Capítulo

5 OPERAÇÕES CONDICIONAIS

Para você dizer em algum momento que sabe programar será necessário dominar dois tópicos básicos: operações condicionais e operações de repetição. Ambos são operações de controle de fluxo e são consideradas essenciais para programar qualquer procedimento, por mais simples que seja.

As operações condicionais estão relacionadas com tomada de decisão entre diferentes alternativas, isto é, escolha de uma opção entre duas ou mais possibilidades. Para ser capaz de compreender as operações condicionais é necessário entender pelo menos 4 tópicos: os operadores condicionais, operadores de comparação, operadores lógicos e operadores condicionais aninhados.

5.1 OPERADORES CONDICIONAIS

O primeiro tópico a ser entendido de operações condicionais é sobre os operadores condicionais, que é a estrutura geral das operações de condição. A maioria das linguagens de programação apresentam um operador condicional básico, que geralmente é encontrado em três versões básicas.

Operadores condicionas	
----------	----------
Se Condições: Tarefas	[se] [if]
----------	----------
Se Condições: $Tarefas_1$ **Caso Contrário:** $Tarefas_2$	[se] [caso contrário] [if] [else]
----------	----------
Se $Condições_1$: $Tarefas_1$ **Caso Contrário Se $Condições_2$:** $Tarefas_2$ **Caso Contrário:** $Tarefas_3$	[se] [caso contrário se] [caso contrário] [if] [else if] [else]
----------	----------

As decisões nos operadores condicionais são controladas pelas condições, as condições são construidas utilizando uma ou mais expressões que devem ser avaliadas em relação a sua veracidade. Se a condição for verdadeira, uma sequência de instruções (tarefas) deverão ser executadas, caso for falsa, isto é, caso contrário, uma sequência alternativa de instruções deverá ser executada.

5.2 OPERADORES DE COMPARAÇÃO

As expressões de condições geralmente são construidas utilizando os operadores de comparação e combinadas utilizando os operadores lógicos no caso de mais de uma expressão.

Operadores de comparação	
==	Igual
!=	Diferente
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a

Na sequência é apresentado o operador condicional `if`. Na Figura 3 é apresentado o fluxograma de ações referentes à decisão.

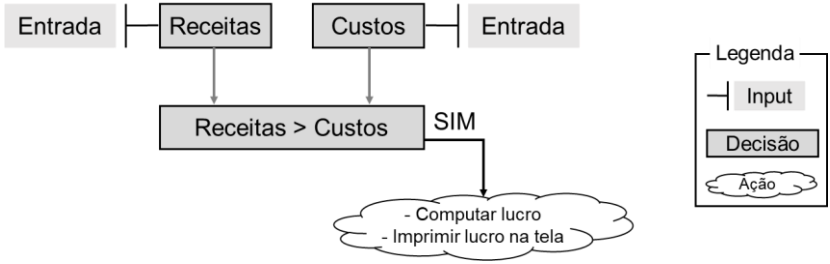

Figura 3. Decisão e ações em relação à avaliação de projetos para o exemplo `if`.

Observe que se a condição for falsa, isto é, se receitas forem menor ou igual aos custos, o código não executa instrução nenhuma, nem imprime na tela e nem computa o lucro (prejuízo no segundo exemplo).

```
# Operador condicional if
receitas <- 7500
custos <- 5000
if(receitas > custos){ # imprime o lucro na tela
  lucro <- receitas - custos
  cat("O lucro foi de R$", lucro)
}
## O lucro foi de R$ 2500
receitas <- 4500
custos <- 5000
if(receitas > custos){ # nenhuma tarefa é executada
  lucro <- receitas - custos
  cat("O lucro foi de R$", lucro)
}
##
```

O código será aperfeiçoado incluindo a instrução `else` (caso contrário) e as tarefas relacionadas a ela. Outro aperfeiçoamento será o encapsulamento do código em uma função, isso para evitar a repetição de código como apresentado acima. O fluxograma é apresentado na Figura 4 e o código é apresentado a seguir.

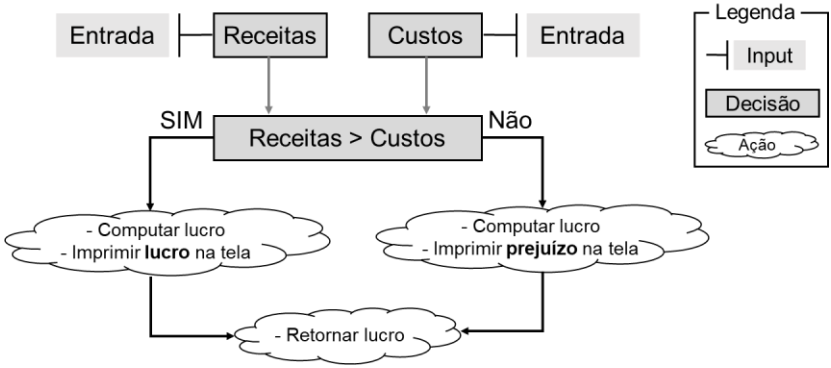

Figura 4. Decisão e ações em relação à avaliação de projetos para o exemplo `if-else`.

