Universidade Estadual de Maringá
Departamento de Informática
Professor: Wagner Igarashi
Disciplina: Paradigmas de Programação Lógica e Funcional - 6902
Aluno: Thiago Issao Yasunaka **RA:** 103069

PROGRAMA PARA ANÁLISE DE CÓDIGO ESCRITOS NA LINGUAGEM RACKET

Maringá, Paraná

2021

Descrição

A implementação do avaliador de código em racket desenvolvido no projeto visa obter uma análise geral de determinado arquivo escrito na linguagem racket. Para fazer esta avaliação alguns critérios foram estabelecidos a fim de ser possível estabelecer uma nota final. No tópico abaixo estão descritos todos os critérios definidos para a avaliação.

Critérios de Avaliação

Foram implementadas quatro métricas para a avaliação do código fonte escrito na linguagem racket, sendo estas:

1. Quantidade de linhas no arquivo.
2. Quantidade de comentários utilizando o character “;”.
3. Quantidade de definições (em racket utiliza-se a palavra-chave *define*).
4. Quantidade de linhas do código que ultrapassaram determinada quantidade de caracteres em uma única linha, o padrão pré-estabelecido foi 70. Ex: 4 linhas no código possuem mais 70 caracteres em uma única linha.

Para cada métrica levantada, existe um peso associado a ela, pois cada métrica possui um grau de relevância para a nota final. Para isto, existe na implementação do trabalho uma função que faz todos estes cálculos. A tabela abaixo detalha os pesos para cada métrica:

Métrica	**Peso**
1	5
2	1
3	2
4	2

Todas as métricas definidas são valores que resultam em um número, isto é, valores que não são de decisão e sim, quantitativos. Para obter uma nota final para cada métrica, é necessário fazer uma avaliação de 0 a 10.0. A tabela abaixo mostra em detalhes como são determinadas as notas para cada métrica:

1. Quantidade de linhas no arquivo	
Quantidade de linhas	Nota
Até 300 linhas	10.0

Até 400 linhas	6.0
Até 500 linhas	3.0
Acima de 500 linhas	0.0

2. Quantidade de comentários utilizando o character “;”	
Quantidade de comentários	Nota
Até 10% do total de linhas no código	10.0
Até 50% do total de linhas no código	5.0
Até 75% do total de linhas no código	2.0
Acima de 75%	0.0

3. Quantidade de definições	
Quantidade de definições	Nota
Até 25 definições	10.0
Até 40 definições	8.0
Até 50 definições	6.0
Até 60 definições	3.0
Acima de 60 definições	2.0

4. Quantidade de linhas do código que ultrapassaram determinada quantidade de caracteres em uma única linha	
Quantidade	Nota

Nenhuma linha ultrapassou o limite de caracteres	10.0
ultrapassou 1 ou mais linhas	0.0

EXEMPLOS

Ao executar o avaliador para os arquivos testes, é gerado todos os resultados obtidos e para conseguir observar estes resultados basta digitar o seguinte comando: (No projeto, existem dois códigos fonte para fim de testes, eles estão fixos em */examples*) .

- `Para o exemplo 1`
  `>(finalGrade testsFile2)`

- `Para o exemplo 2`
  `>(finalGrade testsFile1)`

Pelo fato do exemplo 1 ter um código fonte muito extenso, abaixo será exibido apenas as informações do exemplo 2.

- **Este é o código fonte do exemplo 2:**

```
#lang racket

(require rackunit)
(require rackunit/text-ui)

;;;;;;;;;;;;;;;;;;;
;; Definição de Ponto

(struct ponto (x y) #:transparent)
;; Ponto representa um ponto no plano cartesiano
;;   x : Número - a coordenada x
;;   y : Número - a coordenada y
;; Exemplos
#; (define p1 (ponto 3 4))
#; (define p2 (ponto 8 2))
;; Devolve uma versão não aninhada de lst, isto é,
;; uma lista com os mesmos
```

- **A saída para o exemplo 2 será o seguinte:**

```
------------------------racket file-------------------------
Nota para quantidade de linhas: 5.0
Nota para quantidade de comentários: 0.2
Nota para quantidade de definições: 2.0
Nota para comprimento de uma linha: 2.0
NOTA FINAL: 9.2
-------------------------------------------------
```

Além de executar os testes, é possível executar um único arquivo escolhido pelo usuário ou então, executar um diretório completo contendo apenas arquivos `.rkt`. Para realizar esta operação, basta fazer os seguintes processos:

- **Executar um arquivo escolhido pelo usuário**
  `>(evaluateFile)`
  `Exemplo de entrada: examples/example2.rkt`

- **Executar um diretório completo contendo apenas arquivos racket**
  `>(evaluateDir)`
  `Exemplo de entrada: ../{DIR_NAME}/`
  `Obs: A barra (“/”) no final é obrigatória`

A saída resultante da execução do algoritmo será igual a saída mostrada no exemplo 2 acima, porém para a avaliação por diretório é mostrado o nome do arquivo na qual se está sendo realizada a avaliação, por exemplo:

```
------------------------main.rkt-------------------------
Nota para quantidade de linhas: 5.0
Nota para quantidade de comentários: 0.2
Nota para quantidade de definições: 2.0
Nota para comprimento de uma linha: 2.0
NOTA FINAL: 9.2
-------------------------------------------------
... outros arquivos presentes no diretório
```

Referências

DEVELOPERS, Racket. **Racket Documentation**. Disponível em: https://docs.racket-lang.org/. Acesso em: 30 abr. 2021.

Todos os documentos disponibilizados na disciplina.