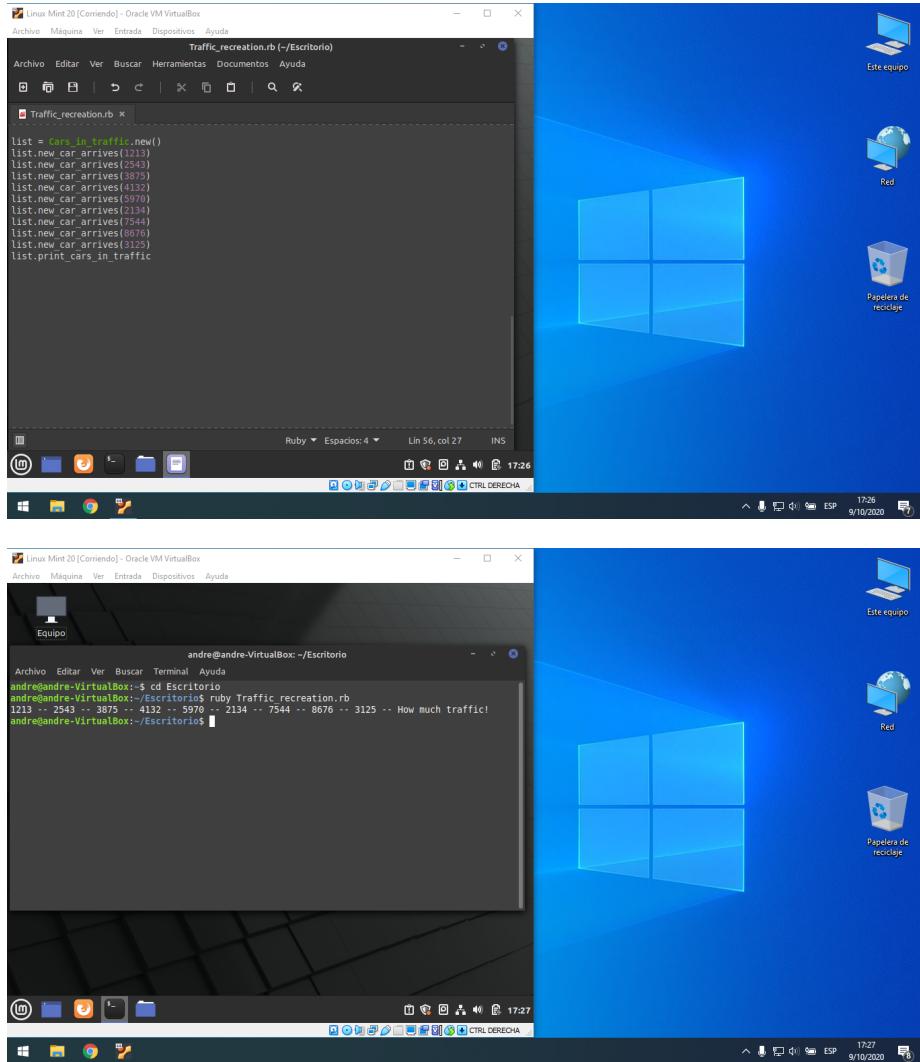
class Cars_in_traffic
  attr_accessor :car_ahead, :car_back
  def initialize()
    @car_ahead = @car_back = nil
  end

  def new_car arrives(plate)
    new_car = New_car.new(plate)
    @car_ahead = new_car if @car_ahead.nil?
    @car_back.next = new_car unless @car_back.nil?
    @car_back = new_car
  end

  def print_cars_in_traffic()
    current = @car_ahead
    while current
      print current.car_plate.to_s + ' - '
      current = current.next
    end
    puts 'How much traffic!'
  end
end

```

First of all, we define our class Cars in traffic which simulates a group of cars stalled in traffic and we can differentiate them through their Plate. Inside our first defined class, we will mainly create three methods: The first one, initializes the variables defined by the access method attr accessor to null value. The second, adds one more car to traffic, identifying it by means of its license plate. It also checks if the queue is empty (in null value) and if that were the case, it increases the cart to said empty queue. The third, prints all the cars in traffic by means of a 'while' that ends when the head reaches the null value. Within our second defined class, we will only create the node New car by initializing it and defining its variables using attr accessor and def.



The image shows two screenshots of a Linux Mint 20 desktop environment within Oracle VM VirtualBox. Both screenshots feature a blue Windows-style desktop background with icons for 'Este equipo', 'Red', and 'Papelera de reciclaje'.

**Screenshot 1 (Top):** A terminal window titled 'Traffic\_recreation.rb (-/Escritorio)' is open, displaying Ruby code. The code defines a class 'Cars\_in\_traffic' with a 'new()' method that adds car arrival times to a list. It then prints the list. The terminal window has a dark theme and includes status bars at the bottom showing 'Ruby' and '17:26'.