O fluxograma da Figura 4 apresenta uma sequência de instruções para o caso do projeto operar no prejuízo (Receitas não são maiores que custos), além disso o retorno da função também é representado no fluxograma.

```
# Operador condicional if-else
aval_econ <- function(receitas, custos){ # avaliação econômica
  if(receitas > custos){
    lucro <- receitas - custos
    cat("O lucro foi de R$", lucro, "\n")
  } else{
    lucro <- receitas - custos
    cat("O prejuízo foi de R$", abs(lucro), "\n")
  }
  return(lucro)
}
aval_econ(7500, 5000)
## O lucro foi de R$ 2500
## [1] 2500
aval_econ(4500, 5000)
## O prejuizo foi de R$ 500
## [1] -500
```

Observe que no bloco `else`, foi incluido a função `abs` no objeto de lucro, a função `abs` retornar o valor absoluto de um número, isto é, se o valor é negativo passa a ser positivo. Isso foi feito porque no texto explicativo já acompanha a palavra prejuízo, e assim já indicando ser um valor negativo.

É possivel apresentar mais um aperfeiçoamento, observe que a avaliação econômica de um projeto ainda pode apresentar um terceiro cenário, além dos dois já apresentados, de lucro e prejuízo. Seria o cenário de nem lucro e nem prejuízo, isto é, elas por elas, receitas iguais aos custos. O fluxograma é apresentado na Figura 5.

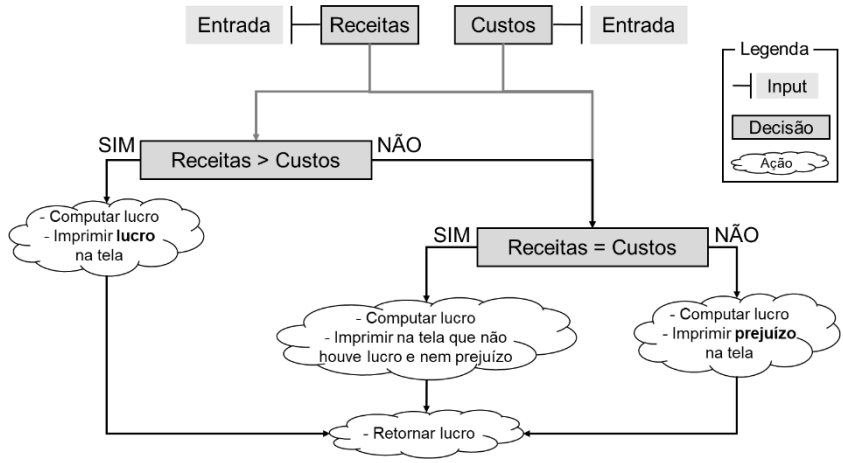

Figura 5. Decisão e ações em relação à avaliação de projetos para o exemplo `if-else if-else`.

Os operadores condicionais funcionam realmente como um processo de tomada de decisão, se a resposta para primeira pergunta (Receitas são maiores do que Custos?) for sim, as tarefas são executadas e o processo caminha para seu encerramento, isto é, para o retorno do resultado (valor de lucro). Observe que não será necessário fazer a segunda pergunta (Receitas são iguais aos Custos?) referente à segunda decisão.

```
# Operador condicional if-else if-else
aval_econ <- function(receitas, custos){ # avaliação econômica
  if(receitas > custos){
    valor <- receitas - custos
    cat("O lucro foi de R$", valor, "\n")
  } else if(receitas == custos){
    valor <- receitas - custos
    cat("Não apresentou lucro e nem prejuízo \n")
  } else{
    valor <- receitas - custos
    cat("O prejuizo foi de R$", abs(valor), "\n")
  }
  return(valor)
}
meu_projeto <- aval_econ(6000, 3000)
## O lucro foi de R$ 3000
meu_projeto
## [1] 3000
nosso_projeto <- aval_econ(5000, 5000)
## Não apresentou lucro e nem prejuízo
nosso_projeto
## [1] 0
seu_projeto <- aval_econ(4000, 6000)
## O prejuizo foi de R$ 2000
seu_projeto
## [1] -2000
```

