4 Reunião 05: 15/SET/2020

4.1 Avisos

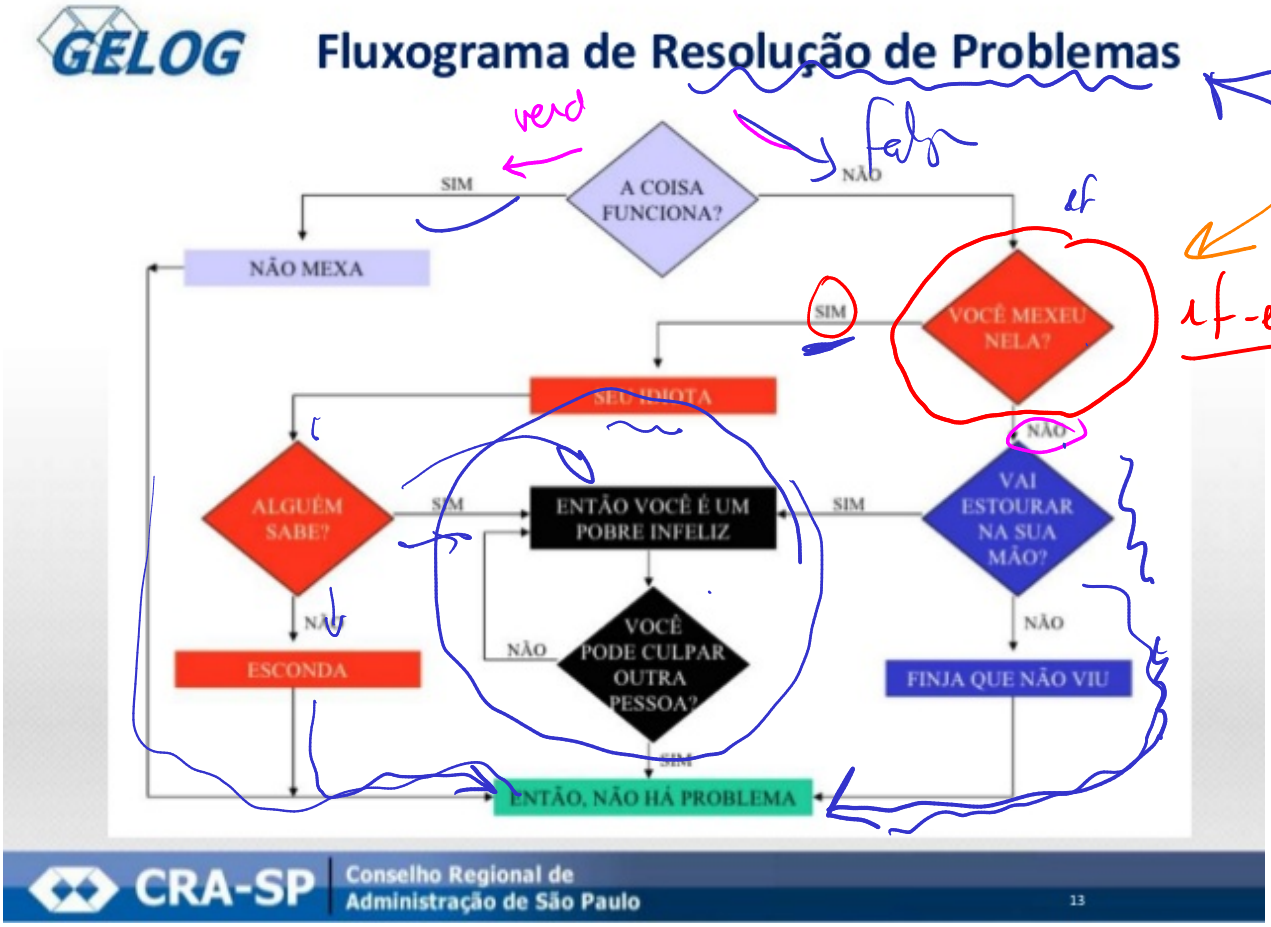
- EPs
- provinhas
- avaliação dos EPs é divulgada **imediatamente após** a submissão
- avaliação das provinhas é divulgada **imediatamente após** o fim período de disponibilidade
- Não deixe de visitar Haile, toda segunda às 13h.

