AulaRemota30Marco

March 29, 2020

1 Aula de 30/03 - Um pouco mais de recursão

Nessa aula veremos alguns problemas, que vão usar técnicas distintas, mas ainda usando formas simples de recursão

Uma das coisas que podemos fazer com números inteiros é separar os seus dígitos um a um. Observe o cógido abaixo:

```
In [3]: function imprimedigito(n)
            if n <= 9
                println(n)
                println("fim")
            else
                println(n % 10)
                imprimedigito(n ÷ 10)
            end
        end
```

```
Out[3]: imprimedigito (generic function with 1 method)
```

```
In [4]: imprimedigito(65767)
```

```
7
6
7
5
6
fim
```

A função, imprime os dígitos do número de entrada do menos significativo ao mais significativo. Para isso, combina duas operação, o resto por 10, e a divisão por 10.

É possível fazer algumas operações úteis com essa técnica, como por exemplo, somar os dígitos de um número. Se um número é divisível por três a soma dos seus dígitos também é divisível por três.

Mas, para começar, vamos fazer o teste para uma função **somadigitos(n)** Quais seria casos interessantes a verificar?

```
In [ ]: function testesd()
        end
```

Agora sim, vamos pensar em como fazer para somar os dígito de um número, a cada etapa, temos que descascar uma parte (% 10), para em seguida tirar essa parte (div 10).

```
In [ ]: function somadigitos(n)
        end
```

```
In [ ]: # aqui é o local para brincar com a função :) por exemplo chamando a função recursivam
```

Podemos tamném verificar se a valem algumas propriedades, como por exemplo, sera que dados dois números a soma dos seus dígitos é igual ao à soma dos dígitos da soma deles?

De outra forma: somadigitos(a) + somadigitos(b) == somadigitos(a + b)

e para o produto?

Quem sabe usar números aleatórios **rand(Int)** pode ajudar nessa verificação

```
In [5]: rand(Int)
```

```
Out[5]: -441004277476167877
```

Uma outra propriedade interessante que pode ser verificado usando o operador de resto da divisão é a primalidade. Isso é ver se um número n é divisível apenas por ele mesmo e por 1.

Mas, como sempre, vamos começar com os estes, assumindo que teremos a função **éprimo(x)**

```
In [ ]: function testeéprimo()
        end
```

Agora sim, vamos pensar em como fazer a função, a forma simples e direta é tentar dividir o número candidato, por número de 2 até n / 2, se algum dividir, temos que o número não é primo. Caso contrário ele é primo.

Agora que já temos a nossa primeira função, podemos pensar em como otimizar

Para terminar, vamos imprimir os 100 primeiros primos?

Se der tempo, vamos imprimir uns primos de Mersenne. Um primo é de Mersenne se ele é da forma $2^n - 1$.