```

list = Cars_in_traffic.new()
list.new_car_arrives(1213)
list.new_car_arrives(2134)
list.new_car_arrives(3452)
list.new_car_arrives(4875)
list.new_car_arrives(4132)
list.new_car_arrives(5970)
list.new_car_arrives(2134)
list.new_car_arrives(2134)
list.new_car_arrives(3452)
list.new_car_arrives(6576)
list.new_car_arrives(3125)
list.print_cars_in_traffic

```

**Screenshot 2 (Bottom):** A terminal window titled 'andrei@andre-VirtualBox: ~/Escritorio' is open, showing the output of the Ruby script. The output lists various car arrival times followed by the message 'How much traffic!'. The terminal window has a dark theme and includes status bars at the bottom showing '17:27' and 'andre@andre-VirtualBox:~/Escritorio\$'.

```

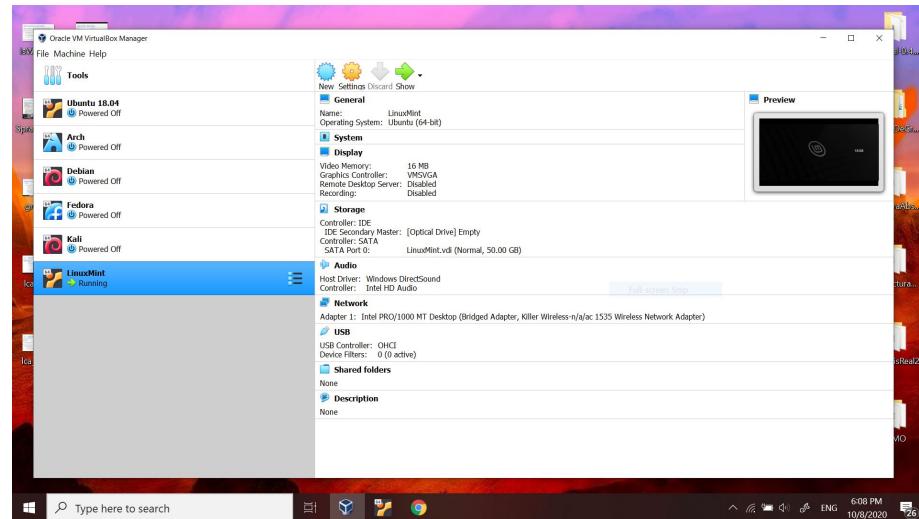
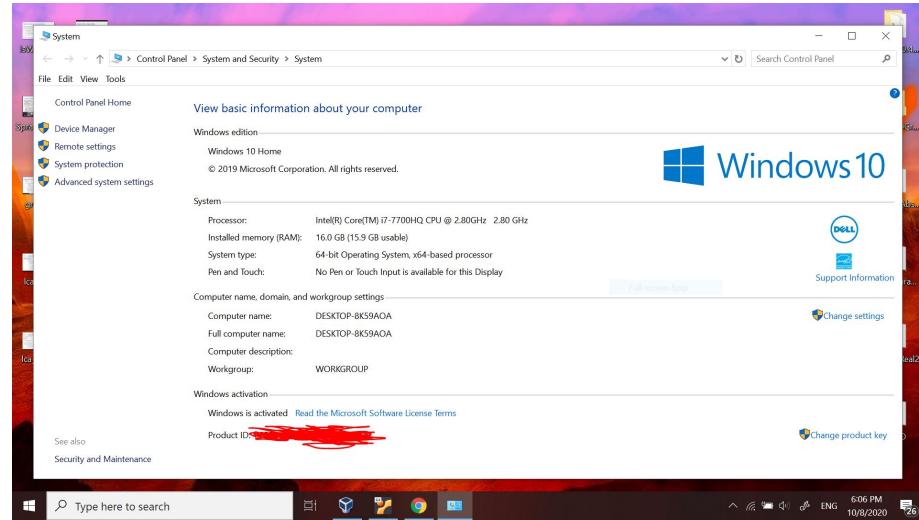
andrei@andre-VirtualBox:~/Escritorio$ ruby Traffic_recreation.rb
1213 -- 2543 -- 3875 -- 4132 -- 5970 -- 2134 -- 7544 -- 8676 -- 3125 -- How much traffic!
andrei@andre-VirtualBox:~/Escritorio$ 

