Representação de algorítmos

Fluxogramas e pseudo-código

Prof. Walmes Zeviani
walmes@ufpr.br

Laboratório de Estatística e Geoinformação
Departamento de Estatística
Universidade Federal do Paraná

Atualizado em 2018-08-01

Justificativas

- Documentar rotinas de forma independente de linguagem é importante para comunicação entre programadores.
- Também é funciona como uma "planta-baixa" para a implementação, reduzindo muitos erros de programação.

"If the map and the terrain disagree, trust the terrain."

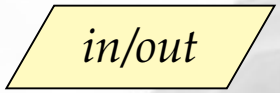
– Swiss Army Aphorism

Objetivos

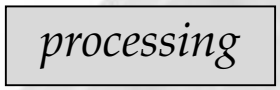
- Conhecer a simbologia de fluxogramas para representação de rotinas.
- Conhecer o vocabulário de pseudo código ou portugol.
- Exercitar a escrita e leitura com estas formas de representação.

Simbologia de fluxogramas

Início ou final de rotina.

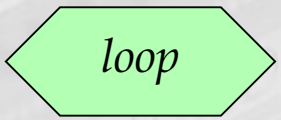


Entrada ou saída de dados.



Cálculos e procedimentos gerais.

Estrutura de controle condicional (decisão).

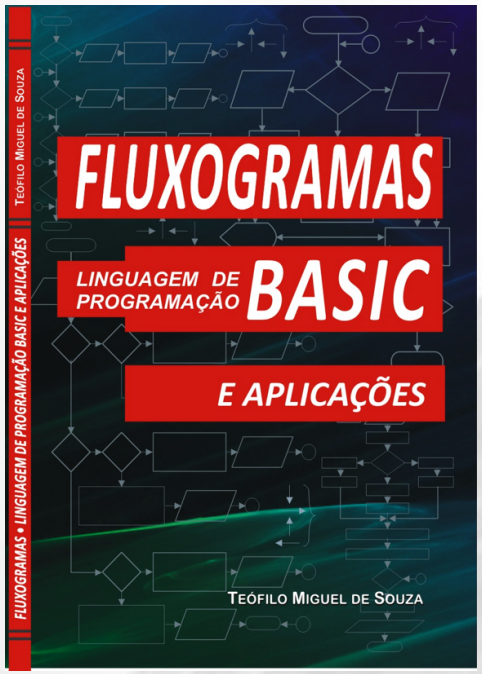


Estruturas de repetição (laços).

Fluxo de execução.

Conexão.

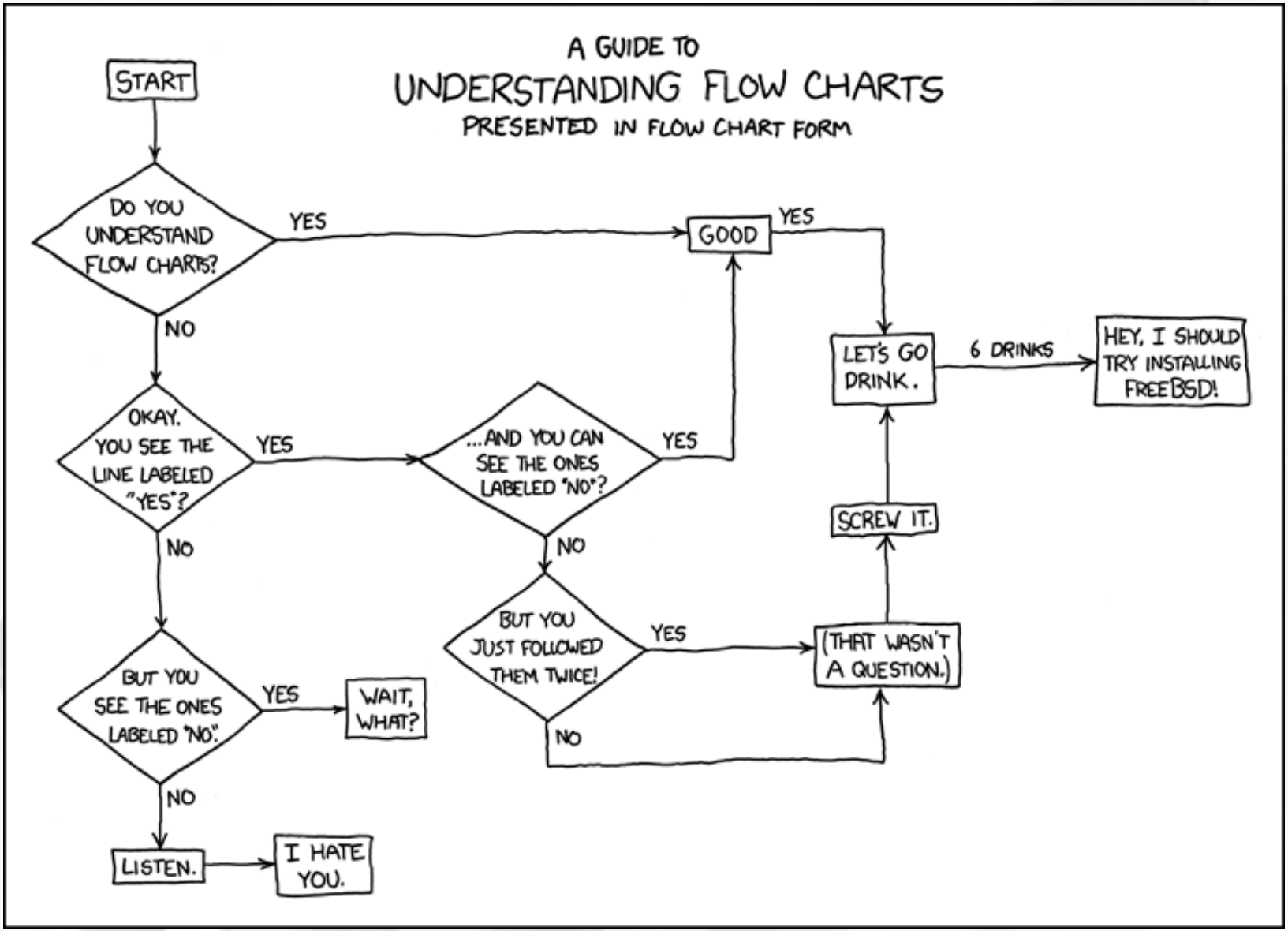
Chamada de subrotina.



