

Lógica da Computação - 2019/2

Roteiro Zero - Simple Calculator **v0.1.x**

Raul Ikeda - rauligs@insper.edu.br

Entrega: 11/Aug/2019

Nome:

GitHub:

Objetivos

1. Iniciar a construção de um compilador.
2. Relembrar máquina de estados.

Compilador

1. Vamos iniciar a construção do compilador, mas qual é o principal propósito de um compilador?

- >> Ver Cap. 1.1 Aho et al
2. Como ele funciona?

>> Ver Cap. 1.2 Aho et al

IR vs ML

1. Faça um comparativo elencando as principais vantagens e desvantagens entre Linguagem de Máquina e Representação Intermediária.

>> Ver Pag. 228 e 321 Aho et al

Por dentro do *Blackbox*

>> Ver Cap. 1.6 J. J. Neto

Em qual etapa seria melhor otimizar o código de saída?

O papel de cada etapa do compilador

>> Ver Cap. 1.5 J. J. Neto

Tarefas do Roteiro:

1. Construir um programa que recebe como argumento uma cadeia de somas e subtrações de números inteiros de múltiplos dígitos. Ao final deve exibir o resultado da operação.
- Escolher uma linguagem orientada a objetos, com recursão e com dicionários.
 - Montar um repositório **privado** no GitHub. Adicionar o professor como colaborador.
 - Receber o argumento na chamada do programa como *string*. Por exemplo:

```
$ python compilador.py '1+1'
```

Base de Testes:

```
>> 1+2
>> 3-2
>> 11+22-33
>> 789 +345 - 123
```

Fazendo *Releases* no Git

Para realizar a entrega do código da versão atual, você deve fazer um *release* da versão no Git. Para tal, é necessário acrescentar uma *tag* e realizar um *commit* específico.

No exemplo abaixo, como esse roteiro implementa a versão 0.1.x do compilador, utilizaremos os seguintes comandos:

```
$ git tag -a v0.1.1 -m "Mensagem sobre o release"
$ git push origin v0.1.1
```

```
# Para seleccionar a versão (note que não é um branch):  
$ git checkout v0.1.1
```

Atenção: Só é possível colocar uma *tag* apenas uma vez, portanto solte o *release* quando houver acabado apenas. Caso precise soltar uma correção, utilize um incremento no número do *build*, por exemplo 0.1.2, 0.1.3, etc. MAS nunca deixe de realizar *commit* e *push* a cada modificação.

Questionário

1. Explique como foi feito para reconhecer múltiplos dígitos e realizar múltiplas operações.
2. Pense na estrutura de alguma linguagem procedural (C por exemplo), indique com detalhes como você expandiria o seu programa para compilar um programa nessa linguagem.