Computer Science Programming Basics with Ruby examples

Example 1-1. Plain language → Ruby code

```
puts "Enter first name: "     # Ask for the first name
first_name = gets             # Store the first name
puts "Enter last name: "      # Ask for the last name
last_name = gets              # Store the last name
puts "Enter middle initial: " # Ask for the middle initial
middle_initial = gets         # Store the middle initial
```

Example 3-1. Code with comments

```
puts "What is the length?" # Ask for the length
length_input = gets        # Stores the length
puts "What is the width?"  # Ask for the width
width_input = gets         # Stores the width
length = length_input.to_i # Convert length to integer
width = width_input.to_i   # Convert width to integer
area = length * width      # Calculate rectangle area
puts area                  # Display area
```

Example 4-3. If statement

```
if (condition)
  # section 1
end
```

Example 4-4. Determine if the number is even

```
# if a number is even, print out "Even"
puts "Enter a number" # print message
number = gets.to_i    # get number as an integer
if (number % 2 == 0)  # check if even
  puts "Even"
end
```

Example 4-5. If-else statement

```
if (condition)
  # section 1
else
  # section 2
end
```

Example 4-6. Theater 1

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# Determine the cost based on age
if (age < 12)
  cost = 9
else
  cost = 18
end
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-7. Theater 2

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# Determine the cost based on age
if (age <= 12)
  cost = 9
else
  cost = 18
end
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-8. Elsif statements

```
if (condition1)
  # section 1
elsif (condition2)
  # section 2
elsif (condition3)
  # section 3
else
  # section 4
end
```

Example 4-9. Theater 3

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# Determine the cost based on age
if (age <= 12)
  cost = 9
elsif (age >= 65)
  cost = 12
else
  cost = 18
end
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-10. Advanced conditionals

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# Determine the cost based on age
if ((age <= 12) or (age >= 65))
  cost = 9
else
  cost = 18
end
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-11. Case statement

```
case
when (expression1)
  # section 1
when (expression2)
  # section 2
else
  # section 3
end
```

Example 4-12. Theater 4

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# Determine the cost based on age
case
when (age <= 12)
  cost = 9
when (age >= 65)
  cost = 12
else
  cost = 18
end
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-13. Debugging example 1

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# Determine the cost based on age
case
# '=' is assignment NOT equality test '=='
when (age = 12) then
  cost = 9
when (age >= 65) then
  cost = 12
else
  cost = 18
end
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-14. Debugging example 2

```
puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i
# DEBUG
puts age
# Determine the cost based on age
case
# '=' is assignment NOT equality test '=='
when (age = 12) then
  cost = 9
when (age >= 65) then
  cost = 12
else
  cost = 18
end
# DEBUG
puts age
# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 4-15. Debugging example 3

```
# Flag for debugging (change to false when finished debugging)
DEBUG_MODULE_1 = true

puts "Enter the customer's age: "
# Get an integer age value from the user
age = gets.to_i

# Determine the cost based on age
if DEBUG_MODULE_1
  puts age
end
case
# '=' is assignment NOT equality test '=='
when (age = 12) then
  cost = 9
when (age >= 65) then
  cost = 12
else
  cost = 18
end
if DEBUG_MODULE_1
  puts age
end

# Print out the final cost
puts "Ticket cost: " + cost.to_s
```

Example 5-1. While loop construct

```
while (condition)
  # statement 1
  # statement 2
  # ...
  # statement n
end
```

Example 5-2. Counting program

```
n = 5
i = 0
while (i <= n)
  puts i
  i = i + 1
end
```

Example 5-3. Until loop construct

```
until (condition)
  # statement 1
  # statement 2
  # ...
  # statement n
end
```

Example 5-4. For loop

```
for num in (0..5)
  puts num
end
```

Example 5-5. Nested loop construct

```
for i in (1..3)
  puts "Outer loop: i = " + i.to_s
  for k in (1..4)
    puts "Inner loop: k = " + k.to_s
  end
end
```

Example 5-6. Infinite loops

