

SUMÁRIO - AULA 14

- Paradigma Funcional
 - 2º aula (final)



NÚMEROS

(+ 4 #b111)

Tente somar

- 9999... ; inteiros
- #b111 ; binário => 7
- #o111 ; octal => 73
- #x111 ; hexa => 273
- 3.14 ; reais
- 6.02e+23 ; racionais
- $1/2$; racionais
- $1+2i$; numeros complexos

The screenshot shows the DrRacket application window titled "SemNome 5 - DrRacket". The menu bar includes "Arquivo", "Editar", "Ver", "Linguagem", "Racket", "Insert", "Tabs", and "Ajuda". Below the menu is a toolbar with icons for opening files, saving, undo, redo, running, and other development actions. A dropdown menu shows "SemNome 5 ▾ (define ...) ▾". The main editing area contains a single line of code: "#lang racket". At the bottom, there is a status bar with the text "Determine language from source ▾" and a zoom level indicator set to "2:0".

SemNome 5 - DrRacket

Arquivo Editar Ver Linguagem Racket Insert Tabs Ajuda

SemNome 5 ▾ (define ...) ▾

#lang racket

Bem-vindo a **DrRacket**, versão 5.3.6 [3m]
Linguagem: racket, memory limit: 128 MB.
> 99999999999999999999999999999999 ; inteiros
#b111 ; binário => 7
#o111 ; octal => 73
#x111 ; hexadecimal => 273
3.14 ; reais
6.02e+23 ; racionais
1/2 ; racionais
1+2i ; numeros complexos
99999999999999999999999999999999
7
73
273
3.14
6.02e+023
1
2
1+2 i
>

Determine language from source ▾ 2:0

STRINGS

Concatenação

- (string-join '("one" "two" "three" "four"))
- (string-join '("one" "two" "three" "four") ", ")

separador

Repetição

- (make-string 3 #\z)

```

1 | #lang racket

Bem vindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (string-join '("one" "two" "three" "four"))
"one two three four"
> (string-join '("one" "two" "three" "four") ", ")
"one, two, three, four"
> (make-string 3 #\z)
"zzz"
>
  
```

STRINGS

Convertendo valores para String via ~a

- (~a "um texto já") ; texto
- (~a 17) ; número
- (~a pi) ; constante
- (~a #\b) ; caractere
- (write (~a pi))
- (write (~a "o valor de pi é: " pi))

```

1 | #lang racket

Bem vindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (~a "um texto já")
"um texto já"
> (~a 17) ; número
"17"
> (~a pi) ; constante
"3.141592653589793"
> (~a #\b) ; caractere
"b"
> (write (~a pi))
"3.141592653589793"
> (write (~a "o valor de pi é: " pi))
"o valor de pi é: 3.141592653589793"
>
  
```

STRINGS

❑ Caracteres Especiais

❑ Nova linha

- ❑ (printf "Pulando linhas \n\n\n")
- ❑ (printf "Pulando linhas ~n~n~n")
- ❑ (printf "Pulando linhas ~%~%~%")

```

1 | #lang racket

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (printf "Pulando linhas \n\n\n")
Pulando linhas

> (printf "Pulando linhas ~n~n~n")
Pulando linhas

> (printf "Pulando linhas ~%~%~%")
Pulando linhas

>
  
```

STRINGS

❑ Caracteres Especiais

❑ Escape

- ❑ (printf "Eu sou \"Racket\". Venha aprender!\n")

❑ Caracteres

- ❑ #\A ;=> #\A
- ❑ #\λ ;=> #\λ
- ❑ #\u03BB ;=> #\λ

```

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (printf "Eu sou \"Racket\". Venha aprender!\n")
Eu sou "Racket". Venha aprender!

> #\A ; => #\A
#\A
> #\λ ; => #\λ
#\λ
> #\u03BB ; => #\λ
#\λ
>
  
```

STRINGS

Formatar

(format "~a x ~a" "Time A" "Time B")

Retirar caracter

(string-ref "Amizade" 0) ; => #\A

(string-ref "Amizade" 2) ; => #\i

```

SemNome 12 - DrRacket
Arquivo Editar Ver Linguagem Racket Insert Tabs Ajuda
SemNome 12 (define ...)
Bem-vindo a DrRacket, versão 5.3.6 [3m]
Linguagem: racket; memory limit: 128 MB.
> (format "~a x ~a" "Time A" "Time B")
"Time A x Time B"
> (string-ref "Amizade" 0) ; => #\A
#\A
> (string-ref "Amizade" 2) ; => #\i
#\i
Determine language from source
7:35

```



IGUALDADE (eq? v1 v2)

eq? retorna #t se v1 e v2 referem-se ao mesmo objeto e #f caso contrário.

