Introdução à programação
Conceitos e aplicação de Lógica

Vanderlei Júlio Debastiani (vanderleidebastiani@yahoo.com.br)

03 Março 2021

Índice de conteúdo

1 Introdução

Lógica tem grande importância no mundo atual. A lógica busca discutir o uso de raciocínio em qualquer atividade sendo também definida como estudo normativo, filosófico do raciocínio válido. Discutida principalmente nas disciplinas de filosofia, matemática e ciência da computação a lógica pode ser definida como uma sequência coerente de ideias ou o modo pelo qual se encadeiam naturalmente os acontecimentos.

Aplicação da lógica em computações pode ser vista como processos de raciocínio e simbolização formais objetivando a racionalidade e o desenvolvimento de técnicas que cooperem para soluções de um problema. É portanto uma técnica de encadear pensamentos para atingir um determinado objetivo. Os passos executados até atingir um objetivo podem ser definidos como uma sequência lógica.

Este texto trata dos conceitos básicos de lógica aplicada à programação, apresenta de maneira bastante resumida os principais conceitos de algoritmos, processamento de dados, operadores e estruturas de controle de fluxo. Os exemplos mostrados utilizam como base a linguagem R. A ideia é que o leitor seja capaz de reproduzir os resultados aqui contidos e exercitar os conceitos apresentados usando R. Os conceitos apresentados podem ser entendidos independentemente do conhecimento prévio de qualquer linguagem de programação.

2 Algoritmo

Em computação a lógica é usada na construção de algoritmos, estes que podem ser formalmente definidos com uma sequência finita de instrução que visam obter uma solução para um determinado tipo de problema. As definições dessas instruções devem ser claras e precisas, não podem ser subjetivas. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa, sendo independentes das linguagens de programação. Os fluxogramas (Figura 1) podem ser usado para representação esquemática de um processo ou algoritmo, onde as linhas de fluxo mostram a ordem das operação.

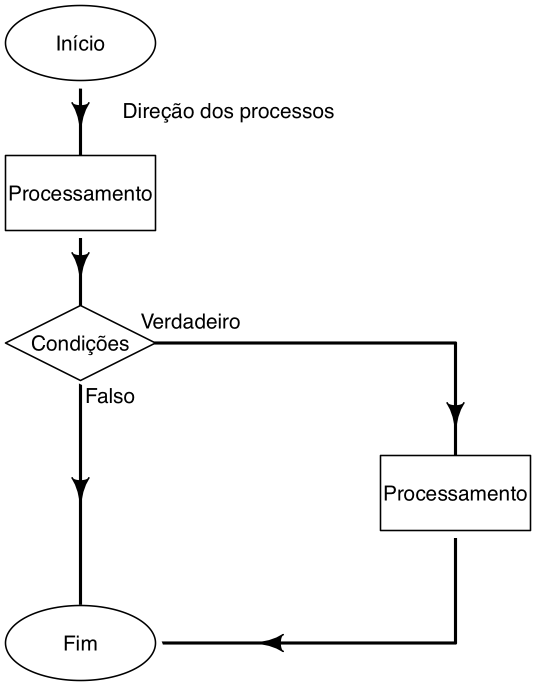

Figura 1: Principais elementos de um fluxograma. Existe uma padronização para a simbologia do fluxograma, apenas alguns tipos são mostrados.

Qualquer processo pode ser descrito por sequências lógicas. No fluxograma abaixo (Figura 2) é mostrado um processo simplificado de fazer café.