```
puts "Count from 0 to ? "
n = gets.to_i
i = 5
while (i > 0)
  i = i + 2
end
```

Example 5-7. Prime numbers

```
# Initialize our counter
i = 1
# i: [0, 100]
while (i <= 100)
  # Initialize prime flag
  prime_flag = true
  j = 2
  # Test divisibility of i from [0, i/2]
  while (j <= i / 2)
    # puts " i ==> " + i.to_s + " j ==> " + j.to_s
    if (i % j == 0)
      prime_flag = false
      # break
    end
    j = j + 1
  end
  # We found a prime!
  if prime_flag
    puts "Prime ==> " + i.to_s
  end
  # Increment the counter
  i += 1
end
```

Example 5-8. Code for Exercise 2

Explain what it does.

```
puts "Enter a number >= 0: "
n = gets.to_i
a = 1
while (n > 1)
  a = (n * (n - 1)) * a
  n = n - 2
end
puts a
```

Example 6-1. Initializing an array

```
arr = Array.new()
arr[0] = 73
arr[1] = 98
arr[2] = 86
arr[3] = 61
arr[4] = 96
```

Example 6-2. Initializing an array

```
arr = [73, 98, 86, 61, 96]
```

Example 6-3. Changing the value of an element

```
arr = [5,6]
arr[0] = arr[0] + 10
puts arr[0]
```

Example 6-4. Displaying array content

```
arr = [73, 98, 86, 61, 96]
index = 0
while (index < arr.size)
  puts arr[index]
  index = index + 1
end
```

Example 6-5. Find the max

```
# Initialize array and loop values
arr = [73, 98, 86, 61, 96]
index = 0
max = 0
# Loop over each element in arr
while (index < arr.size)
  if (arr[index] > max)
    # Update max
    max = arr[index]
  end
  index = index + 1
end

# Output calculated max
puts "Max ==> " + max.to_s
```

Example 6-6. Outputting multidimensional arrays

```
# Initialize array and loop values
arr = [
       [73, 98, 86, 61, 96],
       [60, 90, 96, 92, 77],
       [44, 50, 99, 65, 10]
      ]
row = 0
column = 0

# Loop over each row
while (row < arr.size)
  puts "Row: " + row.to_s
  # Loop over each column
  while (column < arr[row].size)
    # Print the item at position row x column
    puts arr[row][column]
    column = column + 1
   end
   # Reset column, advance row
   column = 0
   row = row + 1
 end
```

Example 6-7. Find the max, modified

```
# initialize the array and index/score variables
arr = [
       [73, 98, 86, 61, 96],
       [60, 90, 96, 92, 77],
       [44, 50, 99, 65, 10]
      ]
row = 0
column = 0
maxscore = 0
maxrow = 0

 # for each row
 while (row < arr.size)
   # for each column
   while (column < arr[row].size)
     # update score variables
     if (arr[row][column] > maxscore)
       maxrow = row
       maxscore = arr[row][column]
     end
     # increment column
     column = column + 1
   end
   # reset column, increment row
   column = 0
   row = row + 1
 end

# output name and high score information
if maxrow == 0
  puts "Geraldo has the highest score."
elsif maxrow == 1
  puts "Brittany has the highest score."
elsif maxrow == 2
  puts "Michael has the highest score."
else
  puts "Something didn't work correctly."
end
puts "The high score was: " + maxscore.to_s
```

Example 6-8. Example hash usage

```
scores = Hash.new
scores["Geraldo"] = [98, 95, 93, 96]
scores["Brittany"] = [74, 90, 84, 92]
scores["Michael"] = [72, 87, 68, 54, 10]
```

Example 6-9. Example hash accessor usage

```
scores = Hash.new
scores["Geraldo"] = [98, 95, 93, 96]
scores["Brittany"] = [74, 90, 84, 92]
scores["Michael"] = [72, 87, 68, 54, 10]
name = "Brittany"
puts name + " first score is: " + scores[name][0].to_s
```

Example 6-10. Find the max—hash