4.2 Aulas passadas revisão

- variáveis
- comando de atribuição
- função de escrita na tela (= `print()`)
- função de leitura via teclado (= `input()`)
- tipos `str`, `int`, `bool` e `float`
- funções de conversão `str()`, `int()`, `bool()` e `float()`
- f-strings
- expressões e operadores relacionais
- Spyder
- comando de repetição `while`
- contador de iterações: imprime primeiros ímpares

4.3 Hoje

- execução condicional `if`
- execução alternativa `if-else`
- operadores de divisão inteira `//` e resto de divisão `%`



4.4 Exercício: conta pares e ímpares

Dados um número inteiro `n`, `n > 0`, e uma sequência com `n` números inteiros, determinar quantos números da sequência são **pares** e quantos são **ímpares**.

Por exemplo, para a seqüência

`6 -2 7 0 -5 8 4`

o seu programa deve escrever

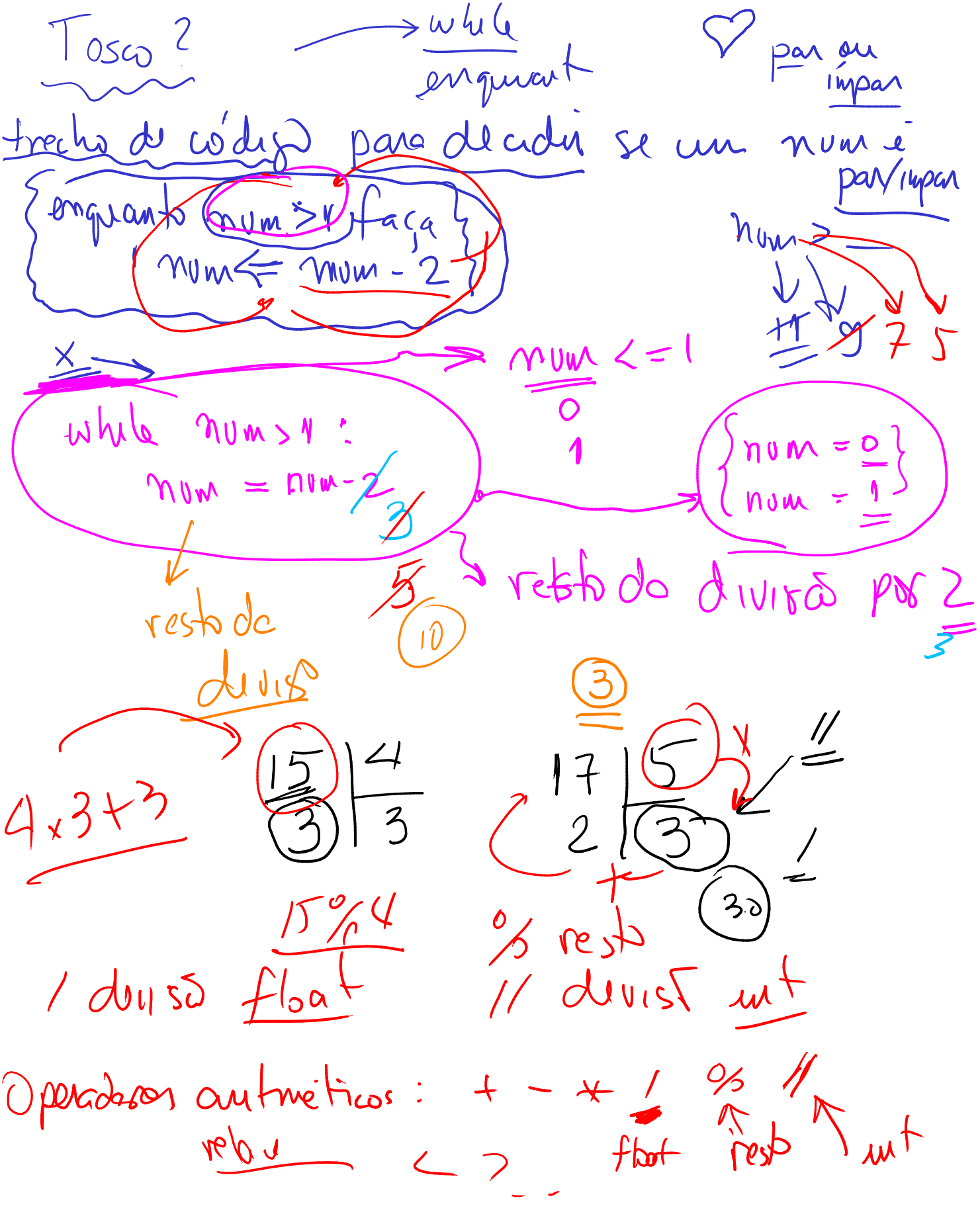
```
4 números pares
2 números ímpares
```

