Programação Estruturada

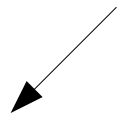
Prof. Rodrigo Hausen

http://progest.compscinet.org

Condicionais e Laços

RELEMBRANDO - AULA PASSADA

- Definindo Funções



tipo de retorno (saída)

```
tipo nomeFuncao(tipo par1, tipo par2) {
  ________
  ________
  ________
}
```

parâmetros (entrada)

FUNÇÕES em C - exercícios

- Congruência de Zeller

Seja ***K*** os dois últimos algarismos de um ano, ***J*** os demais algarismos, ***q*** o dia do mês e ***m*** o mês no formato: mar=3, abr=4, ..., dez=12, jan=13, fev=14.

A fórmula abaixo dá o dia da semana

$$\left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor + 5J \right) \bmod 7$$

onde 0 = sáb, 1 = dom, 2 = seg, ...

FUNÇÕES em C - exercícios

Usando a congruência de Zeller, crie uma função diaDaSemana que recebe o dia, mês (de 1 a 12) e ano (com todos os algarismos) e calcula o dia da semana onde 1=dom, 2=seg, ..., 7=sáb.

FUNÇÕES em C - exercícios

```
int diaDaSemana(int dia, int mes, int ano)
{
  int K,J,q,m,h;
  K = ano % 100;
  J = ano / 100;
  q = dia;
  m = mes >= 3 ? mes : mes+12;
  h = (q + 13*(m+1)/5 + K + K/4 +
       J/4 + 5*J) % 7;
  return h > 0 ? h : 7;
}
```

FUNÇÕES – boas práticas

- ANTES de começar a programar, certifique-se de ter entendido BEM a descrição do problema.

- Suas funções devem se comportar EXATAMENTE como descrito.

- P. ex.: se não está escrito que uma função deve imprimir algo, então ela NÃO DEVE imprimir nada!

FUNÇÕES – boas práticas

- Via de regra, cada função deve **fazer apenas uma coisa, e fazê-la bem**

- P. ex. se não está escrito que uma função vai fazer entrada de valores do usuário, ela **não deve** ler a entrada do usuário.

- Escreva funções **curtas**. Se o código de uma função estiver muito comprido, provavelmente é necessário dividi-la

FUNÇÕES – boas práticas

- Documente suas funções! Minimamente, você deve colocar um comentário antes de cada função, descrevendo:

  * O que ela faz. A primeira frase é crucial.

  * Significado dos parâmetros e valores válidos para eles.

  * Significado do valor de retorno da função.

FUNÇÕES – documentação

```
/**
 * Cálculo do dia da semana.
 * Usa a congruência de Zeller.
 * Parâmetros:
 *  dia: dia do mês
 *  mês: mês do ano (1=jan, 2=fev, ...)
 *  ano: ano completo no calendário
 *       gregoriano
 * Retorna: dia da semana no formato
 * 1=dom, 2=seg, 3=ter, ..., 7=sáb
 */
int diaDaSemana(int dia, int mes, int ano)
{
  …
```

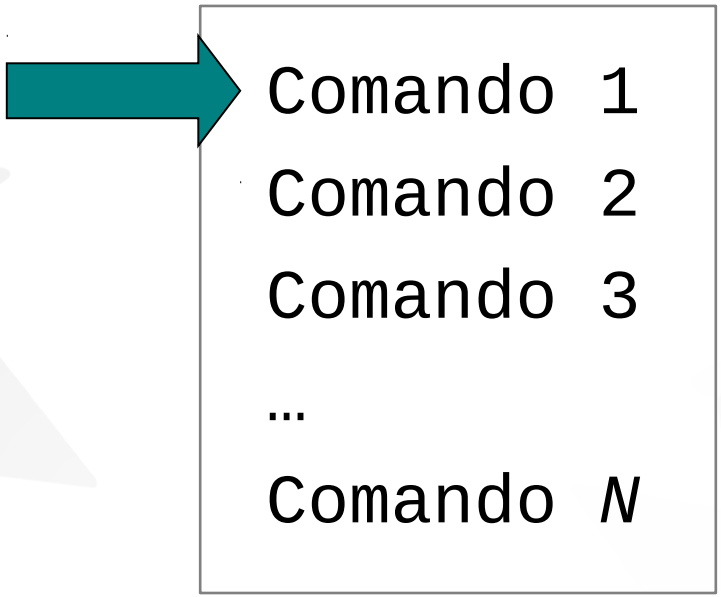
Estrutura sequencial

- Estrutura padrão em toda forma de algoritmo.
- Conjunto de comandos que serão executados em sequência linear, de cima para baixo.

```
Comando 1
Comando 2
Comando 3
…
Comando N
```

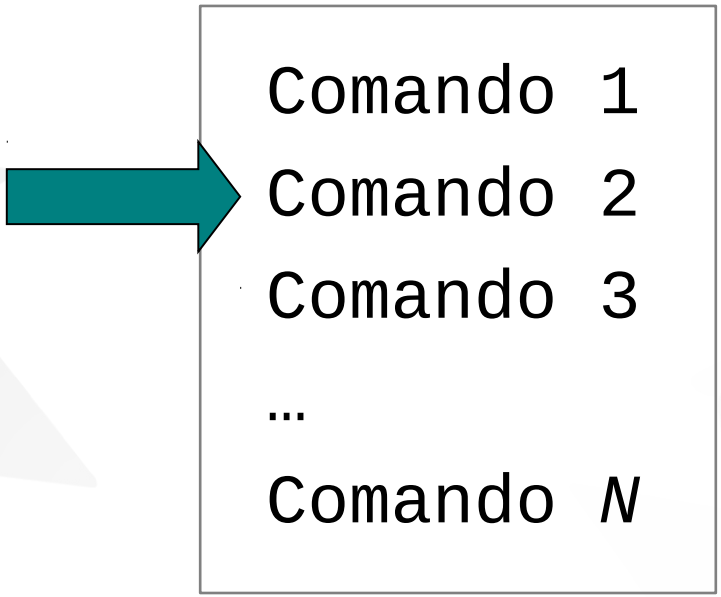
Estrutura sequencial

- Estrutura padrão em toda forma de algoritmo.
- Conjunto de comandos que serão executados em sequência linear, de cima para baixo.