(eq? 'yes 'yes)

#t

(eq? (cons 1 2) (cons 1 2))

#f

(eq? (make-string 3 #\z) (make-string 3 #\z))

#f

```

SemNome 4 - DrRacket
Arquivo Editar Ver Linguagem Racket Insert Tabs Ajuda
SemNome 4 (define ...)
Bem-vindo a DrRacket, versão 5.3.6 [3m]
Linguagem: racket; memory limit: 128 MB.
> (eq? 'yes 'yes)
#t
> (eq? (cons 1 2) (cons 1 2))
#f
> (eq? (make-string 3 #\z) (make-string 3 #\z))
#f
> |
Determine language from source
9:2

```




IGUALDADE (equal? v1 v2)

- ❑ `equal?` retorna `#t` se `v1` e `v2` tem o mesmo valor e `#f` caso contrário.
 - ❑ `(equal? 'yes 'yes)`
 - ❑ `#t`
 - ❑ `(equal? 2 2.0)`
 - ❑ `#f`
 - ❑ `(equal? (expt 4 2) (expt 2 4))`
 - ❑ `#t`
 - ❑ `(equal? (expt 2 100) (expt 2 100))`
 - ❑ `#t`


```

1 | #lang racket

Benvido a DrRacket, versão 5.3.6 [3m].
Linguagem: racket, memory limit: 128 MB.
> (equal? 'yes 'yes)
#t
> (equal? 2 2.0)
#f
> (equal? (expt 2 100) (expt 2 100))
#t
> (equal? (expt 4 2) (expt 2 4))
#t
>
  
```

EXERCÍCIO



- ❑ Implementar em Racket uma **calculadora** com as 4 operações **básicas** e chamar as operações:
 - ❑ a) individualmente
 - ❑ Ex: `(somar 2 3)`
 - ❑ Ex: `(multiplicar 2 5)`
 - ❑ b) em cadeia 
 - ❑ Ex: `(multiplicar 2 (somar 2 3))`

```

1 | #lang racket
2 | (define (somar a b)
3 |   (+ a b)
4 | )

Welcome to DrRacket, version 6.10.1 [3m].
Language: racket, with debugging; memory limit: 512 MB.
> (somar 7 3)
10
>
  
```

RESPOSTA DO EXERCÍCIO



```

1 #lang racket
2
3 (define (somar a b)
4   (+ a b)
5 )
6
7 (define (subtrair a b)
8   (- a b)
9 )
10
11 (define (multiplicar a b)
12   (* a b)
13 )
14
15 (define (dividir a b)
16   (/ a b)
17   ;(write(list "dividiu: " (/ a b) ))
18 )
  
```

Bem-vindo a DrRacket, versão 5.3.6 [3m].
 Linguagem: racket, memory limit: 128 MB.
 > (multiplicar 2 (somar 2 3))
 10
 >

Background expansion finished
 Determine language from source ▼ 17:2

Caso queria debugar...

EXERCÍCIO



❑ Incrementar a calculadora com as funções:

- ❑ dobro
- ❑ triplo
- ❑ metade
- ❑ quadrado
- ❑ cubo

```

1 #lang racket
2
3 (define (dobro a)
4   (multiplicar a 2)
5 )
6
7 (define (triplo a)
8   (multiplicar a 3)
9 )
10
11 (define (metade a)
12   (dividir a 2)
13 )
14
15 (define (quadrado a)
16   (multiplicar a a)
17 )
18
19 (define (cubo a)
20   (multiplicar a (quadrado a))
21 )
  
```

Bem-vindo a DrRacket, versão 6.1.1 [3m].
 Linguagem: racket, memory limit: 128 MB.
 > (dobro 5)
 10
 >

Determine language from so... 19:17 493.87 MB

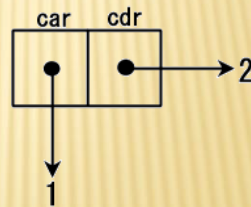
reuso

PARES DE VALORES



- Um **cons** (*construct*) é um registro de 2 **campos** que permite criar estruturas de dados.