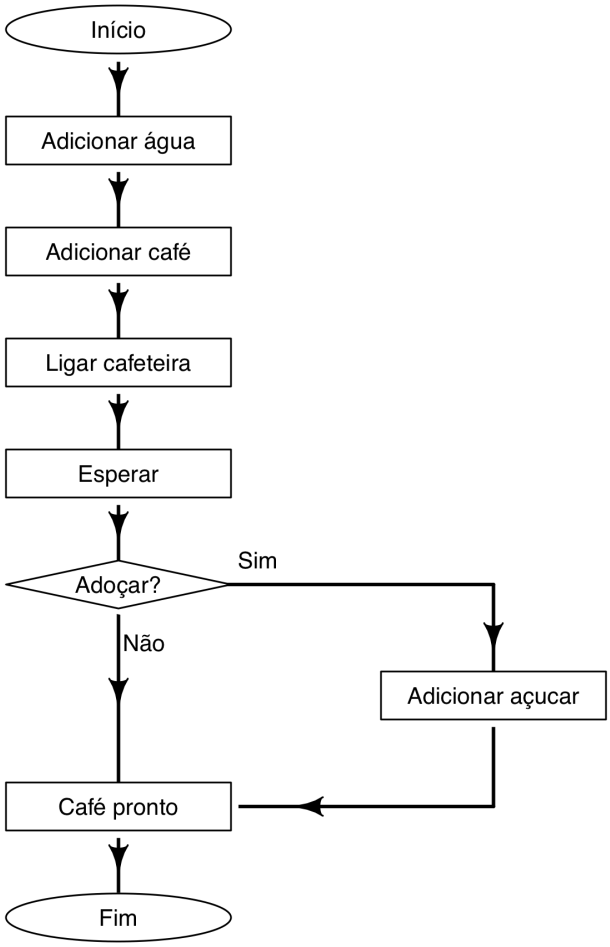

Figura 2: Fluxograma para fazer café.

2.1 Fases

A definição de um algoritmo pode ser divida e apresentada em quatro fases fundamentais:

- Objetivo do algoritmo - Definição do problema e objetivo do algoritmo;
- Entrada de dados – Dados de entrada do algoritmo;
- Processamento – Os procedimentos utilizados para chegar ao resultado final;
- Saída – Dados já processados.

2.2 Algoritmos e funções em R

A aplicação do conceito de algoritmo pode ser entendido também como uma função de realiza uma determinada tarefa. Na linguagem R as funções são declaradas com a palavra **function**. As funções podem também chamar outras funções, de modo que cada função realiza apenas uma parte do procedimento completo. Cada uma dessas funções tem um objetivo, seus argumentos de entrada (dados de entrada), uma ou várias

etapas de processamento e retorna os dados já processados. Estes dados já processados poderão ser utilizados em etapas intermediárias do próprio algoritmo e/ou utilizados como resultados finais do algoritmo.

3 Processamento de dados

O advento dos computadores dinamizou o tratamento das informações. Os computadores processam dados, estes são orientados por um conjunto de instruções e destinados a produzir resultados conforme previamente programadas. Processamento sistemático de dados com o objetivo de ordenar, classificar ou efetuar quaisquer transformações nos dados visando a obtenção de um determinado resultado.

3.1 Tipos de Dados

Os tipos de dados utilizados pelos comutadores são basicamente de três tipos:

- Numéricas - Armazenamento de números, que posteriormente poderão ser utilizados para cálculos.
- Caracteres - Armazenamento de conjunto de caracteres que não contenham números (literais).
- Lógicas - Armazenam de dados lógicos binário que representam os estados de Verdadeiro ou Falso.

3.1.1 Tipos de dados no R

A linguagem R suporta todos os tipos de dados descritos acina. Os tipos são denominados como numéricos (*numeric*), caracteres (*character*) ou lógicos (*logical*). O tipo caractere são escritos obrigatoriamente entre aspas duplas (**""**) ou simples (**”**). As dados lógicos são definidos com letras maiúsculas, sendo os verdadeiros como **TRUE** (abreviação **T**) e os falsos como **FALSE** (abreviação **F**). No R o valor zero (0) é considerado sempre falso (**FALSE**) e números diferentes de zero são considerados verdadeiros (**TRUE**).

