仮入部員向け競プロ講習会
Ruby編

initial_d

2020年6月20日

目 次

1　自己紹介

名前：initial_d UEC19 2 年生 I3 クラス

プログラミングは高校生の時から少しやっていて、その当時には HSP などをやっていた。入学してから触れた Ruby が楽しいと感じ、講習会を経て競技プログラミングに入門参加回数が足りないせいかいまだに灰色。

2　Ruby

2.1　Hello,World

Listing 1: Hello_World

```
puts ‘Hello,World’
```

2.2　解説 (C++とは違う点)

…はい。短すぎて説明する部分が本当にありません。
なので、この先は C++との差異をいくつか紹介していきます。
Ruby にはヘッダファイルはありません。しか要求しないと使えないライブラリがいくつかあります。
おそらく Ruby には名前空間の概念はありません。
一行分の処理が終わった際に; が必要ありません。逆に一行中に複数書くときに; が必要になります。
Ruby ではプログラムの終わりに return などを書く必要はありません。

2.3　型

Ruby では、C++のような型の概念がありますが、この言語では型を動的に指定することができる、すなわち後付けで型を変えることができるということです。

Ruby での型は以下の通りです。

種別	名前	他からの変換方法
整数	Integer	.to_i を末尾につける
浮動小数点	Float	.to_f を末尾につける
文字列	String	.to_s を末尾につける

2.4　入力、出力

Ruby では標準入力によって入力されたものは「stdin」という関数の中に入っている。これを何かしらの変数へと代入すれば入力が取れたことになる。

また、ruby で標準出力にしゅつりょくするには、基本的には「puts」を使えばよい。使用例は以下に書いてある。

2.5　四則演算

ここについてはあまり C++と変わりがありません。

3 A - Welcome to AtCoder

3.1 Ruby

Listing 2: Welcome to AtCoder

```
# 整数の入力
a = gets.to_i
# スペース区切りの整数の入力
b,c=gets.chomp.split(" ").map(&:to_i);
# 文字列の入力
s = gets.chomp
# 出力
print("#{a+b+c} #{s}\n")
```

3.2 解説

to_i：Ruby では基本的に入力を文字列として受けているので、これによって数字として扱うようにすることができる。
chomp：文字列の末尾にある改行文字 (\n) を削除することができる。
split：カッコ内で指定した文字で文字を分割することができる。
#{}： この内部での計算結果をを文字列として:出力することができる。（式展開）

4 A - abc of ABC

4.1 Ruby

Listing 3: abc of ABC

```
n = $stdin.gets.chomp

flag = 0
if n[0] != n[1]
  if n[1] != n[2]
    if n[0] != n[2]
      flag = 1
    end
  end
end

if flag == 1
  puts "Yes"
else
  puts "No"
end
```

4.2 解説

n[0]：Ruby では文字列と配列をほとんど同様に操作することができる。

5 B - Small and Large Integers

5.1 Ruby

Listing 4: Small and Large Integers

```ruby
a,b,k = $stdin.gets.chomp.split(' ').map(&:to_i)

num = a

k.times do
  puts num
  num = num + 1
  if num > b
    exit
  end
end

if num > b-k
    (b - num + 1).times do
    puts num
    num = num + 1
  end
else
  num = b - k + 1
  k.times do
    puts num
    num = num + 1
  end
end
```

5.2 解説

k.times：ここから end までの内容を k 回ぶん繰り返し行うことができる。

なお atcoder では過去の提出を参照できる。より参考になる提出例があるかもしれない。