```
scores = Hash.new
scores["Geraldo"] = [98, 95, 93, 96]
scores["Brittany"] = [74, 90, 84, 92]
scores["Michael"] = [72, 87, 68, 54, 10]
maxscore = 0

for name in scores.keys
  column = 0
  while (column < scores[name].size)
    if (scores[name][column] > maxscore)
      maxname = name
      maxscore = scores[name][column]
    end
    column = column + 1
  end
end

puts maxname + " has the highest score."
puts "The highest score is: " + maxscore.to_s
```

Example 6-11. Code for Exercise 1

What is the output given a[0] = 9, a[1] = 2, a[2] = 5, a[3] = 4, a[4]= 3

```
i = 0

while (i < a.size)
  puts a[i]
  i = i + 1
end
```

Example 6-12. Code for Exercise 2 Fix the sort code

```
arr = [5, 22, 29, 39, 19, 51, 78, 96, 84]
i = 0
while (i < arr.size - 1 and arr[i] < arr[i + 1])
  i = i + 1
end
puts i

arr[i] = arr[i + 1]
arr[i + 1] = arr[i]
```

Example 7-1. Code for selection sort

```
# Code for selection sort
# 35 students in our class
NUM_STUDENTS = 35
# Max grade of 100%
MAX_GRADE = 100
num_compare = 0
arr = Array.new(NUM_STUDENTS)

# Randomly populate arr
for i in (0..NUM_STUDENTS - 1)
  # Maximum possible grade is 100%, keep in mind that rand(5) returns
  # possible values 0-4, so
  # we must add 1 to MAX_GRADE
  arr[i] = rand(MAX_GRADE + 1)
end

# Output current values of arr
puts "Input list:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end

# Now let's use a selection sort. We first find the lowest number in the
# array and then we move it to the beginning of the list
for i in (0..NUM_STUDENTS - 2)
  min_pos = i
  for j in (i + 1)..(NUM_STUDENTS - 1)
    num_compare = num_compare + 1
    if (arr[j] < arr[min_pos])
      min_pos = j
    end
  end
  # Knowing the min, swap with current first element (at position i)
  temp = arr[i]
  arr[i] = arr[min_pos]
  arr[min_pos] = temp
end

# Now output the sorted array
puts "Sorted list:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end

puts "Number of Comparisons ==> " + num_compare.to_s
```

Example 7-2. Code for insertion sort

```
# Code for insertion sort
# Declare useful constants
NUM_STUDENTS = 35
MAX_GRADE = 100
num_compare = 0
arr = Array.new(NUM_STUDENTS)

# Randomly populate arr

for i in (0..NUM_STUDENTS - 1)
  arr[i] = rand(MAX_GRADE + 1)
end

# Output randomly generated array
puts "Input array:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end

# Now let's use an insertion sort
# Insert lowest number in the array at the right place in the array
for i in (0..NUM_STUDENTS - 1)
  # Now start at current bottom and move toward arr[i]
  j = i
  done = false
  while ((j > 0) and (! done))
    num_compare = num_compare + 1
    # If the bottom value is lower than values above it, swap it until it
    # lands in a place where it is not lower than the next item above it
    if (arr[j] < arr[j - 1])
      temp = arr[j - 1]
      arr[j - 1] = arr[j]
      arr[j] = temp
    else
      done = true
    end
    j = j - 1
  end
end

# Now output the sorted array
puts "Sorted array:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end
puts "Number of Comparisons ==> " + num_compare.to_s
```

Example 7-3. Code for bubble sort

```
# Code for bubble sort
NUM_STUDENTS = 35
# Max grade of 100%
MAX_GRADE = 100
num_compare = 0
arr = Array.new(NUM_STUDENTS)

# Randomly put some final exam grades into arr

for i in (0..NUM_STUDENTS - 1)
  arr[i] = rand(MAX_GRADE + 1)
end

# Output randomly generated array
puts "Input array:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end

# Now let's use bubble sort. Swap pairs iteratively as we loop through the
# array from the beginning of the array to the second-to-last value
for i in (0..NUM_STUDENTS - 2)
  # From arr[i + 1] to the end of the array
  for j in ((i + 1)..NUM_STUDENTS - 1)
    num_compare = num_compare + 1
    # If the first value is greater than the second value, swap them
    if (arr[i] > arr[j])
      temp = arr[j]
      arr[j] = arr[i]
      arr[i] = temp
    end
  end
end