(cons 1 2)



(define p (cons 1 2))

PARES DE VALORES



- Por razões históricas*, os **campos** são chamados de **car** e **cdr**:

- Contents of the Address part of Register number*

- Contents of the Decrement part of Register number*

- ***registradores** utilizados para se implementar o **cons** via **assembler** na **primeira máquina** de Lisp

PARES DE VALORES



❑ Criando um par via **cons**:

❑ `(cons 1 2)`

❑ `(cons 1 '())`

❑ `(cons "head" empty)`

❑ `(cons "head" null)`

```

SemNome 8 - DrRacket
Eicheiro Editar Ver Linguagem Racket Insert Tabs Ajuda
SemNome 8 (define ...)
1 | #lang racket

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (cons 1 2)
'(1 . 2)
> (cons 1 '())
'(1)
> (cons "head" empty)
'("head")
> (cons "head" null)
'("head")
>
Determine language from source 9:22

```

PARES DE VALORES



❑ Listas encadeadas podem ser construídas a partir de **pares** (cons):



❑ o **car** de cada **cons** guarda os **elementos** da lista e

❑ o **cdr** aponta ou para **outro cons** ou para **null/nil**

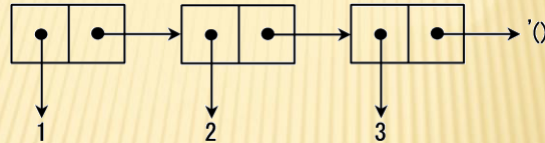
```

SemNome 2 - DrRacket
Eicheiro Editar Ver Linguagem Racket Insert Tabs Ajuda
SemNome 2 (define ...)
Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (define X (cons "A" (cons "B" null)))
> X
'("A" "B")
> |
Determine language from source 6:2

```


PARES DE VALORES

- ❑ Criando uma lista via **cons**:



- ❑ `(cons 1 (cons 2 (cons 3 null)))`

```

SemNome 4 - DrRacket
Eicheiro  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
SemNome 4  (define ...)
Linguagem: racket; memory limit: 128 MB.
> (cons 1 (cons 2 (cons 3 null)))
'(1 2 3)
>
Determine language from source 5:2
  
```

PARES DE VALORES

- ❑ Criando uma lista via **cons**:

- ❑ Retornado os valores:

- ❑ `(car(cons 1 2))`
 - ❑ retorna o primeiro
- ❑ `(cdr(cons 1 2))`
 - ❑ retorna o segundo

```

SemNome 9 - DrRacket
Eicheiro  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
SemNome 9  (define ...)
Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> (cons 'um' (dois três quatro))
'(um dois três quatro)
> (car (cons 1 2))
1
> (cdr (cons 1 2))
(2)
>
Determine language from source 9:2
  
```

- ❑ `(cons 'um'(dois três quatro))`
 - ❑ unindo: oposto das funções **car** e **cdr**

LISTAS, A ESSÊNCIA

❑ Lisp (LISt Processor)



LISTAS, A ESSÊNCIA



❑ Declarar lista via **apóstrofo** e **list**. Ex:

- ❑ '() ou **null**; **lista vazia**
 - ❑ **NIL** em outras funcionais

- ❑ '(1 2 3) ; **lista normal**

```
#lang racket

Bem vindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket; memory limit: 128 MB.
> '()
> null
> '(1 2 3)
>
```

LISTAS, A ESSÊNCIA



- ❑ Declarar lista via **apóstrofo** e **list**. Ex:
 - ❑ `'(1 2 "teste")` ; diferentes tipos
 - ❑ `(list (circle 10) (filled-rectangle 10 20))` ; via funções

```
#lang slideshow

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: slideshow, memory limit: 128 MB.
> '(1 2 "teste") ; diferentes tipos
> (list (circle 10) (filled-rectangle 10 10))
> (O ■)
>
```