3.2 Constantes e Variáveis

Variáveis e constantes são os elementos básicos que um programa manipula. Esses elementos são espaços reservado na memória do computador para armazenar um tipo determinado de dado. Esses elementos devem receber nomes para poderem ser referenciados e modificados quando necessário.

3.2.1 Constantes

As constante são valores fixos que não se modificam durante a execução de um programa.

3.2.2 Variáveis

As variáveis são objetos capazes de reter e representar um ou mais valores ou expressões. O conteúdo de uma variável pode se alterado ao longo da execução de um programa.

3.2.3 Variáveis em computação e estatística

O conceito da palavra *variável* em informática é menos restritiva que a utilizada em estatística. Em estatística normalmente variável é definida como um atributo, mensurável ou não, sujeito à variação quantitativa ou qualitativa. Já em informática variável é qualquer objeto situado na memória do computador que representa um valor ou expressão.

3.3 Expressões

Expressões são os meios fundamentais das etapas de processamento dos dados. As expressões combinam constantes, variáveis e operadores para calcular novos valores de acordo com as instruções. Após calculado o novo valor é retornado.

4 Operadores

Os operadores são meios pelo são realizadas operações matemáticas, como somas e multiplicações, e operações de comparação e lógicas. Em informática os operadores são praticamente os mesmos usados em matémática. Os operadores lógicos são bastante utilizados na tomada de decisões em algoritmos.

4.1 Operadores aritméticos

Os operadores aritméticos são os utilizados para obter resultados numéricos, como operações matemáticas simples (Tabela 1).

Tabela 1: Operadores aritméticos.

Operação	Símbolo
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponenciação	^

4.1.1 Hierarquia das operações aritméticas

Existe uma hierarquia na solução das operações aritméticas. Ela pode ser definida da seguinte maneira:

1. Parênteses;
2. Exponenciação;
3. Multiplicação e divisão;
4. Adição e subtração.

Alguns exemplos de operações aritméticas são mostrados na Tabela 2.

Tabela 2: Exemplo de operações aritméticas usadas em R.

Expressão	Resultado
2+2	4
5-2	3
2*4	8
9/3	3
3^3	27

4.2 Operadores relacionais

Os operadores relacionais são utilizados para comparar caracteres e números. Estes operadores sempre retornam valores lógicos (verdadeiro ou falso) (Tabela 3).

Tabela 3: Operadores relacionais.

Descrição	Símbolo matemático	Símbolo em R
Igual a	=	==
Diferente de	$\neq$	!=
Maior que	>	>

Descrição	Símbolo matemático	Símbolo em R
Menor que	$<$	<
Maior ou igual a	$\geq$	>=
Menor ou igual a	$\leq$	<=

Alguns exemplos de operações relacionais são mostrados na Tabela 4.

Tabela 4: Exemplo de operações relacionais usadas em R.

Expressão (sendo A = 2; B = 1 e C = 2)	Resultado
A==B	FALSE
A!=B	TRUE
A>B	TRUE
A<B	FALSE
A>=B	TRUE
A<=C	TRUE

4.3 Operadores lógicos

Os operadores lógicos servem para combinar resultados de expressões, retornando se o resultado final é verdadeiro ou falso. Os operadores lógicos são mostrados na Tabela 5.

Tabela 5: Operadores lógicos usados em R.

Descrição	Símbolo matemático	Símbolo em R
E - verdadeira se todas as condições forem verdadeiras	$\wedge$	& ou &&
OU - verdadeira se pelo menos uma condição for verdadeira	$\vee$	| ou ||
NÃO - inverte o valor da condição	$\neg$	!

Alguns exemplos de operações relacionais são mostrados na Tabela 6.