SOUZA, T. M. **Fluxogramas, linguagem de programação basic e aplicações**. 1 ed. Ligthouse Editora. 2015.

▸ Comprar

Como entender fluxogramas



Exemplos da aplicação de fluxogramas

- http://www.inf.puc-rio.br/~inf1005/material/slides/backup/2010_2/tema02_algoritmos.pdf
- https://uomustansiriyah.edu.iq/media/lectures/5/5_2016_03_22!07_06_46_PM.pdf
- http://www2.ic.uff.br/~otton/graduacao/programacao/exemplos_pseudo_codigo.pdf
- http://producao.virtual.ufpb.br/books/edusantana/pseudocodigo-livro/livro/livro.chunked/ch02s04.html

Recursos para confecção de fluxogramas

- https://www.draw.io/.
- https://cloud.smartdraw.com/.
- https://www.lucidchart.com/.
- Inkscape.
- Tikz.
- Dia.
- Graphviz.
- Flowgorithm.
- https://code2flow.com/.
- Edpcs.
- LaTeX.

Vantagens e desvantagens dos fluxogramas

Vantagens

- Representação visual fácil de entender.
- Útil para comunicação com pessoas sem experiência em programação.
- Útil para prototipar e reduzir tempo de escrita de código.

Desvantagens

- Não existe uma correspondência lógicas rígida com comandos de linguagens.
- Portanto não serve como meio de documentação precisa para arquivamento (é complementar).

Recordando sobre escrita de funções R

- Dicas para nomes de função: ação $\rightarrow$ verbo.
- Argumentos de função: parâmetro $\neq$ argumento.
- Retorno da função: posição, conteúdo e impressão.
- Comunicação com usuário (▸ Advanced R).
  - Erros (*erros*).
  - Avisos (*warnings*).
  - Mensagens (*messages*).
- Tratamento de exceções.
- Formas de chamar funções.
- O uso de `...` em funções.

Benchmarking, profiling e debbuging

Benchmarking é o processo de avaliar a performance de operações específicas repetidamente.

Profiling é o processo de fazer o benchmark para cada instrução de uma função/rotina.

Debugging é o processo de buscar e resolver erros. Técnicas de debugging permitem avaliar a função passo a passo quando chamada.

https://bookdown.org/csgillespie/efficientR/.

Exemplo de aplicação

Benchmark

```
library("microbenchmark")

df <- data.frame(v = 1:4, name = letters[1:4])
microbenchmark(df[3, 2], df[3, "name"], df$name[3])


## Unit: microseconds
##           expr    min      lq     mean  median      uq     max neval cld
##       df[3, 2] 12.310 13.2240 13.78684 13.7430 14.2715  19.509   100   b
##  df[3, "name"] 12.313 13.4135 15.10646 13.9515 14.4335 115.277   100   b
##     df$name[3]  8.236  9.7755 10.36409 10.2830 10.6775  20.072   100  a
```

Exemplo de aplicação

Profiling

```
library("profvis")

profvis({
    x <- runif(10000000)
    z <- rpois(10000000, lambda = 5)
    x <- sort(x)
    z <- sort(z)
    mean(x)
    mean(z)
    median(x)
    median(z)
})
```

Exemplo de aplicação

Debugging

```
baskara <- function(a, b, c) {
    stopifnot(a != 0)
    delta <- b^2 - 4 * a * c
    if (delta >= 0) {
        den <- 2 * a
        sqrt_delta <- sqrt(delta)
        x <- (-b + c(-1, 1) * sqrt_delta)/den
    } else { x <- NULL }
    return(x)
}
# Entra no modo de debug.
debug(fun = baskara)
# Chama a função.
baskara(a = -3, b = 1, c = 2)
# Sai do modo de debug.
undebug(fun = baskara)
```

Próximo assunto

- Programação funcional.
- Família `*apply`.
- Pacote **purrr**.

Semana que vem

- Sabatina no Moodle.
- Aula com outro professor (talvez).