4.4.1 Exemplos

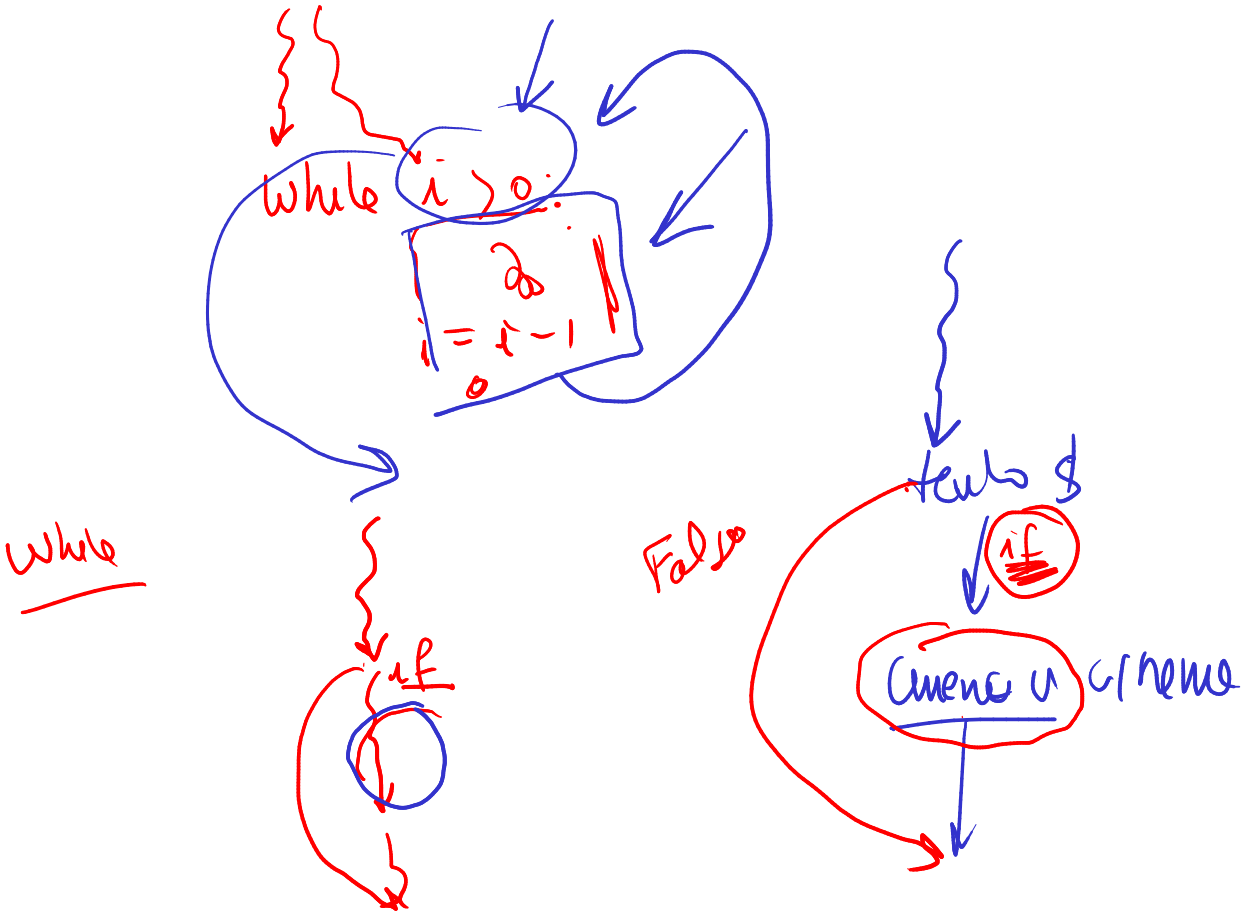
```
Digite o tam da sequencia: 6
Digite um número: -2
Digite um número: 3
Digite um número: 0
Digite um número: 28
Digite um número: -123
Digite um número: 8
A sequência é composta por:
    4 números pares e
    2 números ímpares

Digite o tam da sequencia: 4
Digite um número: 1
Digite um número: 3
Digite um número: 5
Digite um número: 7
A sequência é composta por:
    0 números pares e
    4 números ímpares
```