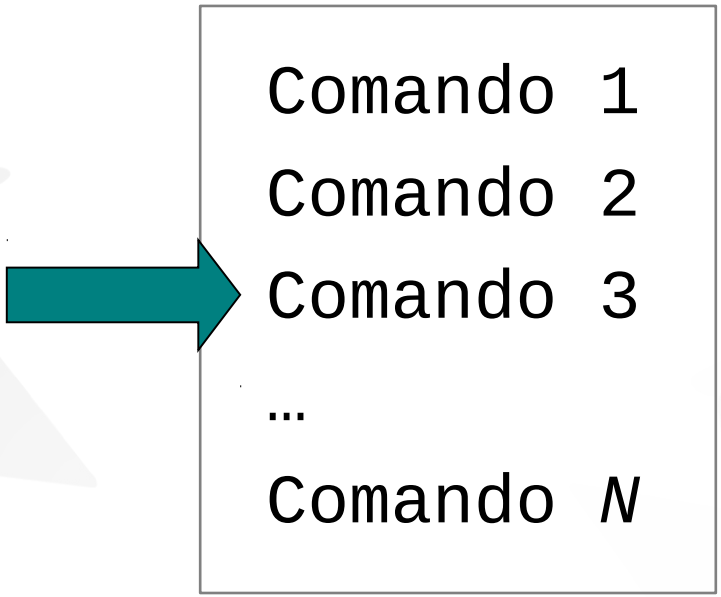
Estrutura sequencial

- Estrutura padrão em toda forma de algoritmo.
- Conjunto de comandos que serão executados em sequência linear, de cima para baixo.



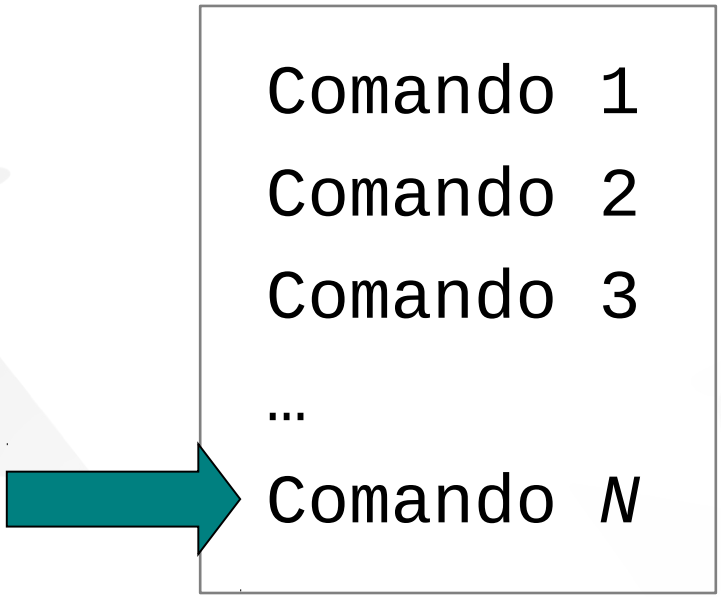
Estrutura sequencial

- Estrutura padrão em toda forma de algoritmo.
- Conjunto de comandos que serão executados em sequência linear, de cima para baixo.