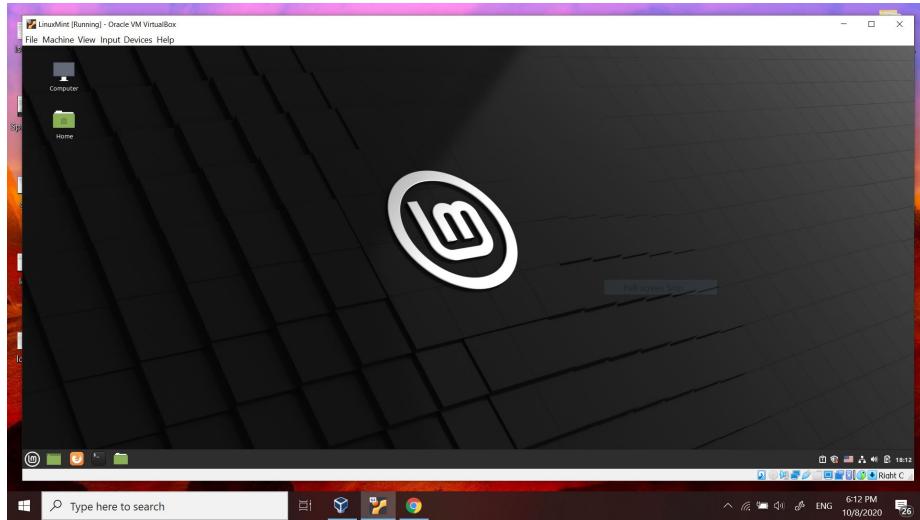
```

In the example shown, first the class 'Cars in Traffic' is instantiated in a variable named for this case 'list' (list of cars stuck in traffic), consequently we proceed to increase values to the queue one by one to finally print all cars.

## 4 Loaiza Vasquez, Manuel Alejandro

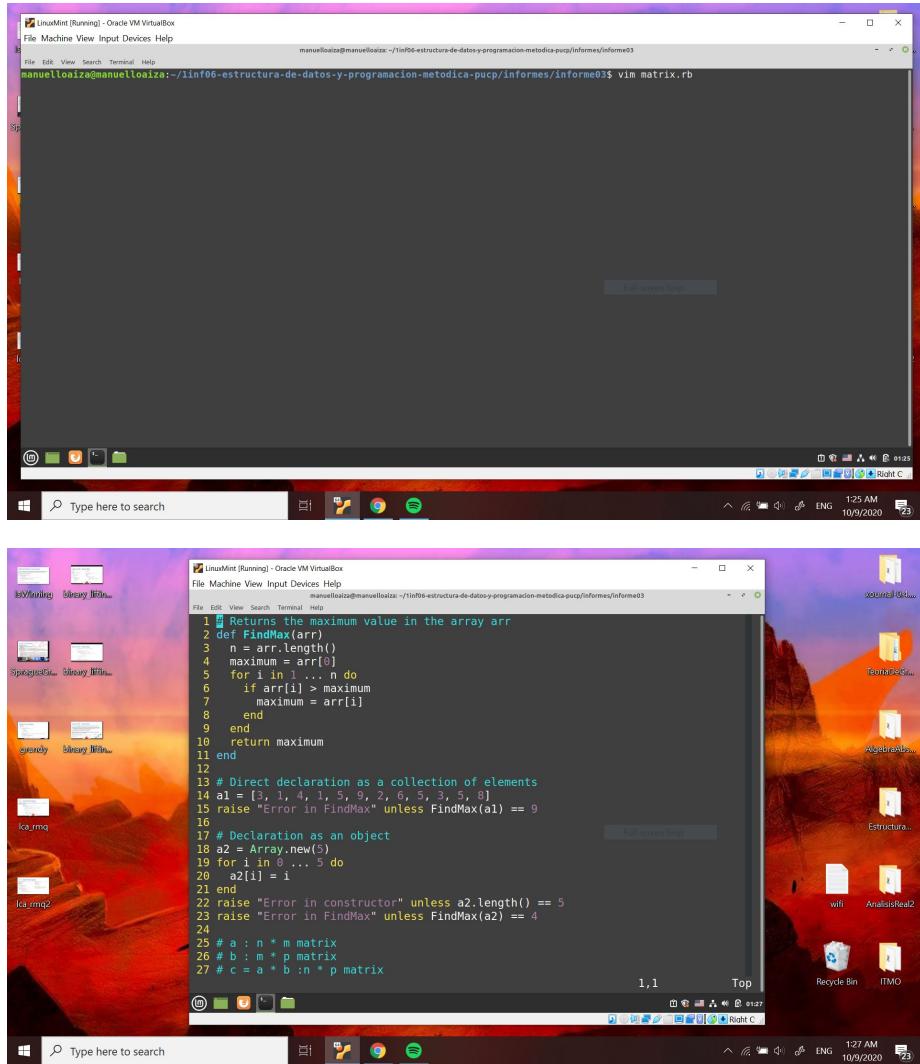
### 4.1 Resources and Oracle VM Virtual Box parameters





## 4.2 Language research: Arrays, strings, exceptions and files

For the adjacency matrix representation of a graph we need a 2D array and for the adjacency list representation of a graph we need a one dimensional array, so it was very important to do some research about them. Arrays in Ruby are instances of a class called `Array` which has methods such as `length`, many constructors that can receive one, many or no elements and we can even give the constructor the number of elements and which elements are going to be there by default. We can see how arrays work in my example on section 4.3 in which I have used an array of lists to iterate over them to obtain the sorted list and other arrays to store information about positions and length to decrease the time complexity of the algorithm. In the following example, I am going to declare arrays in different ways, iterate through them via indices with a `for` loop, and also creating arrays of arrays, also known as matrices, which I am going to print and multiply using some functions that I will define.



Linux Mint [Running] - Oracle VM VirtualBox

