Programowanie w Ruby - część I i II

ćwiczenia

Choinka

Napisz program rysujący tekstową choinkę w konsoli. Na przykład taką:

```
    *
   ***
  *****
 *******
   ***
  *****
 *******
*********
***********
    *
```

Metody, które mogą Ci się przydać to:

- print - podobnie jak puts wypisuje na ekran podany obiekt, ale bez dodatkowego znaku nowej lini
- Integer#times - pętla po liczbach z zakresu 0..n-1
- Numeric#step - pętla z zadanym skokiem

Postaraj się napisać program tak by choinka była chociaż częściowo generowana (a więc by program nie zawierał samych wywołań typu `puts " *** "`).

Ciąg Fibonnaciego

Ciąg Fibonnaciego to taki ciąg w którym dwa pierwsze elementy mają wartość odpowiednio 0 i 1 a każdy inny element jest sumą dwóch poprzednich.
Zatem kilka pierwszych elementów tego ciągu to: 0, 1, 1, 2, 3, 5, 8, 13.

Napisz program, który wypisze N pierwszych elementów tego ciągu, gdzie N jest parametrem przekazanym do tego programu.

Metody i konstrukcje, które mogą Ci się przydać to:

- Przypisanie równoległe w postaci a, b = b*2, a*2

Przykład wywołania:

```
$ ruby fib.rb 10
0
1
1
2
3
5
8
13
21
34
```

Znajdź indeks

Oto szkielet programu.

```
def find_index(array, value)
  # ...
end

find_index([1, 2, 3, 4, 5], 3)
find_index([5, 8, 6, 2, 2, 10], 2)
find_index([10, 20, 30], 100)
find_index([], 0)
```

Twoim zadaniem jest zaimplementowanie metody `find_index` tak aby wypisywała indeks (liczbę), pod którym znajduje się wartość `value` w tablicy `array`. Jeśli elementu nie ma w tablicy metoda powinna wypisać “not found”. Jeśli elementów o danej wartości jest więcej niż jeden funkcja powinna wypisać indeks pierwszego z nich.

```
$ ruby index.rb
2
3
not found
not found
```

Postaraj się zaimplementować najpierw tę metodę używając zwykłej pętli. Kiedy Ci się to uda zrób to jeszcze raz, ale z pomocą metody Array#index.

Tabliczka mnożenia

Napisz program, który wyświetli tabliczkę mnożenia 10x10.

```
$ ruby multiplication.rb
     1    2    3    4    5    6    7    8    9    10
     ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
1  | 1    2    3    4    5    6    7    8    9    10
2  | 2    4    6    8    10   12   14   16   18   20
3  | 3    6    9    12   15   18   21   24   27   30
4  | 4    8    12   16   20   24   28   32   36   40
5  | 5    10   15   20   25   30   35   40   45   50
6  | 6    12   18   24   30   36   42   48   54   60
7  | 7    14   21   28   35   42   49   56   63   70
8  | 8    16   24   32   40   48   56   64   72   80
9  | 9    18   27   36   45   54   63   72   81   90
10 | 10   20   30   40   50   60   70   80   90   100
```

Jednym z problemów jaki musisz rozwiązać jest odpowiednie formatowanie liczb. Sprawdź co zwróci wywołanie metody String#%, np. `"%-3d" % 84` i wykorzystaj je w swoim programie.

Suma liczb

Napisz funkcję, która liczy sumę parzystych liczb z zakresu od 1..n. Spróbuj stwórzyć kilka wersji:

- korzystając z wybranej pętli, np. while
- korzystając z metod Range#step i Enumerable#sum
- wyprowadzając/znajdując wzór na sumę takich liczb (podpowiedź: jest to ciąg arytmetyczny)

FizzBuzz

Napisz program który wypisze na ekran kolejne liczby od 1 do 100. Jednakże dla liczb podzielnych przez 3 zamiast liczby wypisze “Fizz”, dla liczb podzielnych przez 5 wypisze “Buzz”. Dla liczb podzielnych zarówno przez 3 i 5 program powinien wypisać “FizzBuzz”.

```
$ ruby fizzbuzz.rb
1
2
Fizz
4
Buzz
Fizz
7
...
```

Algorytm Euklidesa

Napisz program wyliczający największy wspólny dzielnik dwóch liczb używając algorytmu Euklidesa.

```
$ ruby gcd.rb 17856 9792
576
```

Do konwersji łańcucha znaków do liczby użyj metody String#to_i.

Kalkulator

Napisz program będący prostym kalkulatorem, z którym interakcja wygląda jak na poniższym przykładzie:

```
$ ruby calc.rb

Choose operation:
 1. Add numbers
 2. Subtract numbers
 3. Multiply numbers
 4. Divide numbers
 5. Quit
What is your choice?: 1
Enter first number: 4
Enter second number: 9
==> 4 + 9 = 13

Choose operation:
1. Add numbers
2. Subtract numbers
```

```
3. Multiply numbers
4. Divide numbers
5. Quit
What is your choice?: 4
Enter first number: 12
Enter second number: 4
==> 12 / 4 = 3

Choose operation:
1. Add numbers
2. Subtract numbers
3. Multiply numbers
4. Divide numbers
5. Quit
What is your choice?: 5
Bye, bye
```

Pamiętaj o dzieleniu przez 0!

PESEL

Napisz program, który będzie weryfikował poprawność numeru PESEL oraz wyświetlał informacje o dacie urodzenia i płci na jego podstawie. Program powinien przyjmować numer pesel jako parametr.

```
$ ruby pesel.rb 44051401458
Pesel 44051401458 jest prawidłowy.
Informacje o peselu:
 - płeć: mężczyzna
 - data urodzenia: 14-05-1944
```

```
$ ruby pesel.rb 67071961453
Pesel 67071961453 jest nieprawidłowy.
```