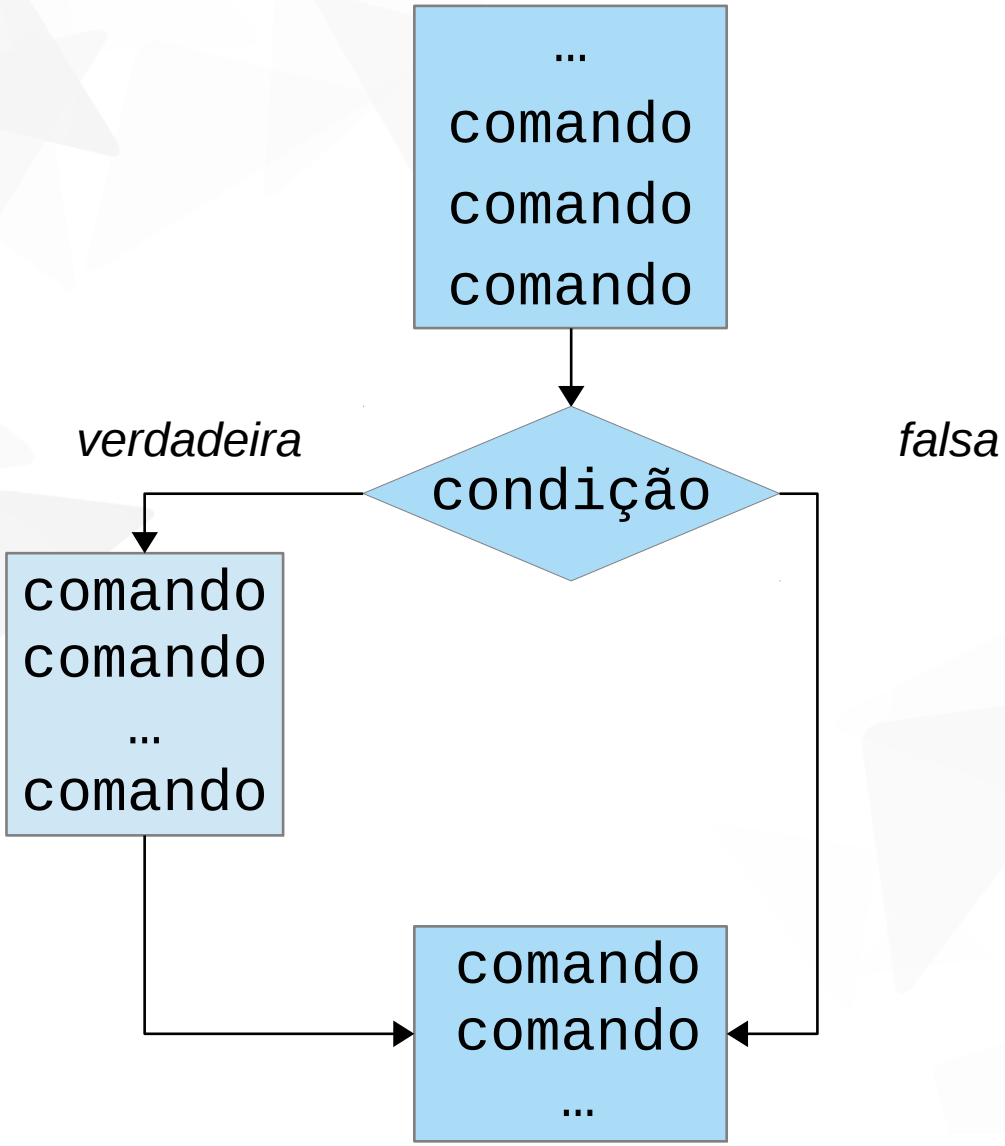
Estrutura sequencial

- Estrutura padrão em toda forma de algoritmo.
- Conjunto de comandos que serão executados em sequência linear, de cima para baixo.



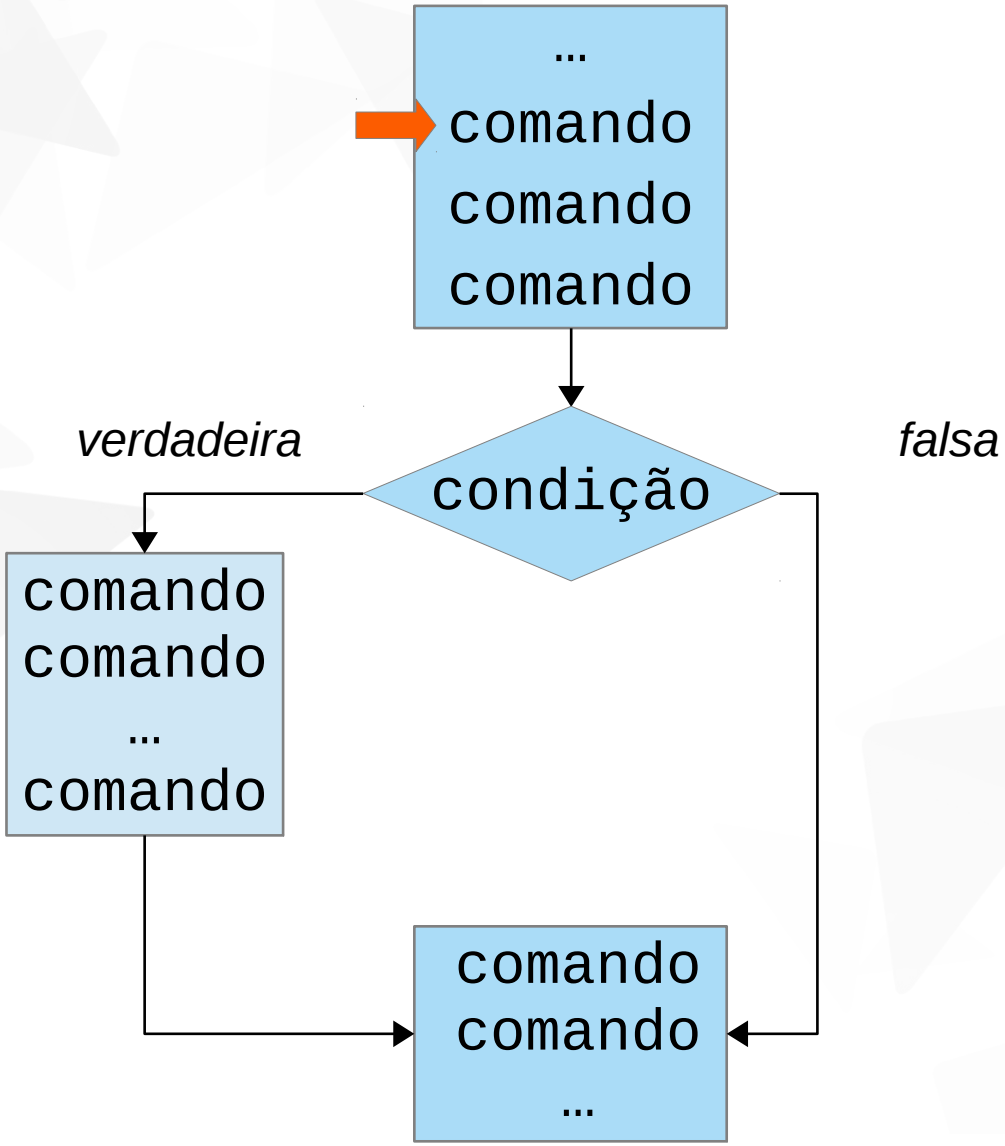
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