OPERAÇÕES COM LISTAS

- ❑ adicionar elementos:
 - ❑ `(append '(1 2) '(3 4))` ; lista
 - ❑ `(append '(1 2) '() '(3 4))` ; listas



```
1 #lang racket

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket, memory limit: 128 MB.
> (append '(1 2) '(3 4)); lista
> (append '(1 2 3 4)
> (append '(1 2) '() '(3 4)); listas
> (1 2 3 4)
>
```



OPERAÇÕES COM LISTAS



- ❑ remover elementos
 - ❑ `(remove 3 '(1 2 3 4))`
- ❑ reverter os elementos:
 - ❑ `(reverse '(1 2 3 4))`
- ❑ tamanho
 - ❑ `(length '(1 2 3 4)) ; 4`
 - ❑ `(length '()) ; 0`

```

SemNome 16 - DrRacket
Eicheiro  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
SemNome 16  (define ...)
1 | #lang racket

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket, memory limit: 128 MB.
> (remove 3 '(1 2 3 4))
'(1 2 4)
> (reverse '(1 2 3 4))
'(4 3 2 1)
> (length '(1 2 3 4))
4
> (length '())
0
> |
Determine language from source 11:2
  
```

OPERAÇÕES COM LISTAS

- ❑ Retorna os **n** primeiros elementos
 - ❑ `(take '(1 2 3 4) 2); => '(1 2)`

- ❑ A partir do **n** elemento
 - ❑ `(drop '(1 2 3 4) 2); => '(3 4)`


```

SemNome 14 - DrRacket
Eicheiro  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
SemNome 14  (define ...)

Bemvindo a DrRacket, versão 5.3.6 [3m].
Linguagem: racket, memory limit: 128 MB.
> (take '(1 2 3 4) 2); => '(1 2)
'(1 2)
> (drop '(1 2 3 4) 2); => '(3 4)
'(3 4)
> |
Determine language from source 7:2
  
```


OPERAÇÕES COM LISTAS



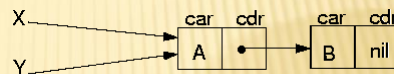
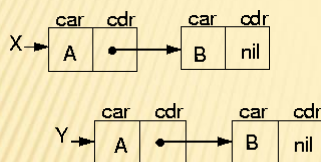
❑ retorna o 1º elemento de uma lista:

- ❑ `(car '(1 2 3))`
- ❑ `head` (cabeça)

❑ retorna **todos menos** o 1º elemento da lista:

- ❑ `(cdr '(1 2 3))`
- ❑ `tail` (cauda)

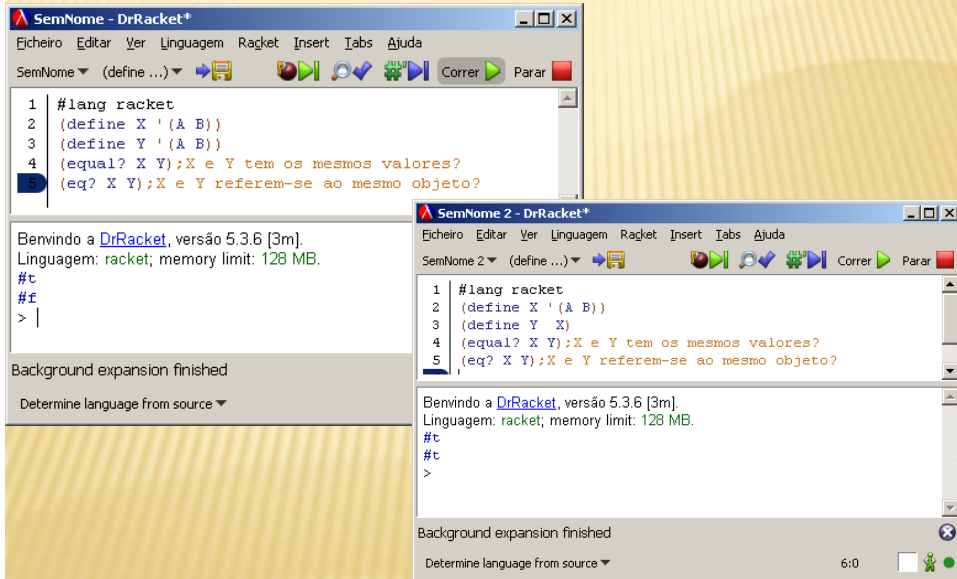
IGUALDADE DE LISTAS



```
#lang racket
(define X '(A B))
(define Y '(A B))
(equal? X Y); X e Y tem os
mesmos valores?
(eq? X Y); X e Y referem-se
ao mesmo objeto?
```

```
#lang racket
(define X '(A B))
(define Y X)
(equal? X Y); X e Y tem os
mesmos valores?
(eq? X Y); X e Y referem-se
ao mesmo objeto?
```