5.3 OPERADORES LÓGICOS

Os operadores lógicos principais são o ou e e, representados pelos símbolos "|" e "&", respectivamente. O operador "&" pode ser considerado mais restritivo, uma vez que só retorna verdadeiro se todas as expressões de condição forem verdadeiras. Já o operador "|" é considerado mais permissivo, pois basta uma expressão ser verdadeira para toda a condição

retornar verdadeiro no final. A escolha do operador, obviamente vai depender do problema em questão. Ainda existe a negação lógica, representada pelo símbolo "!", que inverte o resultado de uma expressão condicional, por exemplo, se o resultado for verdadeiro, é então convertido em falso e vice-versa. Também existe o operador `xor` (ou exclusivo), que retorna verdadeiro se ambos os valores de entrada forem diferentes entre si, e retorna falso se forem iguais.

Operadores lógicos			
e "&"	ou "|"	ou exlusivo "xour"	Negação "!"
V e V = V	V ou V = V	xor(V, V) = F	!V = F
V e F = F	V ou F = V	xor(V, F) = V	!F = V
F e V = F	F ou V = V	xor(F, V) = V	
F e F = F	F ou F = F	xor(F, F) = F	

O exemplo a seguir demonstra o uso dos operadores lógicos, o objetivo é identificar qual será a quantidade de área plantada no ano seguinte baseando-se na performance do ano atual. Se a relação receita / custo do projeto for menor um, isto é, obteção de prejuízo, não será feita expansão de plantio, deve-se plantar exatamente a quantidade de área plantada do ano atual; se a relação receita / custo for entre 1 e 1,5, será feita expansão de plantio de 20%; se a relação for entre 1,5 e 2, a expansão será de 50%; mas caso a relação for superior a 2, a expansão será de 100% e a área de plantio será dobrada.

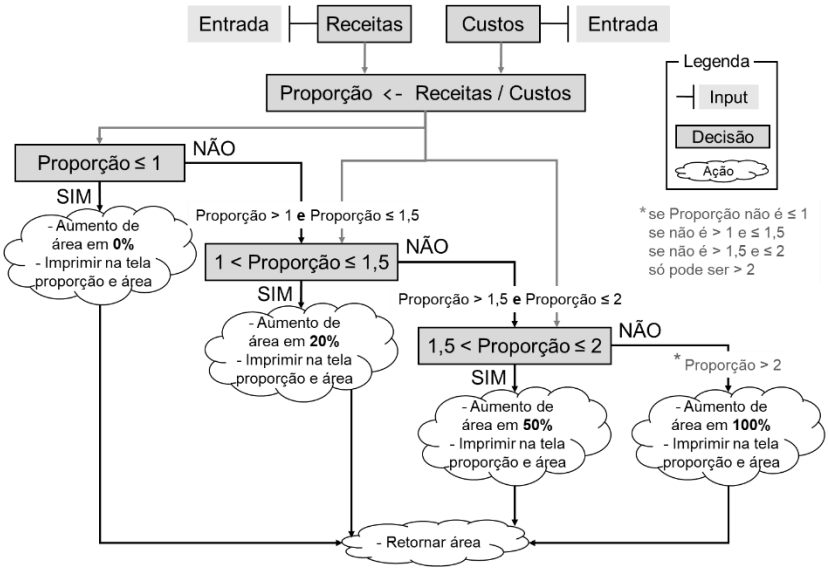

Figura 6. Decisão e ações em relação à quantidade de área de plantio para o exemplo `if-else` `if-else`, com auxílio de operadores lógicos.

```r
area_plantio <- function(receitas, custos, area){
  prop <- receitas / custos # proporção receitas / custos
  if(prop <= 1){
    area_seg <- area # aumento de 0%
    cat(paste("Proporção Receita/Custo:", round(prop, 1),
              "; Área de plantio:", area_seg, "ha\n"))
  }else if(prop > 1 & prop <= 1.5){
    area_seg <- area * 1.2 # aumento de 20%
    cat(paste("Proporção Receita/Custo:", round(prop, 1),
              "; Área de plantio:", area_seg, "ha\n"))
  }else if(prop > 1.5 & prop <= 2){
    area_seg <- area * 1.5 # aumento de 50%
    cat(paste("Proporção Receita/Custo:", round(prop, 1),
              "; Área de plantio:", area_seg, "ha\n"))
  }else{
    area_seg <- area * 2 # aumento de 100%
    cat(paste("Proporção Receita/Custo:", round(prop, 1),
              "; Área de plantio:", area_seg, "ha\n"))
  }
  return(area_seg)
}
receitas <- 5000 # R$
custos <- 5500 # R$
area_atual <- 200 # ha
area_plantio(receitas, custos, area_atual)
## Proporção Receita/Custo: 0.9 ; Área de plantio: 200 ha
## [1] 200
custos <- 3600 # R$
area_plantio(receitas, custos, area_atual)
## Proporção Receita/Custo: 1.4 ; Área de plantio: 240 ha
## [1] 240
area_plantio(receitas, 2800, area_atual)
## Proporção Receita/Custo: 1.8 ; Área de plantio: 300 ha
## [1] 300
area_plantio(receitas, 2400, area_atual)
## Proporção Receita/Custo: 2.1 ; Área de plantio: 400 ha
## [1] 400
```

