Red Programming Language

Guaracy Monteiro

9 de Outubro de 2017

Conteúdo

1 Listagens

2 Título

2.1 Tipagem

Apesar de não fazer distinção entre códigos e dados (homoiconicidade) os dados possuem tipos. Na realidade, existe uma grande variedade de tipos para facilitar a programação e a legibilidade. Para referenciar um determinado valor, não é necessário especificar o tipo do mesmo. Mesmo na declaração de uma função, a declaração dos tipo é opcional. Para a operação entre os valores, se não for possível a coerção ou a operação não for permitida, o programa irá parar informando um erro. Por exemplo:

Listagem 2.1: **Função com parâmetros dinâmicos**

```
soma: func[a b][
    a + b
]
print soma 1 2
print soma 1.5 2.6
print soma 2x4 4x4
print soma "no" "me"
```

```
3
4.1
6x8
*** Script Error: + does not allow string! for its value1 argument
*** Where: +
*** Stack: soma
```

Como podemos ver, a chamada ocorreu normalmente mas, no momento em que passamos duas strings como parâmetros, o programa mostrou um erro dizendo que a operação soma não aceitava tipos string. Vejamos os tipos aceitados por `+`:

Listagem 2.2: **Tipos aceitos pela soma (+)**

```
help +
```

```
USAGE:
     value1 + value2

DESCRIPTION:
     Returns the sum of the two values.
     + is an op! value.

ARGUMENTS:
     value1       [number! char! pair! tuple! vector! time! date!]
     value2       [number! char! pair! tuple! vector! time! date!]

RETURNS:
     [number! char! pair! tuple! vector! time! date!]
```

Da mesma forma, podemos especificar os tipos dos parâmetro aceitos por uma função que definimos. Por exemplo, se desejamos que aceite apenas valores inteiros e percentuais, declaramos como segue:

Listagem 2.3: **Função com parâmetros tipados**

```
1 soma: func[a [integer! percent! time!] b [integer! percent! time!]][
2     a + b
3 ]
4 print soma 1 2
5 print soma 10:05:22 0:5:40
6 print soma 2x3 6x7
```

```
3
10:11:02
*** Script Error: soma does not allow pair! for its a argument
*** Where: soma
*** Stack: soma
```

Quando ela foi chamada com um par, não chegou a executar a soma, acusando o erro no momento da chamada da função.

3 Examplos

3.1 Foo

Exemplo tirado do site de **Red** . O bloco abaixo foi informado manualmente pois o suporte a GTK3 ainda não foi desenvolvido pelo time principal de **Red** .

Também estou enchendo um pouco de lingüiça (com trema que fica mais legal) para ver o bloco quebrar um pouco mais abaixo.

Os blocos automáticos possuem a opção para não quebrar.

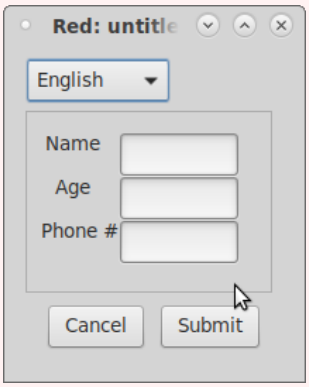
Listagem : **Exemplo em GTK3**

```
langs:  ["English" "French" "Português"]
labels: [
    ["Name" "Age" "Phone #" "Cancel" "Submit"]
    ["Nom" "Age" "Tél." "Abandon" "Envoyer"]
    ["Nome" "Idade" "Telefone" "Cancelar" "Enviar"]
]
set-lang: function [f event][
    root: f/parent
    condition: [all [face/text face/type <> 'drop-list]]

    list: collect [foreach-face/with root [keep face/text] condition]
    forall list [append clear list/1 labels/(f/selected)/(index? list)]

    foreach-face/with root [
        pads: any [metrics?/total face 'paddings 'x 0]
        prev: face/size/x
        face/size/x: pads + first size-text face
        face/offset/x: face/offset/x + ((prev - face/size/x) / 2)
    ][face/type = 'button]
]
view [
    style txt: text right 45
    drop-list data langs select 1 on-change :set-lang return
    group-box [
        txt "Name"  field return
        txt "Age"   field return
        txt "Phone" field
    ] return
```

```
    pad 15x0 button "Cancel" button "Submit"
]
```