4.4.2 Rascunhos



Mais rascunhos



Rascunho 1

```
leia n
enquanto há números serem lidos faça
   leia num

   se num é par
      contapar <-- contapar + 1
```

Rascunho 2

```
leia n

enquanto há números serem lidos faça
    leia num

    se num é par
        contapar <-- contapar + 1

    se num é ímpar
        contaímpar <-- contaímpar + 1
```

Rascunho 3

```
leia n

enquanto há números serem lidos faça
    leia num

    se num é par
        contapar <-- contapar + 1
    senão
        contaímpar <-- contaímpar + 1
```

4.4.3 Como saber se num é par?

Tosco?!? Hmm. Engenhoso!

```
enquanto num > 1 faça
    num = num - 2;

enquanto num < -1 faça
    num = num + 2;

se num == 0 então
   contapar   <- contapar + 1
senão
   contaímpar <- contaimpar +1
```

Esse é um trabalho para o super mega hiper operador `%` de **resto de divisão**!

4.4.4 Soluções

Solução 1

```
# 1 leia o número de elementos da sequencia
# atribuição multipla, coisa cosmética
n = n_salvo = int(input("Digite o tam da seq: "))

# 2 crie e inicialize contador de pares
conta_par = 0

# 3 conte o quantidade de pares
while n > 0:
    # 3.1 leia um número
    num = int(input("Digite um num da seq: "))
    # 3.2 verifique se número é par
    if num % 2 == 0:
        conta_par = conta_par + 1
    # 3.3 diminua o contador de números a serem lidos
    n = n - 1

print(f"{conta_par} números pares")
print(f"{n_salvo-conta_par} números ímpares")
```

Solução 2

```
# 1 leia o número de elementos da sequencia
n = int(input("Digite o tamanho da sequência: "))

# 2 crie e inicialize contadores
conta_par = conta_impar = 0

# 3 conte a qunatidade de pares e ímpares
i = 0 # contador de números lidos
while i < n:
    # 3.1 leia um número
    num = int(input("Digite um inteiro: "))
    # 3.2 verifique se número é par
    if num%2 == 0:
        conta_par = conta_par + 1
    else:
        conta_impar = conta_impar + 1
    # 3.3 incremente o contador de números a serem lidos
    i = i + 1

print(f"{conta_par} números pares")
print(f"{conta_impar} números ímpares")
```

4.5 Operadores /, // e %

/ divisão

```
5 / 2 é 2.5
1 / 2 é 0.5
10 / 6 é 1.6666...7
```

// divisão inteira

```
5 // 2  é 2
1 // 2  é 0
10 // 6  é 1
```

% resto da divisão

```
5  % 2 é 1
1  % 2 é 1
10 % 6 é 4
```

Cuidado com números negativos, depende da linguagem:

```
-5 // 2 == -3
-5 %  2 ==  1 # resto do mesmo sinal que divisor
 5 % -2 == -1 # resto menor que o segundo operando em módulo
```

Sempre é verdade que `a == (a//b)*b + a%b`

operadores aritméticos: `+`, `-`, `*`, `/`, `**`, `//`, `%`

operadores relacionais: `<`, `>`, `<=`, `>=`, `==`, `!=`

4.6 Execução condicional: `if`

```
if condicao:
    # lista de comandos
    comando_1
    comando_2
    ...
```

4.6.1 Significado

Executa os comandos tabulados, `lista de comandos` se a condição é verdadeira.

4.7 Execução alternativa: `if-else`

```
if condição:
    # lista de comandos_v
else:
    # lista de comandos_f
```

4.7.1 Significado

Se `condição` é verdadeira executa `lista de comandos_v`.

Se `condição` é falsa executa `lista de comandos_f`.