# Now output the sorted array
puts "Sorted array:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end
puts "Number of Comparisons ==> " + num_compare.to_s
```

Example 7-4. Code for radix sort

```
# Code for radix sort
NUM_STUDENTS = 35
MAX_GRADE = 100
arr = Array.new(NUM_STUDENTS)

# Randomly put some grades into the array *as strings*
for i in (0..NUM_STUDENTS - 1)
  arr[i] = rand(MAX_GRADE + 1).to_s
end

# Output array and find the maximum number of digits in the generated array
puts "Input array: "
max_length = 0
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i]
  if arr[i].length > max_length
    max_length = arr[i].length
  end
end
puts "Max length ==> " + max_length.to_s

# Add 0 padding based on the max length, simplifying the sort algorithm
for i in (0..NUM_STUDENTS - 1)
  arr[i] = arr[i].rjust(max_length, "0")
end

# Now let's use a radix sort. Go through each digit and
# add each element to an array corresponding to the digits.
for i in (0..max_length - 1)
  # Clear out and reset the bucket
  buckets = Hash.new()
  for j in 0..9
    buckets[j.to_s] = Array.new()
  end

  # Add each number to its respective digit bucket
  for j in 0..NUM_STUDENTS - 1
    num = arr[j]
    digit = num[max_length - 1 - i]
    buckets[digit].push(num)
  end
  # Flatten the buckets into a one-dimensional array
  arr = buckets.values.flatten
end

# Now output the sorted array
puts "Sorted array:"
for i in (0..NUM_STUDENTS - 1)
```

```
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end
```

Example 7-5. Code for linear search

```
# Code for linear search
NUM_STUDENTS = 35
MAX_GRADE = 100
arr = Array.new(NUM_STUDENTS)
value_to_find = 8
i = 1
found = false

# Randomly put some student grades into arr
for i in (0..NUM_STUDENTS - 1)
  arr[i] = rand(MAX_GRADE + 1)
end

puts "Input List:"
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end
i = 0
# Loop over the list until it ends or we have found our value
while ((i < NUM_STUDENTS) and (not found))
  # We found it :)
  if (arr[i] == value_to_find)
    puts "Found " + value_to_find.to_s + " at position " + i.to_s + " of the
list."
    found = true
  end
  i = i + 1
end

# If we haven't found the value at this point, it doesn't exist in our list
if (not found)
  puts "There is no " + value_to_find.to_s + " in the list."
end
```

Example 7-6. Code for binary search

```
# Code for binary search
NUM_STUDENTS = 30
MAX_GRADE = 100
arr = Array.new(NUM_STUDENTS)
# The value we are looking for
value_to_find = 7
low = 0
high = NUM_STUDENTS - 1
middle = (low + high) / 2
found = false

# Randomly put some exam grades into the array
for i in (0..NUM_STUDENTS - 1)
  new_value = rand(MAX_GRADE + 1)
  # make sure the new value is unique
  while (arr.include?(new_value))
    new_value = rand(MAX_GRADE + 1)
  end
  arr[i] = new_value
end
# Sort the array (with Ruby's built-in sort)
arr.sort!

print "Input List: "
for i in (0..NUM_STUDENTS - 1)
  puts "arr[" + i.to_s + "] ==> " + arr[i].to_s
end

while ((low <= high) and (not found))
  middle = (low + high) / 2
  # We found it :)
  if arr[middle] == value_to_find
    puts "Found grade " + value_to_find.to_s + "% at position " + middle.to_s
    found = true
  end

  # If the value should be lower than middle, search the lower half,
  # otherwise, search the upper half
  if (arr[middle] < value_to_find)
    low = middle + 1
  else
    high = middle - 1
  end
end

if (not found)
  puts "There is no grade of " + value_to_find.to_s + "% in the list."
end
```