```

File Machine View Input Devices Help
manuelvillalba@manuelvillalba: ~/TIN06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03

25 # a : n * matrix
26 # b : m * p matrix
27 # c = a * b : n * p matrix
28 def MatrixMultiplication(a, b)
29   n = a.length()
30   m = a[0].length()
31   p = b[0].length()
32   raise "Input matrices can't be multiplied" unless m == b.length()
33   # Creating the empty matrix that is going to store our result
34   c = Array.new(n) { Array.new(p) }
35   for i in 0 ... n do
36     c[i] = Array.new(p)
37   end
38   for i in 0 ... n do
39     for j in 0 ... p do
40       c[i][j] = 0
41       for k in 0 ... m do
42         c[i][j] += a[i][k] * b[k][j]
43       end
44     end
45   end
46   return c
47 end
48
49 def PrintMatrix(a)
50   n = a.length()
51   m = a[0].length()

```

51,1 55%

Full-screen Setup

Linux Mint [Running] - Oracle VM VirtualBox

```

File Machine View Input Devices Help
manuelvillalba@manuelvillalba: ~/TIN06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03

44   end
45   end
46   return c
47 end
48
49 def PrintMatrix(a)
50   n = a.length()
51   m = a[0].length()
52   for i in 0 ... n do
53     for j in 0 ... m do
54       print "#(a[i][j]) "
55     end
56     puts ""
57   end
58 end
59
60 # Declaring the matrices that are going to be multiplied
61 a = [[2, 1, 4], [0, 1, 1]]
62 b = [[6, 3, -1, 0], [1, 1, 0, 4], [-2, 5, 0, 2]]
63 c = MatrixMultiplication(a, b)
64
65 puts "Matrix a:"
66 PrintMatrix(a)
67 puts "Matrix b:"
68 PrintMatrix(b)
69 puts "Matrix c = a * b:"
70 PrintMatrix(c)

```

78,1 Bot

Type here to search

Recycle Bin ITMO

wifi AnalisisReal2

Estructura...

AlgebraLinea...

TareaDeG...

zoomed 0.d...

Linux Mint [Running] - Oracle VM VirtualBox

```

File Machine View Input Devices Help
manuelvillalba@manuelvillalba: ~/TIN06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03

44   end
45   end
46   return c
47 end
48
49 def PrintMatrix(a)
50   n = a.length()
51   m = a[0].length()
52   for i in 0 ... n do
53     for j in 0 ... m do
54       print "#(a[i][j]) "
55     end
56     puts ""
57   end
58 end
59
60 # Declaring the matrices that are going to be multiplied
61 a = [[2, 1, 4], [0, 1, 1]]
62 b = [[6, 3, -1, 0], [1, 1, 0, 4], [-2, 5, 0, 2]]
63 c = MatrixMultiplication(a, b)
64
65 puts "Matrix a:"
66 PrintMatrix(a)
67 puts "Matrix b:"
68 PrintMatrix(b)
69 puts "Matrix c = a * b:"
70 PrintMatrix(c)

```

78,1 Bot

Type here to search

Recycle Bin ITMO

wifi AnalisisReal2

Estructura...

AlgebraLinea...

TareaDeG...

zoomed 0.d...