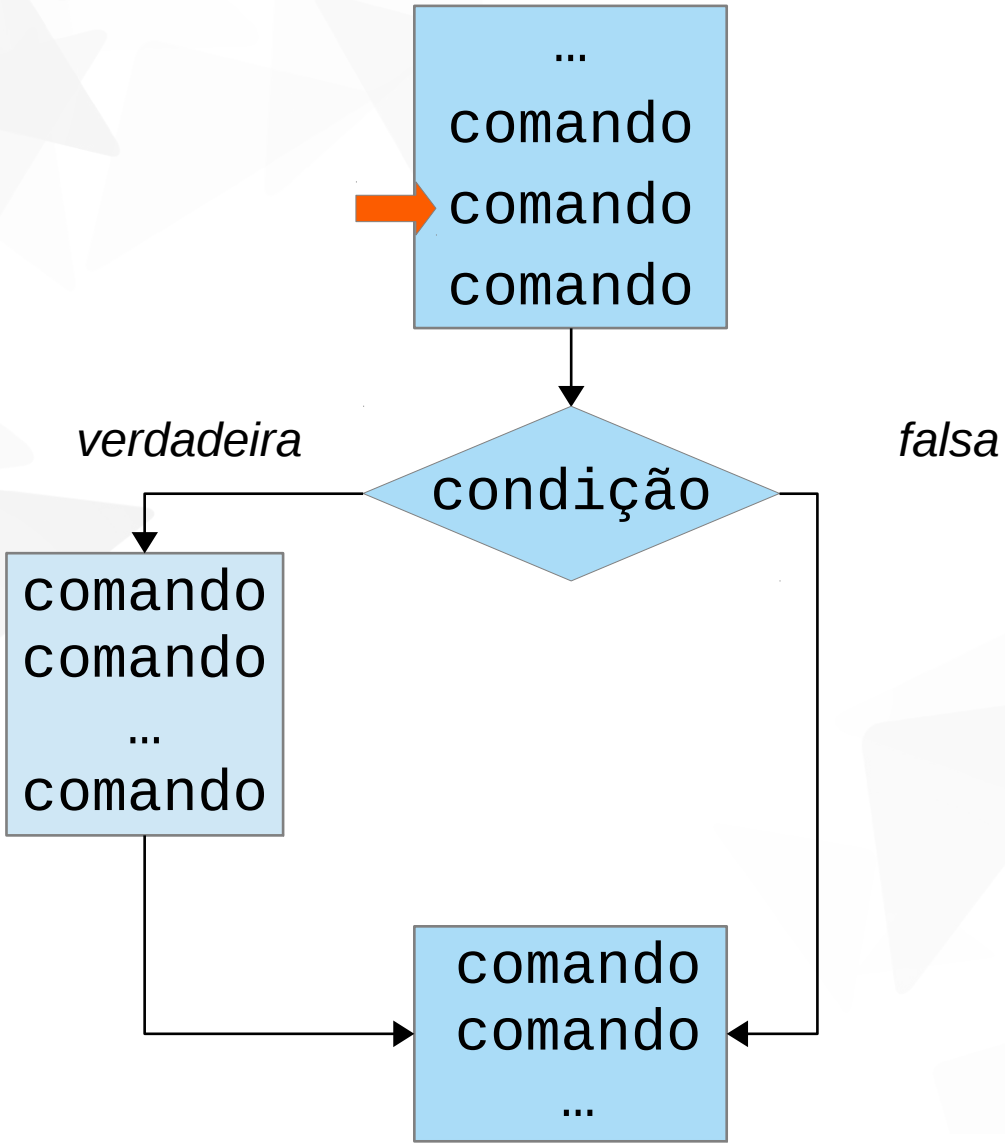
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



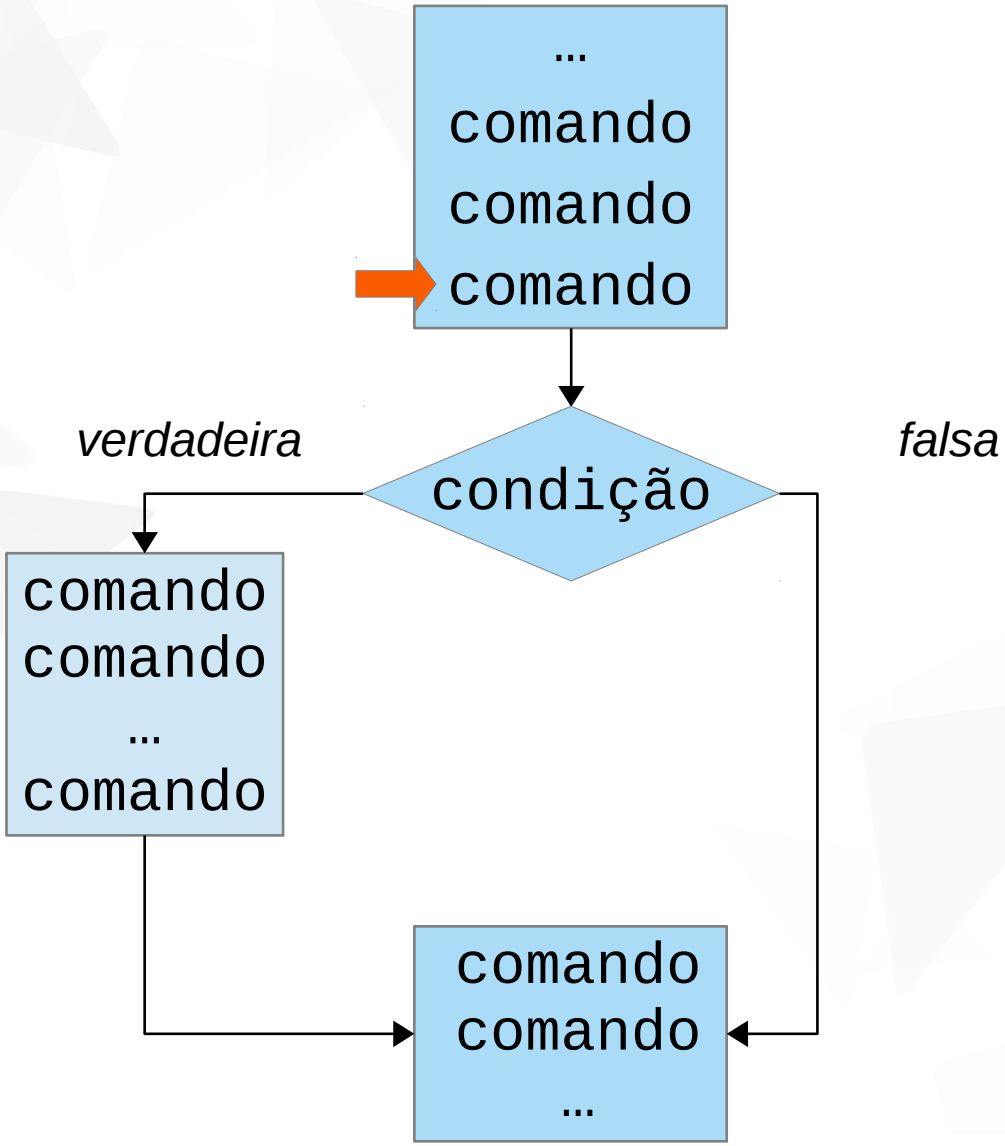
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