IGUALDADE DE LISTAS



EXERCÍCIO



- ❑ Crie a lista (1 2 3 4 5) e encontre:
 - ❑ 1) O primeiro elemento dela:
 - ❑ (car(list 1 2 3 4 5)) ou (car '(1 2 3 4 5))
 - ❑ 2) Todos menos o primeiro elemento dela:
 - ❑ (cdr(list 1 2 3 4 5)) ou (cdr '(1 2 3 4 5))
 - ❑ 3) O segundo elemento dela:
 - ❑ (car(cdr(list 1 2 3 4 5)))

OPERAÇÕES COM LISTAS

- retorna o 1º elemento:

- (first '(2 3))

- retorna o 2º elemento:

- (second '(2 3))

- retorna **todos menos** o 1º elemento da lista:

- (rest '(2 3))

EXERCÍCIOS



- Dada a lista (1 2 3 4 5), crie um programa que:

- 01) Mostre **quais** são os números **pares**.

- (filter even? '(1 2 3 4 5)) ;=> '(2 4)
 - (filter odd? '(1 2 3 4 5)) ;=> '(1 3 5)



- 02) Mostre **quantos** são os números **pares**.

- (count even? '(1 2 3 4 5)) ;=> 2
 - (count odd? '(1 2 3 4 5)) ;=> 3

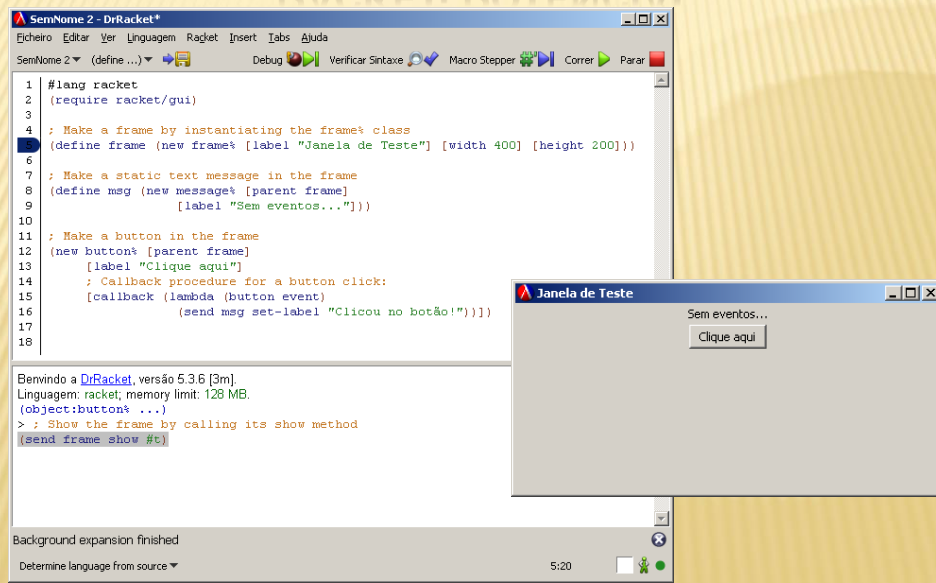
- 03) Faça o mesmo para os números **ímpares**.

EXERCÍCIOS



- ❑ Dada a lista (0 -1 10 2 -3 -4 5), crie um programa que:
 - ❑ 01) Mostre **quais** são os números **positivos**.
 - ❑ (filter positive? '(0 -1 10 2 -3 -4 5)) ; => '(10 2 5)
 - ❑ (filter negative? '(0 -1 10 2 -3 -4 5)) ; => '(-1 -3 -4)
 - ❑ 02) Mostre **quantos** são os números **positivos**.
 - ❑ (count positive? '(0 -1 10 2 -3 -4 5)) ; => 3
 - ❑ (count negative? '(0 -1 10 2 -3 -4 5)) ; => 3
 - ❑ 03) Faça o mesmo para os números **negativos**.