```

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03
informes/informe03$ vim matrix.rb
manuelloaiza@manuelloaiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ ruby matrix.rb
Matrix a:
2 1 4
0 1 1
Matrix b:
6 3 -1 0
1 1 0 4
-2 5 0 2
Matrix c = a * b:
5 27 -2 12
-1 6 0 6
manuelloaiza@manuelloaiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ Full-screen Snap

```

In Ruby, we have the class **String**, a collection of elements of one byte, they usually represent the characters from 0 to 255. In C, it was very easy to do arithmetic operations with them. In Ruby, we can use the methods **ord** and **chr** to obtain the ASCII code of a character and to convert the ASCII code to the character, respectively. Then, as other containers, we can iterate through them via indeces, concatenate strings using + operator, obtain their size using **length**, parse them and much more. In the following program, I am going to test that using some functions to check if a string is palindrome, and converting lowercase characters into uppercase because comparison is case sensitive.

```

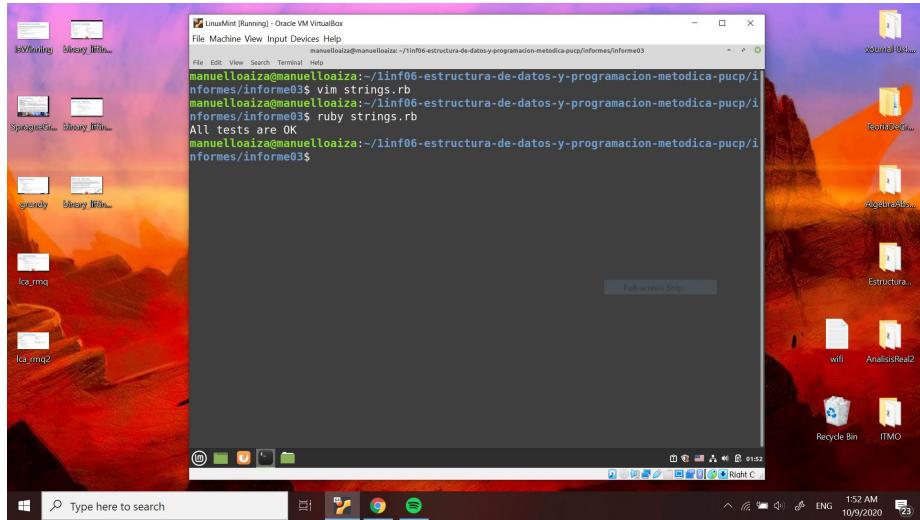
LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03
informes/informe03$ vim strings.rb
manuelloaiza@manuelloaiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ Full-screen Snap

```

```

1 # Recursive functions that returns if a string is palindrome
2 # Time Complexity: O(|s|)
3 def IsPalindrome(s, l, r)
4     if l > r
5         return true
6     end
7     if s[l] == s[r]
8         return IsPalindrome(s, l + 1, r - 1)
9     else
10        return false
11    end
12 end
13
14 # Lowercase characters are changed into uppercase and the original string
15 # is modified because they are passed by reference
16 def ToUpper(s)
17     n = s.length()
18     for i in 0 ... n do
19         if 97 <= s[i].ord && s[i].ord <= 122
20             s[i] = (s[i].ord - 32).chr
21         end
22     end
23 end
24
25 # Direct string declaration
26 name1 = "Manuel"
27 name2 = "Alejandro"
28 name3 = "Ana"
29
30 raise "Error in IsPalindrome" unless !IsPalindrome(name1, 0, name1.length() - 1)
31 raise "Error in IsPalindrome" unless !IsPalindrome(name2, 0, name2.length() - 1)
32 raise "Error in IsPalindrome" unless !IsPalindrome(name3, 0, name3.length() - 1)
33 # Strings can be assigned as common variables
34 name4 = name3
35 # Eventhough "Ana" is not a palindrome
36 raise "Error in IsPalindrome" unless !IsPalindrome(name4, 0, name4.length() - 1)
37 ToUpper(name4)
38 # Now "ANA" is a palindrome
39 raise "Error in IsPalindrome" unless IsPalindrome(name4, 0, name4.length() - 1)
40 puts "All tests are OK"

```



To handle exceptions, we can use `raise ... unless ...` as we use `assert` in C or Python. We can find other ways to use assertions by creating modules and including them in our class definition. I have used `raise` in our testing code on section 1.2 to check if our double linked list methods are correct and in the previous examples with arrays and strings.

When we want to load files from other sources, we can use `require` and write the string that with the path of the file we want to use. If that file is in the same directory of our current file, we can use `require_relative`. In my example below and in the testing, I have used this statement to load our double linked list class definition from `DoubleLinkedList.rb` file.

### 4.3 Example problem: Merged k Sorted Lists

You are given an array of  $k$  double linked lists, each double linked list is sorted in non decreasing order. Merge all the double linked lists into one double linked list and return it.