Tabela 6: Exemplo de operações lógicas usadas em R

Expressão (sendo A = 2; B = 1 e C = 2)	Resultado
(A==B) && (B<C)	FALSE
(A!=B) || (B<C)	TRUE
!(A==B)	TRUE
(A>B) && (B<C)	TRUE
(A<=B) || (C<B)	FALSE
!(A>=B)	FALSE

No linguagem R os operadores **&** e **|** fazem a operação elementar produzindo resultados de comprimento do operador mais longo. Já os operadores **&&** e **||** examinam apenas o primeiro elemento dos operadores e como resultado apenas um único valor lógico é retornado.

5 Estrutura de controle de fluxo

Estrutura de controle de fluxo são as ordem em que instruções e expressões são executadas ou avaliadas em programas de computador. As estruturas determinam no computador a sequência lógica proposta pelo algoritmo. Os tipos de estruturas disponíveis diferem de linguagem para linguagem. Em R as declarações de

controle de fluxo são realizadas por **if**, **else**, **while**, **repeat**, **for**, **break** e **next**. As funções **ifelse** e **switch** também podem ser utilizadas para controle de fluxo. Essas declarações podem ser usadas isoladamente ou em conjunto para executar o procedimento desejado.

5.1 Estrutura sequencial

A estrutura sequencial é uma estrutura de controle do fluxo que realiza um conjunto predeterminado de comandos de forma sequencial, de cima para baixo, na ordem em que foram declarados. Além de manter a sequência especificada não há nenhum desvioo fluxo do algoritmo (Figura 3).

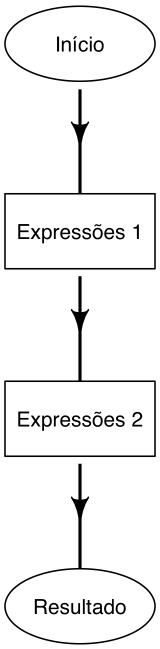

Figura 3: Estrutura de decisão sequencial.

Estrutura sequencial na linguagem R.

```
expressão1
expressão2
expressão3
```

Exemplo:

```
minha.var <- 10
minha.var
[1] 10
minha.var2 <- minha.var + 3
minha.var2
[1] 13
```

5.2 Estrutura de seleção

Estrutura de decisão (seleção ou desvio) é uma estrutura de desvio do fluxo de um algoritmo, e conduzem a estruturas de programas que não são totalmente sequenciais. As decisões lógicas tomadas em função dos

dados ou resultados anteriores determina quais ações são realizadas nas etapas posteriores. A seleção quase sempre é realizada de maniera binária, com condição verdadeira ou falsa. As principais estruturas de decisão são: “Se Então”, “Se então Senão” e “Caso Selecione”

5.2.1 Se Então (if)

A estrutura de decisão Se Então (**if**) vem acompanhada de uma determinada condição que se for satisfeita então execute determinado comando. Os comandos só podem ser executados caso a condição seja verdadeira (Figura 4).

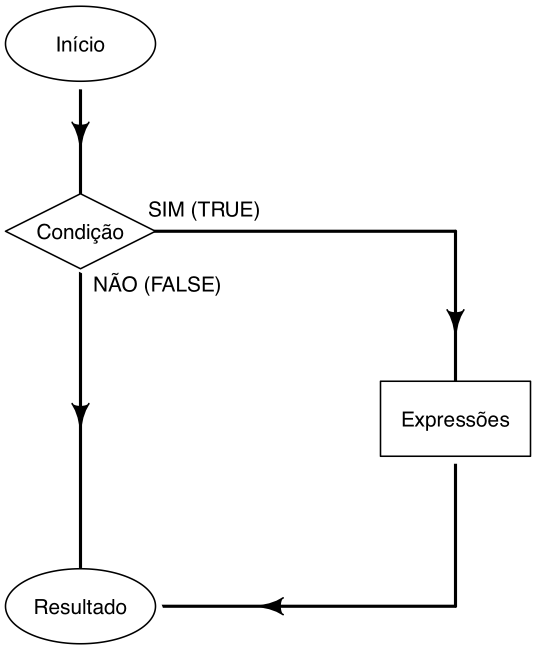

Figura 4: Estrutura de decisão Se Então.