3.2 Bar

Pela facilidade de trabalhar com org-mode, toda documentação foi desenvolvida com ele. A geração do pdf também poderia ter sido mas, não gostei exatamente do resultado da dobradinha código/resultado. Resolvi utilizar o tcolorbox para a apresentação da listagem dos fontes e apresentação dos resultados. Desenvovi normalmente e gerava o pdf para visualização com a apresentação padrão. O fonte gerado pelo org-mode para o latex era assim:

Listagem 3.1: **Bloco original gerado pelo org-mode**

```
\begin{listing}[htbp]
  \begin{minted}[linenos,firstnumber=1]{red}
   soma: func[a b][
       a + b
   ]
   print soma 1 2
   print soma 1.5 2.6
   print soma 2x4 4x4
   print soma "no" "me"
  \end{minted}
 \caption{Função com parâmetros dinâmicos}
\end{listing}
```

Eu precisava de um formato compatível com o tcolorbox e com as seguintes características. Chamaria uma macro `codeFromFile` com os seguintes parâmetros:

- linguagem para salientar a sintaxe
- nome do arquivo fonte para ser incluído (estaria da pasta `listings`)
- título para a listagem
- referência para entrar no índice
- parâmetros adicionais para o `minted` gerados pelo org-mode

No final teria algo assim:

Listagem 3.2: **Bloco desejado para o .tex**

```
\codeFromFile{red}{listings/p3.red}{Função com parâmetros
↪  tipados}{code3}{linenos,firstnumber=1}
```

O script (em **Red** é claro) para processar o arquivo .tex e efetuas as alterações ficou assim (bacalhau por enquanto):

Listagem 3.3: **Programa para alterar o arquivo .tex**

```
doc: read %teste.tex

z: copy ""
ip: 0

f: func[s][
    ip: ip + 1
    print rejoin ["gerando: programa" ip "..."]
    parse s [
        thru "minted}[" copy mint-style to "]"
        skip copy prg-ext thru "}"
        thru lf copy prg-src to "\end{minted}"
        thru "\caption{" copy rem to "}"
    ]
    pgs: rejoin ["listings/p" ip ".red"]
    write to file! pgs prg-src
    either rem/1 = #"*" [
        remove rem
        cmd: "\codeFromFileOnly"
    ][
        cmd: "\codeFromFile"
        insert prg-src "Red []^/"
        write %xyz.red prg-src
        out: copy ""
        call/shell/console/output "red xyz.red" out
        pgr: rejoin ["listings/p" ip ".res"]
        write to file! pgr out
    ]
    z: rejoin [cmd prg-ext "{" pgs "}{" rem "}{code" ip "}{"
    ↪  mint-style "}"]
]

rule: [
    any [
        to "\begin{listing}" p: insert (f p) remove thru
        ↪  "\end{listing}"
    ]
    to end
]
parse doc rule
write %gbm.tex doc
print "Gerando pdf..."
call/shell/console "pdflatex -shell-escape gbm.tex"
```

- **doc** : leitura do arquivo de teste (hardcoded por enquanto)
- **z** : conterá o exto que substituirá o bloco do arquivo original
- **ip** : índice utilizado na geração do nome do arquivo que conterá o fonte e o rótulo para o índice das listagens
- **f** : função que recebe um ponteiro para o ínicio do casamento da ocorrência analizando a entrada e:
  - armazenando estilo para minted em **mint-style**
  - armazenando tipo do programa em **prg-ext**
  - armazenando fonte do programa em **prg-src**
  - armazena rótulo da caixa em **rem**
  - se rótulo inicia com asterisco, não gera a saída do programa (solução encontrada no momento)
  - gera arquivo `xyz.red`, executa, captura a saída e gera arquivo `.red` com o resultado
  - gera texto para substituir a macro no arquivo .tex
- **rule** : se texto casa, chama **f** com ponteiro para o início do bloco, insere nova macro e elimina texto até o final do bloco
- grava em **gbm.tex** novo conteúdo
- executa **pdflatex** para gerar o pdf.

Melhorias para próxima versão:

- se md5 do fonte for diferente/não existe então proceder na gravação do fonte e saída, caso contrário eliminar estas duas etapas.
- programas gráficos, executar /no-wait e salvar screenshot com o nome do programa.png (criar macro adequada para o latex)

4 Fim

Terminou por enquanto. Algumas coisas ainda precisam de ajustes como o índice das listagens gerado pelo tcolorbox.