RACKET: POTENCIAL



RACKET: POTENCIAL

❑ Servidor web

- ❑ #lang web-server/insta
- ❑ (define (start request)
 - (response/xexpr
 - '(html
 - (head (title "My Blog"))
 - (body (h1 "Under construction"))



RACKET: POTENCIAL

❑ Escrever e ler em arquivo:

- ❑ ; importando biblioteca de I/O
- ❑ (require 2http/batch-io)
- ❑ ; escrevendo no arquivo
- ❑ (write-file "C:\\teste.txt" "Alice 25\nBob 40\n")
- ❑ ; recuperando o texto como uma lista de strings
- ❑ (read-lines "highscore.txt").






PostgreSQL

RACKET: POTENCIAL

Conexão com BD (PostresSQL):



```

db.rkt - DrRacket*
Eicheiro  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
db.rkt (define ...)  [Icons]  Verificar Sintaxe  Macro Stepper  Correr  Parar

1  #lang racket
2  (require db)
3  (define pgc
4    (postgresql-connect
5      #:user "postgres"
6      #:database "Racket"
7      #:server "localhost"
8      #:password "root"
9    )
10 )


Benvido a DrRacket, versão 5.3.6 [3m].
Linguagem: racket, memory limit: 128 MB.
> (query-exec pgc "insert into usuarios values (DEFAULT, 'Fulano')")
> (query-rows pgc "select * from usuarios")
'(#(7 "Fulano"))
> (query-exec pgc "insert into usuarios values (DEFAULT, 'Fulano2')")
> (query-rows pgc "select * from usuarios")
'(#(7 "Fulano") #(8 "Fulano2")))
>

Background expansion finished
Determine language from source
7:31
  
```



RACKET: POTENCIAL

Conexão com BD (MySQL):



```

dbMySQL.rkt - DrRacket
Eicheiro  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
dbMySQL.rkt (define ...)  [Icons]  Macro Stepper  Correr  Parar

1  #lang racket
2  (require db)
3  (define pgc
4    (mysql-connect
5      #:user "admin"
6      #:database "myschema"
7      #:server "localhost"
8      #:password "root"
9    )
10 )
11 ;(query-exec pgc "insert into usuarios values (DEFAULT, 'Nome')")
12 ;(query-rows pgc "select * from usuarios")

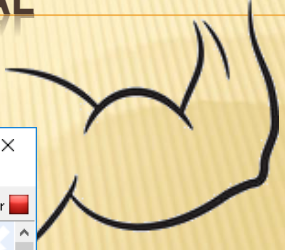
Benvido a DrRacket, versão 6.1.1 [3m].
Linguagem: racket, memory limit: 128 MB.
> (query-exec pgc "insert into usuarios values (DEFAULT, 'Nome')")
> (query-rows pgc "select * from usuarios")
'(#(1 "Nome") #(2 "Nome1") #(3 "Nome")))
>

Determine language from source
6:2  194.96 MB
  
```



RACKET: POTENCIAL

Conexão com BD (SQLite):



```
dbSQLite.rkt - DrRacket*
Ejchero  Editar  Ver  Linguagem  Racket  Insert  Tabs  Ajuda
dbSQLite.rkt (define ...) [Icons] Macro Stepper [Icons] Executar Parar

#lang racket
(require db)
(define s13c
  (sqlite3-connect
    #:database "D:/IFC/Dropbox/aulas/Programação Lógica e
Funcional/db/sqlite/plf.db"
  )
)
; (query-exec s13c "insert into usuarios values (null,'Nome')")
; (query-rows s13c "select * from usuarios")

Bemvindo a DrRacket, versão 6.5 [3m].
Linguagem: racket, with debugging; memory limit: 512 MB.
> (query-exec s13c "insert into usuarios values (null,'Nome')");
> (query-rows s13c "select * from usuarios");
'(#(1 "nome 01") #(2 "Nome"))
```

caminho até o
arquivo do banco