Estrutura Se Então na linguagem R.

```
if(condição) {
    expressão
}
```

Exemplo:

```
minha.var <- 10
minha.var
[1] 10
if(minha.var == 10){
  print("minha.var é 10")
}
[1] "minha.var é 10"

minha.var <- 12
minha.var
```

```
[1] 12
if(minha.var == 10){
  print("minha.var é 10")
}
```

5.2.2 Se então Senão (if else)

A estrutura de decisão Se então Senão (**if else**) funciona de maneira simular a estrutura Se então (**if**). Neste caso a uma escolha é binária do que comando executar, sendo que sempre um comando será executado independente da condição. Caso a condição seja verdadeira (TRUE) o comando da condição será executado, caso contrário o comando da condição falsa (FALSE) será executado (Figura 5).

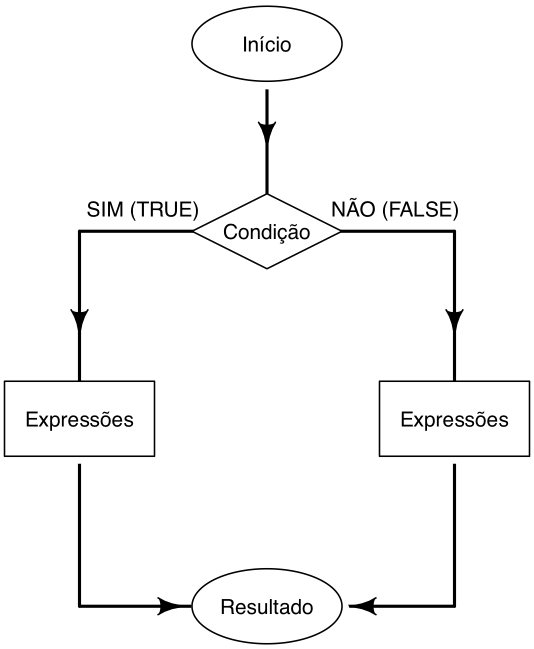

Figura 5: Estrutura de decisão Se Então Senão.

Estrutura Se então Senão na linguagem R.

```
if(condição) {
    expressão
} else {
    expressão alternativa
}
```

Exemplo:

```
minha.var <- -10
minha.var
[1] -10

if(minha.var>=0){
```

```
  print("minha.var é positiva ou zero")
} else {
  print("minha.var é negativa")
}
[1] "minha.var é negativa"
```

5.2.3 Caso Selecione (switch)

A estrutura de decisão Caso Selecione (**switch**) é utilizada para testar uma única expressão que produz um resultado, esse resultado é utilizado para selecionar uma opção (caso) entre diversas opções disponíveis (Figura 6).

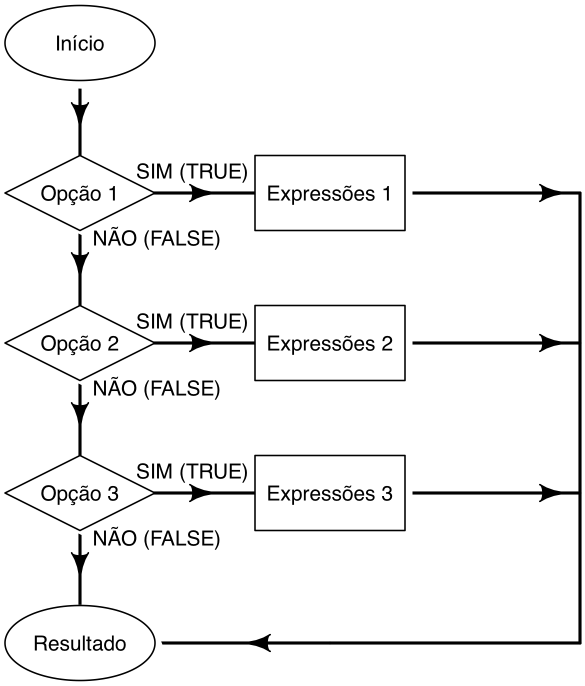

Figura 6: Estrutura de decisão caso selecione.

