Guidiscript

- V2.0 (Fase 2, MAC316)

Grupo:

- Vitor Silva - 10263531
- Vitor Guidi - 8038091
- Lucas Fujiwara - 10737049

Instruções

- Requerimentos:
  - GCC
  - Racket (pacote `plai-typed`)
  - GNU Flex
  - GNU Bison
- Para compilar o projeto: `make`
- Para rodar o programa `./roda.sh` (lembre-se de aplicar `chmod +x roda.sh`)

Organização do projeto

Além dos arquivos `direto.rkt`, `mcalc.l` e `mcalc.y` também temos `repl-start.rkt`, `repl-end.rkt` e `stdlib`.

Eles são combinados junto com o `direto.rkt` pelo `make dinamico.rkt` de modo que todas as funções definidas em `stdlib` estejam disponíveis para uso. Assim as temos uma biblioteca padrão nativa em *guidiscript*.

Sintaxe

- Operações aritméticas normais (infixa, $+$, $-$, $\times$, $\div$)

```
5 + 7 # = 8
5 * 3 + 8 # = 23
10 / 5 # = 2
- 3 # = -3
4 - 5 # = -1
```

- Condicional (`!= 0` significa `true`, caso contrário, é `false`) (semelhante ao op. ternário)

```
quod cond ? ifTrue : ifFalse!
```

- Chamada de função (tipo SmallTalk)

```
[func arg]
[fatorial 5000000000] # Vai demorar
```

- Declaração de variáveis (Fase 2)

```
paro <simbolo> (<value>)              # Cria ou altera um valor de uma
variavel
# exemplo
paro x (10)                           # Cria uma variavel x
tum paro x (x + 10)                   # Atualiza o valor dela com base no
valor antigo
tum propono x                          # E mostra o valor dela
```

- Declaração de funções (Fase 2)

```
transeat <simbolo> ad (<expressão>)     # Cria um lambda
functio <nome> <simbolo> ad (expressão) # Cria uma função (permite auto-
referência)
# exemplo
transeat x ad (x + 1)                   # Cria
[transeat x ad (x + 1) 3]               # Aplica
functio sum x (quod x ? x + [sum x-1] : 0 !) # Cria
tum [sum 10]                            # Calcula
```

- Sequências (Fase 2)

```
pario add1 (transeat x ad (x+1)) tum [add1 6]
```

- Visualizando valor de variaveis (Extra)

```
propono (<expressão>)                    # Mostra o resultado de uma
expressão
```

  Como não mostramos o resultado por padrão, é preciso do propono para visualizar a saida.

Biblioteca

- Função fibonacci

```
[fib 50]
```

  implementação

```
functio fib x (quod x ? quod x - 1 ? [fib x-1] + [fib x-2]: 1 ! : 1 !)
```

- Função fatorial

```
[fatorial 50]
```

  implementação

```
functio fatorial x (quod x ? x * [fatorial x-1] : 1 !)
```

- Função Torre de Hanoi

```
[hanoi 3]
```

A implentação está na `stdlib`, o legal é rodar e depois testar em algum simulador de hanoi. A primeira linha é de onde tira a peça, a segunda aonde coloca e assim repete.

Projeto

- Funções aceitam apenas 1 argumento (ainda)

Realizar testes

Para aplicar os testes basta rodar `./roda.sh < teste`
Os arquivos de teste para a fase 2 estão em `testes/2` e dentro deles tem a saida esperada de cada `propono`.