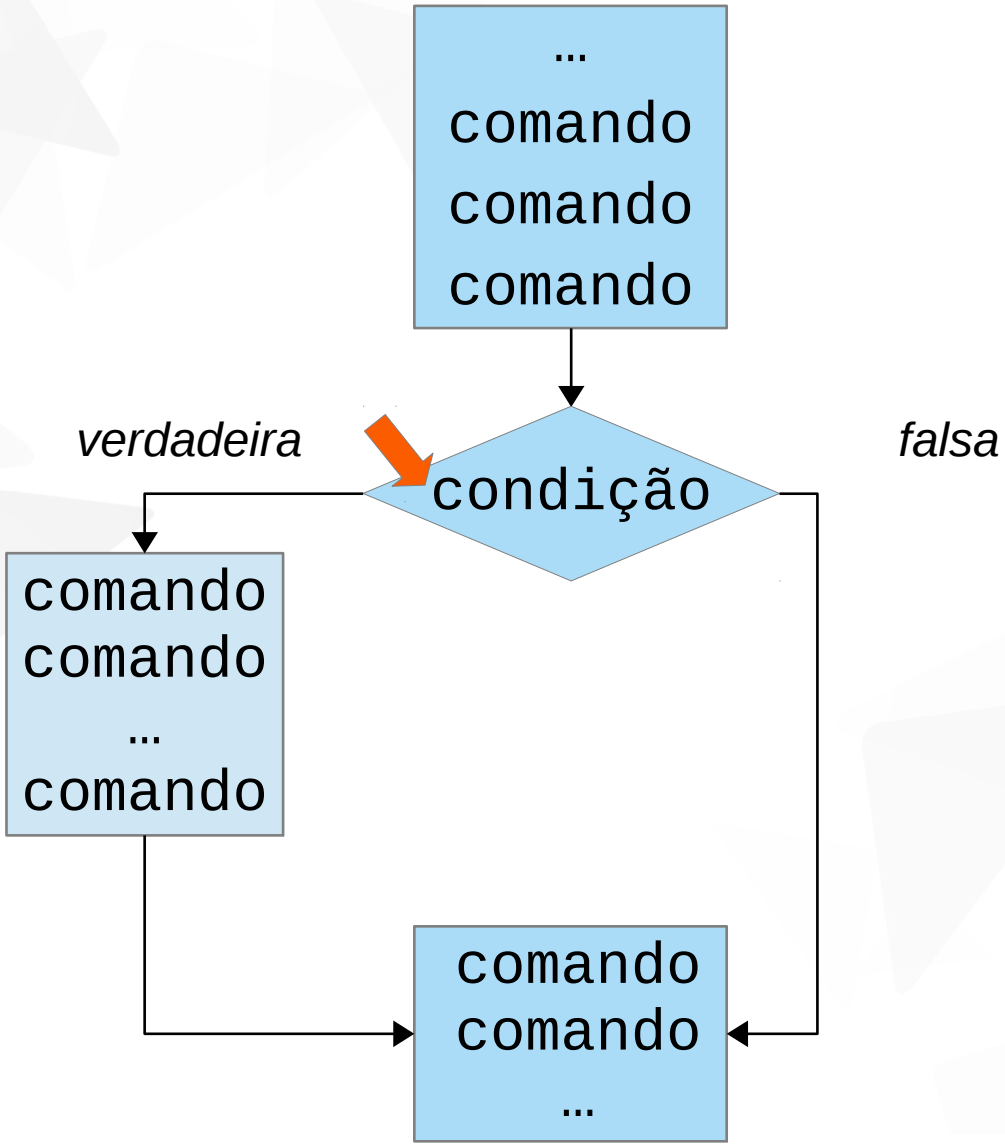
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