Estrutura Caso Selecione na linguagem R.

```
switch(Opção escolhida,
       Opção 1 = {Expressão da Opção 1},
       Opção 2 = {Expressão da Opção 2},
       Opção 3 = {Expressão da Opção 3})
```

Exemplo:

```
Op <- "Op3"
Op
[1] "Op3"
switch(Op,
         Op1 = print("Opção 1"),
         Op2 = print("Opção 2"),
```

```
        Op3 = print("Opção 3"))
[1] "Opção 3"

Op <- 2
Op
[1] 2
switch(Op,
        Op1 = print("Opção 1"),
        Op2 = print("Opção 2"),
        Op3 = print("Opção 3"))
[1] "Opção 2"
```

5.3 Estrutura de repetição

As estrutura de repetição são usadas para determinar que um conjunto de instruções ou comandos sejam executados um número definido ou indefinido de vezes ou enquanto um determinado estado prevalecer ou até que seja um estado seja alcançado. As principais estruturas de repetição são: “Enquanto”, “Repetir Até” e “Para Cada”.

5.3.1 Enquanto (while)

Na estrutura enquanto (**while**) o bloco de operações será executado enquanto a condição for verdadeira. O teste da condição será sempre realizado antes de qualquer operação. Enquanto a condição for verdadeira o processo se repete (Figura 7). O bloco executado deve contar com alguma opção que permita a condição inicial mudar de estado para que a operação não seja executada infinitamente.

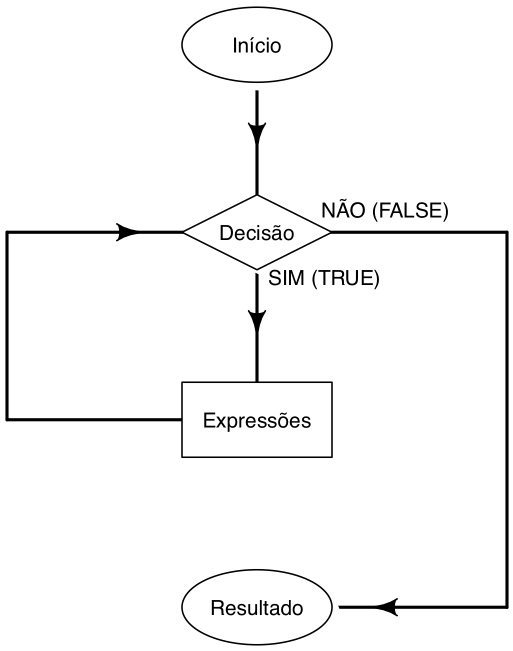

Figura 7: Estrutura de repetição Enquanto.

Estrutura Enquanto na linguagem R.

```
while(condição){
    expressão
}
```

Exemplo:

```
minha.var <- 10
minha.var
[1] 10

while(minha.var<50){
  minha.var <- minha.var+1
}
minha.var
[1] 50
```

5.3.2 Repetir Até (repeat)

A estrutura Repetir Até (**repeat**) permite que o bloco de operações será executado até que a condição seja satisfeita. O bloco será executado enquanto a condição for falsa (Figura 8). Da mesma forma que a estrutura Enquanto (**while**) o bloco executado deve contar com alguma opção que permita a condição mudar de estado para que a operação não seja executada infinitamente. No R a declaração **break** permite a interupção da execução da estrutura Repetir Até.

