Alfredo's MAC0110 Journal

Alfredo Goldman

March 10, 2020

0.1 Aula 03 -*[2020-03-09 seg]*

Objetivo: Ver o interpretador de Julia como uma calculadora poderosa, introduzir a noção de variáveis

0.1.1 Começando com o modo interativo do Julia.

Quem quiser já pode instalar o ambiente de programação, usem esse link

Dentro do Julia (após chamar julia na linha de comando), vamos começar com números inteiros:

```
1 + 2
```

```
3
```

```
40 * 3
```

```
120
```

```
84 / 2
```

Notem que nesse caso, houve uma mudança de tipos, pois 84 e 2 são inteiros e o resultado é um número em ponto flutuante (float)

```
42.0
```

Também é possível pedir o resultado inteiro usando o operador div:

```
div(84,2)
```

```
42
```

Também dá para fazer a exponenciação:

```
2^31
```

```
2147483648
```

Expressões mais complexas também podem ser calculadas:

```
23 + 2 * 2 + 3 * 4
```

```
39
```

Sim, a precedência de operadores usual também é válida em Julia. Mas, ai vem a primeira lição de programação: * Escreva para humanos, não para máquinas *

```
23 + (2 * 2) + (3 * 4)
```

Em julia também podemos fazer operações com números em ponto flutuante:

```
23.5 * 3.14
```

```
73.79
```

ou

```
12.5 / 2.0
```

```
6.25
```

Acima temos mais um exemplo de código escrito para pessoas, ao se escrever 2.0 estamos deixando claro que o segundo parâmetro é um número float.

É importante saber que números em ponto flutuante tem precisão limitada, logo não se espante com resultados inesperados como abaixo:

```
1.2 - 1.0
```

ou

```
0.1 + 0.2
```

ou ainda

```
10e15 + 1 - 10e15
```

Um outro operador interessante é o % que faz o resto da divisão

```
4 % 3
```

0.1.2 Variáveis e seus tipos

Em Julia também temos o conceito de variáveis, que servem para armazenar os diferentes conteúdos de dados possíveis.

```
a = 7
2 + a
```

É importante notar que as variáveis em Julia podem receber novos valores e o tipo da variável depende do que foi atrubuído inicialmente.

```
a = 3
a = a + 1
typeof(a)
```

Aproveitando o momento, podemos ver que há vários tipos primitivos em Julia, sendo os principais:

```
typeof(1)
typeof(1.1)
typeof("Bom dia")
```

```
[....]
```

Falando em strings, elas são definidas por conjuntos de caracteres entre aspas como:

```
s1 = "Olha que legal"
s2 = "Outra String"
```

Dá também para fazer operações como strings como concatenação:

```
s1 = "Tenha um"
s2 = " Bom dia"
s3 = s1 * s2
```

Ou potência:

```
s = "Nao vou mais fazer coisas que possam desagradar os meus colegas"
s ^ 10
```

Ainda sobre variáveis, há umas regras com relação aos seus nomes, tem que começar com uma letra, pode ter dígitos e não pode ser uma palavra reservada. É bom notar que Julia por ser uma linguagem moderna, aceita nomes de caracteres em unicode, pode exemplo

```
\delta = 2  # Para se fazer o delta, deve se digitar \ seguido de delta, seguido de <tab>
```

0.1.3 Saída de dados

Para fazer saídas usam-se dois comandos, print() e o println(), sendo que o primeiro não pula linha e o segundo pula.

```
print("Hello ")
println("World!")
println("Ola, mundo!")
```

Para evitar que se digitem muitos caracteres, por vezes podemos usar "açucares sintáticos".

```
x = 1
x = x + 1
x += 1  # forma equivalente a acima
```

0.2 Aula 04 -*[2020-03-11 qua]*

Objetivo: Começar a entender como funcionam as funções

0.2.1 O uso de funções é uma abstração natural

Na aula passada já vimos umas funções e isso foi bem natural, foram elas:

- typeof() - Que dado um parâmetro devolve o seu tipo
- div() - Que dados dois parâmetros devolve a divisão inteira do primeiro pelo segundo
- print() e println() - Que dados diversos parâmetros os imprime, sem devolver nada

Inclusive, aqui vale a pena ver que podemos pedir ajuda ao Julia para saber o que fazem as funções. Para isso, se usa o ? antes da função:

```
?typeof()
?div()
?print()
```

Ao fazer isso, inclusive descobrimos que o div() pode ser usado também como ÷.
Uma outra função bem útil é a que permite transformar um tipo de valor em outro.

```
parse(Float64, "32")
```

Para conversão de valores em ponto flutuante para inteiros, temos a função trunc.

```
trunc(Int64, 2.25)
```

De forma inversa temos o float.

```
float(2)
```

Finalmente, podemos transformar um valor em uma string, como em:

```
string(3)
```

ou

```
string(3.57)
```

Também tem muitas funções matemáticas prontas como

- sin(x) - calcula seno de x em radianos
- cos(x)
- tan(x)
- deg2rad(x) - converte x de graus em radianos
- rad2deg(x)
- log(x) - calcula o logarítmo natural de x
- log(x, b) - calcula o logarítmo de x na base b
- log2(x) - calcula o logarítmo de x na base 2
- log10(x)
- exp(x) - calcula o expoente da base natural de x
- abs(x) - calcula o módulo de x
- sqrt(x) - calcula a raiz quadrada
- isqrt(x) - calcula a raiz quadrada inteira de x
- cbrt(x) - raiz cúbica de x
- factorial(x) - calcula o fatorial de x