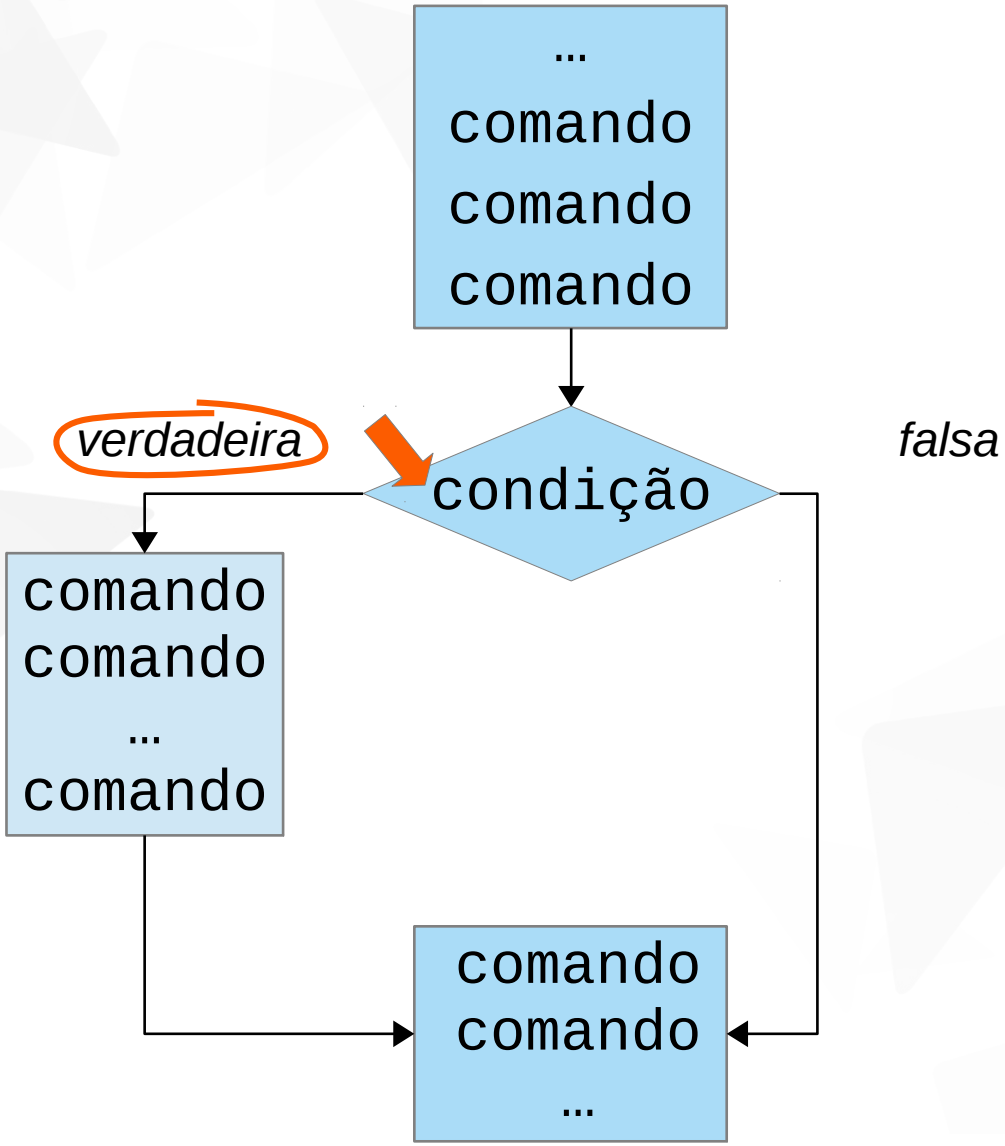
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



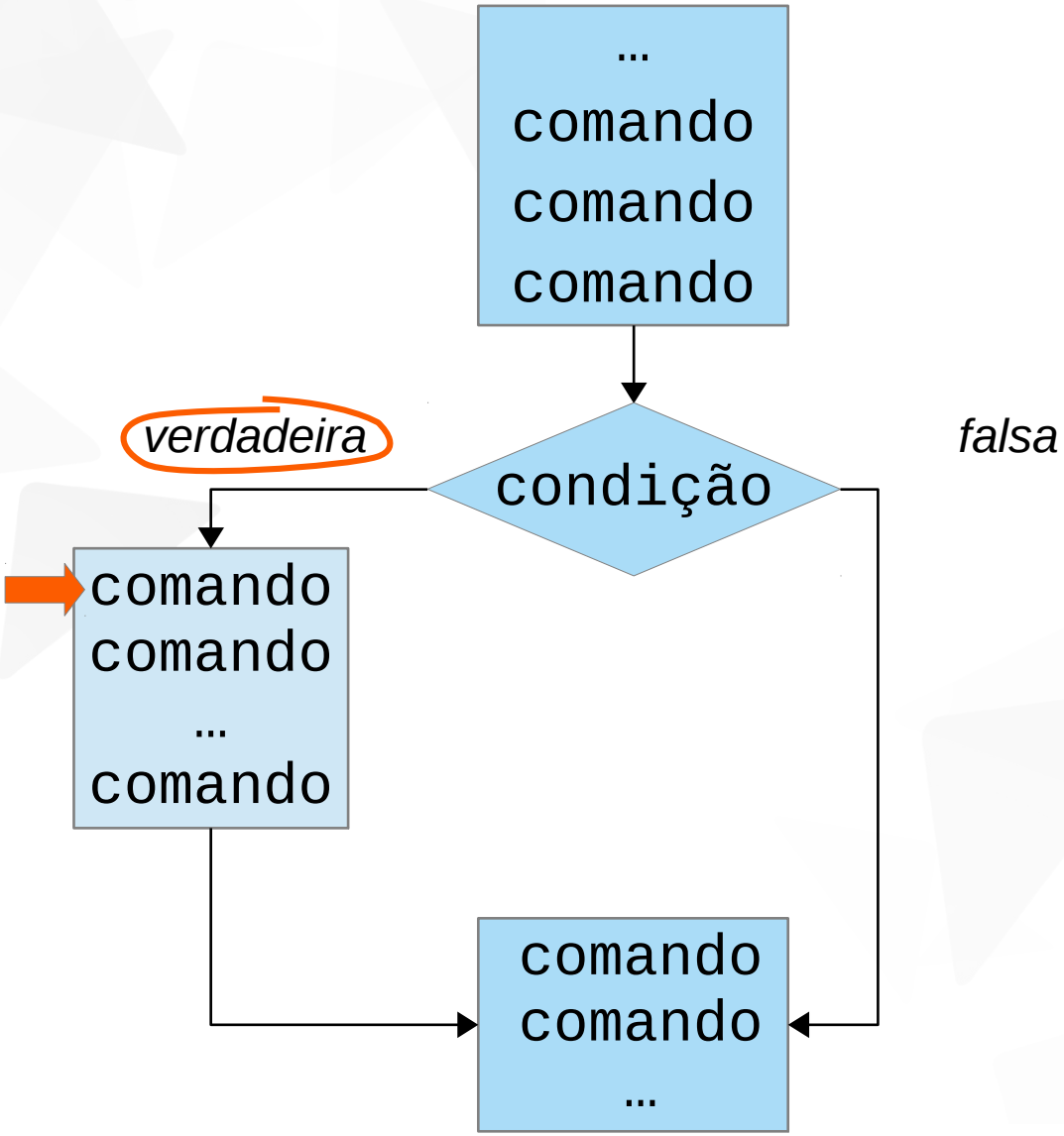
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