```

manuel@manuel-OptiPlex-5090:~/Desktop$ ./DoubleLinkedList.rb
1 #require_relative "./DoubleLinkedList.rb"
2
3 # First, let's create the input of no decreasing ordered lists
4 list1 = DoubleLinkedList.new
5 list1.AddRight 1
6 list1.AddRight 4
7 list1.AddRight 5
8
9 list2 = DoubleLinkedList.new
10 list2.AddRight 1
11 list2.AddRight 3
12 list2.AddRight 4
13
14 list3 = DoubleLinkedList.new
15 list3.AddRight 2
16 list3.AddRight 6
17
18 puts "Input"
19 list1.Print
20 list2.Print
21 list3.Print
22
23 # INF is going to represent a number bigger than all our numbers
24 # lists is going to store our lists
25 # number_lists is going to represent k, the number of lists we want to merge
26 INF = 2 ** 63 - 1

```

```

manuel@manuel-OptiPlex-5090:~/Desktop$ ./DoubleLinkedList.rb
27 lists = [list1, list2, list3]
28 number_lists = lists.length()
29
30 # result is going to store our final ordered double linked list
31 # number_elements is going to have the number of elements in the result
32 result = DoubleLinkedList.new
33 number_elements = 0
34
35 # list_size is going to store the length of each list
36 # current_pos is going to store the index in which we are at on each list
37 list_length = Array.new(number_lists)
38 current_pos = Array.new(number_lists)
39
40 # We are storing the length of each list to query them in O(1) instead of O(n)
41 for i in 0 ... number_lists do
42   list_length[i] = lists[i].Size
43   current_pos[i] = 0
44   number_elements += list_length[i]
45 end
46
47 for i in 0 ... number_elements do
48   # minimum is going to store two values, the minimum number we have until now

```

```

manuellloiza@manuellloiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ ruby MergeKSortedDoubleLinkedLists.rb
Input:
1 <-> 4 <-> 5
1 <-> 3 <-> 4
2 <-> 6
Output:
1 <-> 1 <-> 2 <-> 3 <-> 4 <-> 4 <-> 5 <-> 6
manuellloiza@manuellloiza:~/linf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ 

```

## References

- [1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, Cambridge, Massachusetts, 2009.
- [2] Tutorialspoint: Ruby Basics,  
<https://www.tutorialspoint.com/ruby/index.htm>
- [3] Ruby-docs.org: IO,  
<https://ruby-doc.org/core-2.7.2/IO.html#method-i-puts>

- [4] Stack Overflow: What is the difference between print and puts?,  
<https://stackoverflow.com/questions/5018633/what-is-the-difference-between-print-and-puts>
- [5] Stack Overflow: Is it idiomatic Ruby to add an assert( ) method to Ruby's Kernel class?,  
<https://stackoverflow.com/questions/147969/is-it-idiomatic-ruby-to-add-an-assert-method-to-rubys-kernel-class>
- [6] Ruby-docs.org: Arrays,  
<https://ruby-doc.org/core-2.4.1/Array.html>
- [7] Ruby-docs.org: Strings,  
<https://ruby-doc.org/core-2.4.0/String.html>
- [8] LeetCode: Reverse Linked List,  
<https://leetcode.com/problems/reverse-linked-list/>
- [9] LeetCode: Add Two Numbers,  
<https://leetcode.com/problems/add-two-numbers/>
- [10] LeetCode: Merge k Sorted Lists,  
<https://leetcode.com/problems/merge-k-sorted-lists/>