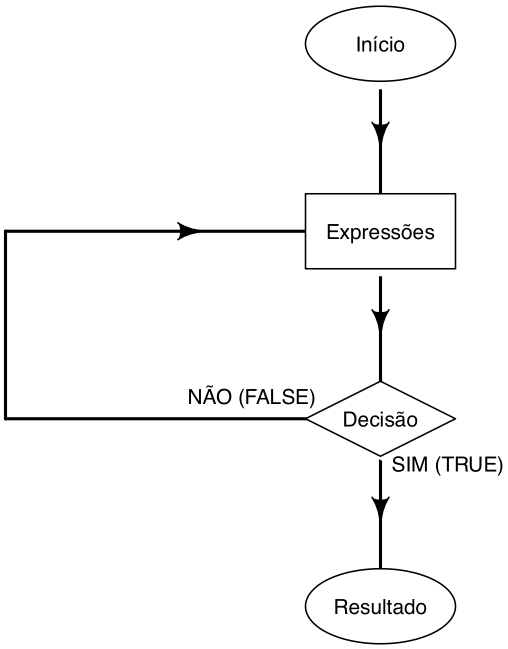

Figura 8: Estrutura de repetição Repetir.

Estrutura Repetir Até na linguagem R.

```
repeat {
    expressão
    if(condição) {
        break
    }
}
```

Exemplo:

```
minha.var <- 10
minha.var
[1] 10

repeat {
  minha.var <- minha.var+1
  if(minha.var>100) {
    break
  }
}

minha.var
[1] 101
```

5.3.3 Para Cada (for)

A estrutura Para Cada (**for**) é usada para repetir um bloco de operações um número conhecido de vezes, é denominada como laço ou loop. Um laço (**for**) é especificado em duas partes, a primeira delas um cabeçalho especificando as iterações e a seguda parte o código executado uma vez por iteração. O cabeçalho declara um contador explícito ou uma variável que é alterada a cada iteração. Esse procedimento permite ao programa saber qual iteração está sendo executada (Figura 9). No R a declaração **next** permite a interupção da iteração atual e avançar para a próxima iteração sem interupção completa do laço.

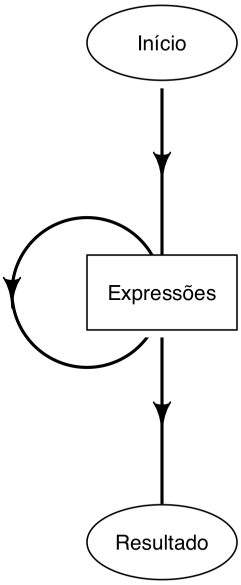

Figura 9: Estrutura de repetição Para Cada.

Estrutura Para Cada na linguagem R.

```
for(variável in sequência){
    expressão
}
```

Exemplo:

```
for(i in 1:5){
  print(i)
}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5

for(i in 1:5){
  if(i == 3){
    next
  }
  print(i)
}
[1] 1
[1] 2
[1] 4
[1] 5
```

```r
for(i in 1:5){
  if(i == 3){
    break
  }
  print(i)
}
[1] 1
[1] 2
```

6 Conclusão

O objetivo deste texto foi introduzir alguns conceitos básicos de programação utilizando os conceitos de lógica aplicada. Os principais operadores e estruturas de controle de fluxo foram mostradas e exemplificadas com base na linguagem R. Espero que este texto tenha sido útil e, por favor, avise-me se tiver dúvidas ou sugestões sobre este texto.

7 Mais informações

Outros textos e tutoriais sobre R podem ser encontrados em https://vanderleidebastiani.github.io/tutoriais.

8 Referências

- Forbellonne, A.L.V.; Eberspacher, H.F.; 2005. **Lógica de Programação – A Construção de Algoritmos e Estruturas de Dados**. São Paulo - Prentice Hall.
- Michaelis; 2018. **Dicionário Brasileiro da Língua Portuguesa**. http://michaelis.uol.com.br/busca?id=OKEKa
- Moraes, P.S.; 2000. **Lógica de programação**. Unicamp - Centro de Computação - DSC
- R Core Team; 2018. **R Language Definition**. https://cran.r-project.org/doc/manuals/R-lang.html