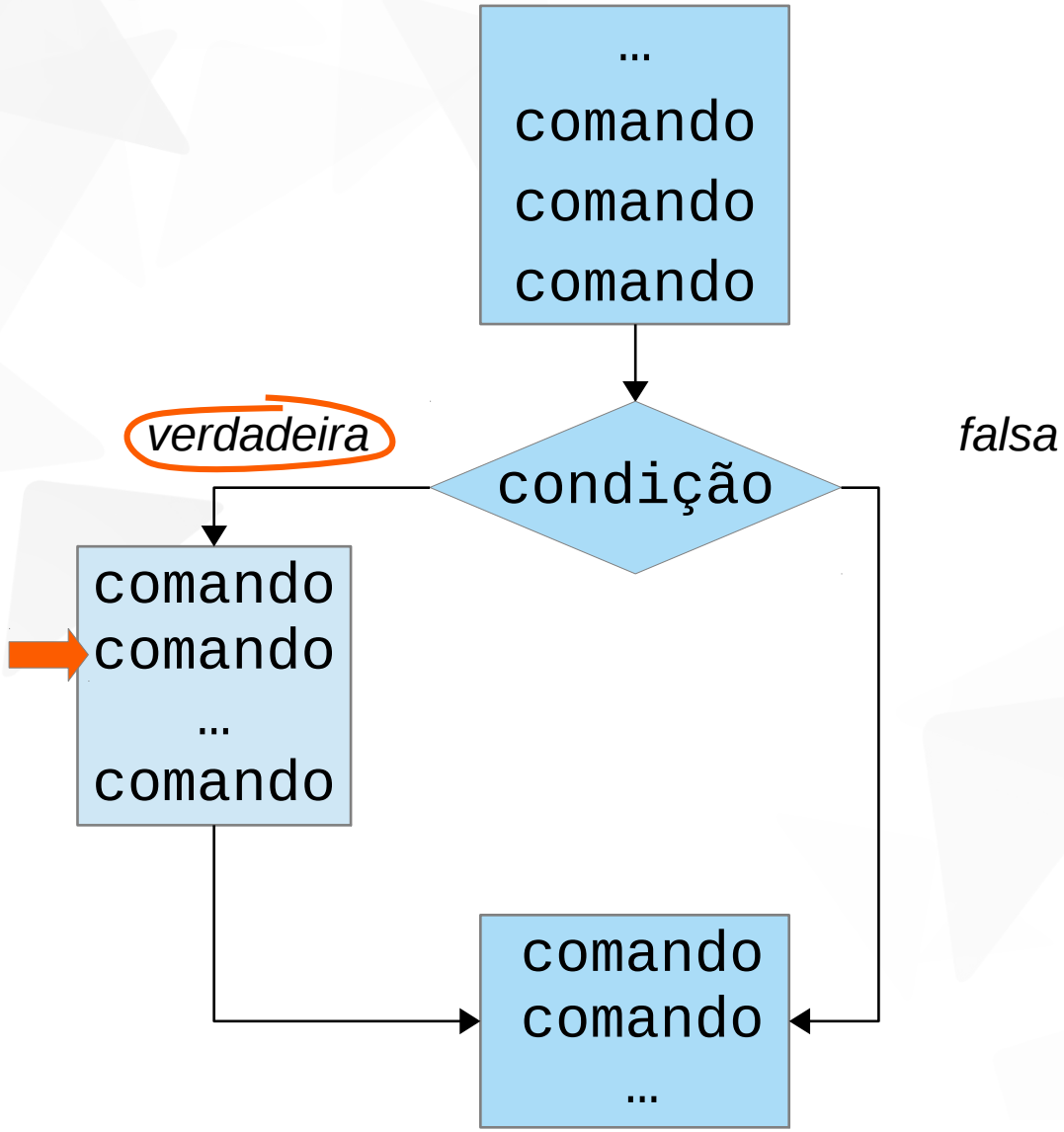
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