5.4 OPERADORES CONDICIONAIS ANINHADOS

Os operadores lógicos podem ser substituídos por operadores condicionais aninhados (ou encaixados). Os operadores condicionais aninhados, são basicamente, um operador condicional dentro do outro, e as vezes vale a pena recorrer para essas estruturas de condição por questões de simplificação. Com os condicionais aninhados o código pode ficar mais simples de implementar e entender.

O próximo exemplo demostra o uso dos condicionais aninhados. O fluxograma abaixo será implementado utilizando a linguagem de programação R.

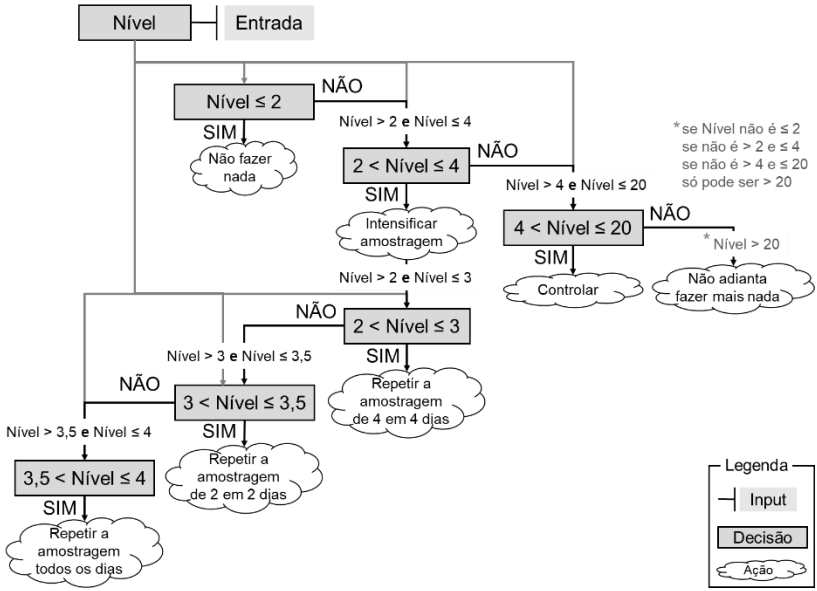

Figura 3. Decisões em relação ao nível de infestação para controle de pragas.

```
controle_pragas <- function(nivel){
  if(nivel <= 2){ # (1) esperar
    cat("Sem danos econômicos: NÃO TENTAR CONTROLAR.\n")
    cat("Manter programação normal de controle de pragas.\n")
  } else if(nivel > 2 & nivel <= 4) { # (2) intensificar amostragem
    cat("Sem danos econômicos: NÃO TENTAR CONTROLAR.\n")
    cat("Mas intensificar a amostragem.\n")
    if(nivel > 2 & nivel <= 3){ # amostragem opção 1
      cat("Repetir a amostragem de 4 em 4 dias.\n")
    }else if(nivel > 3 & nivel <= 3.5){ # amostragem opção 2
      cat("Repetir a amostragem de 2 em 2 dias.\n")
    }else if(nivel > 3.5 & nivel <= 4){ # amostragem opção 3
      cat("Repetir a amostragem todos os dias.\n")
    }
  }else if(nivel > 4 & nivel <= 20){ # (3) ação
    cat("Ação!!! TENTAR CONTROLAR.\n")
  }else{ # (4) já era
    cat("Não adianta fazer nada: NÃO TENTAR CONTROLAR.\n")
    cat("Os dados econômicos já são elevados demais.\n")
  }
}
controle_pragas(1.75)
## Sem danos econômicos: NÃO TENTAR CONTROLAR.
## Manter programação normal de controle de pragas.
controle_pragas(3.1)
## Sem danos econômicos: NÃO TENTAR CONTROLAR.
## Mas intensificar a amostragem.
## Repetir a amostragem de 2 em 2 dias.
controle_pragas(4.6)
## Ação!!! TENTAR CONTROLAR.
controle_pragas(21.2)
## Não adianta fazer nada: NÃO TENTAR CONTROLAR.
## Os dados econômicos já são elevados demais.
```