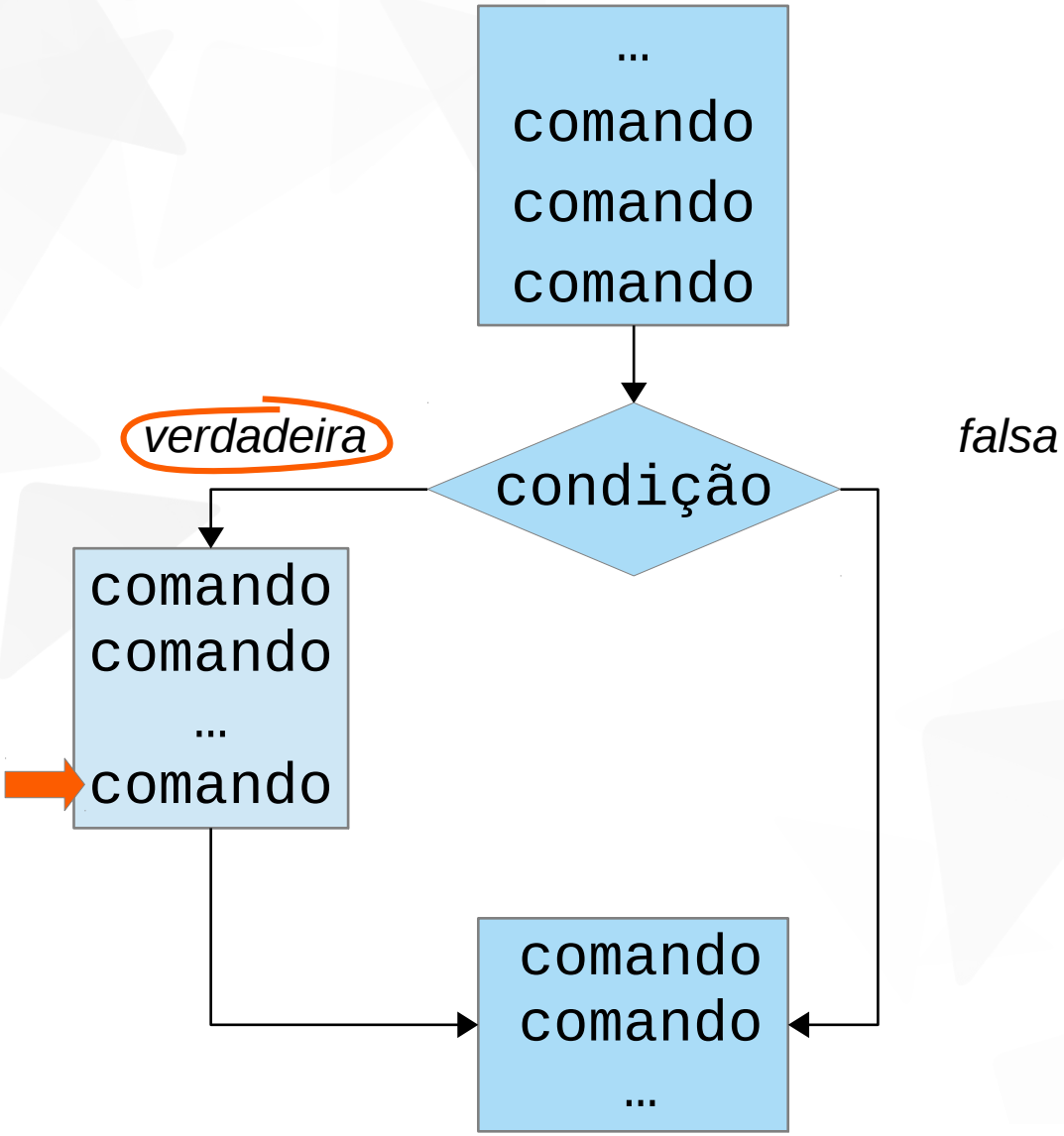
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



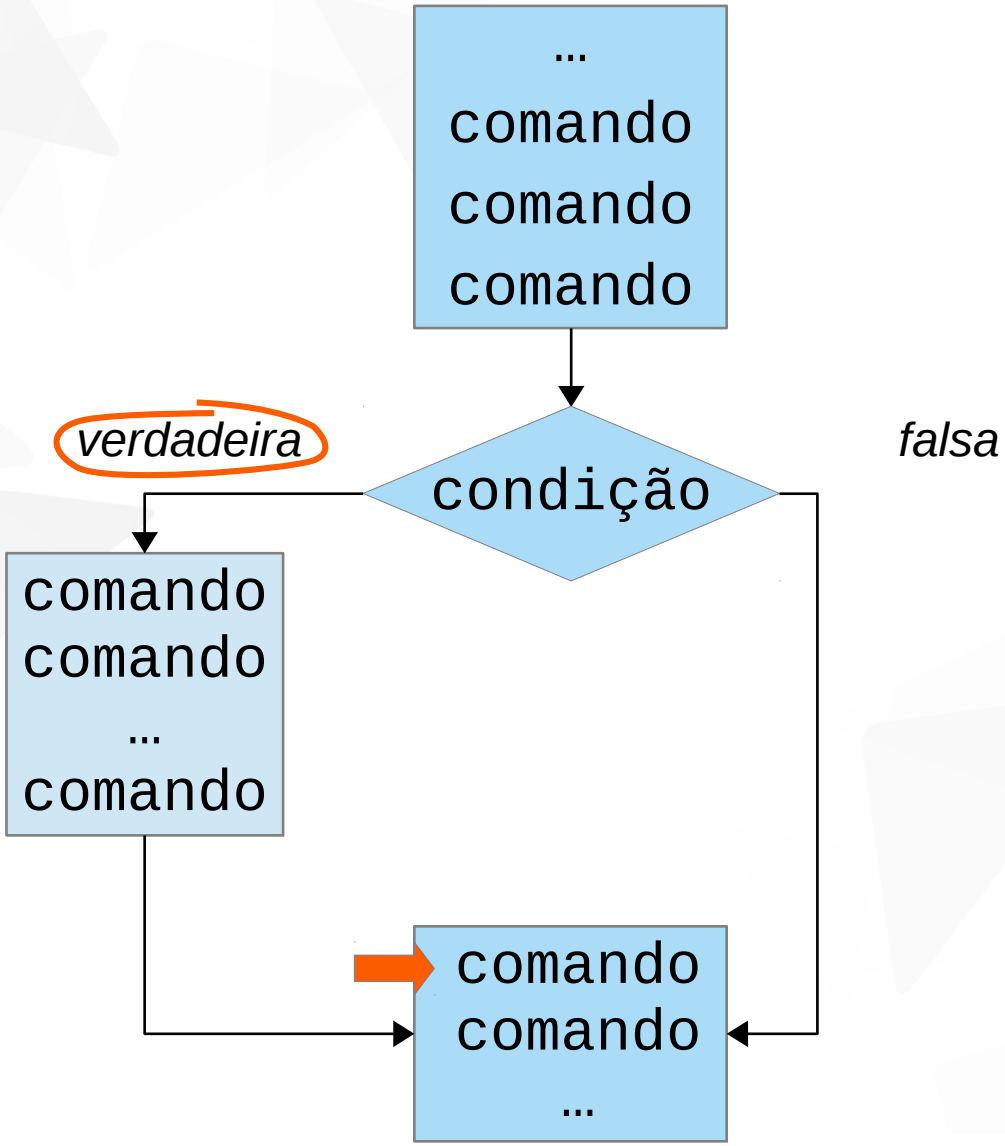
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



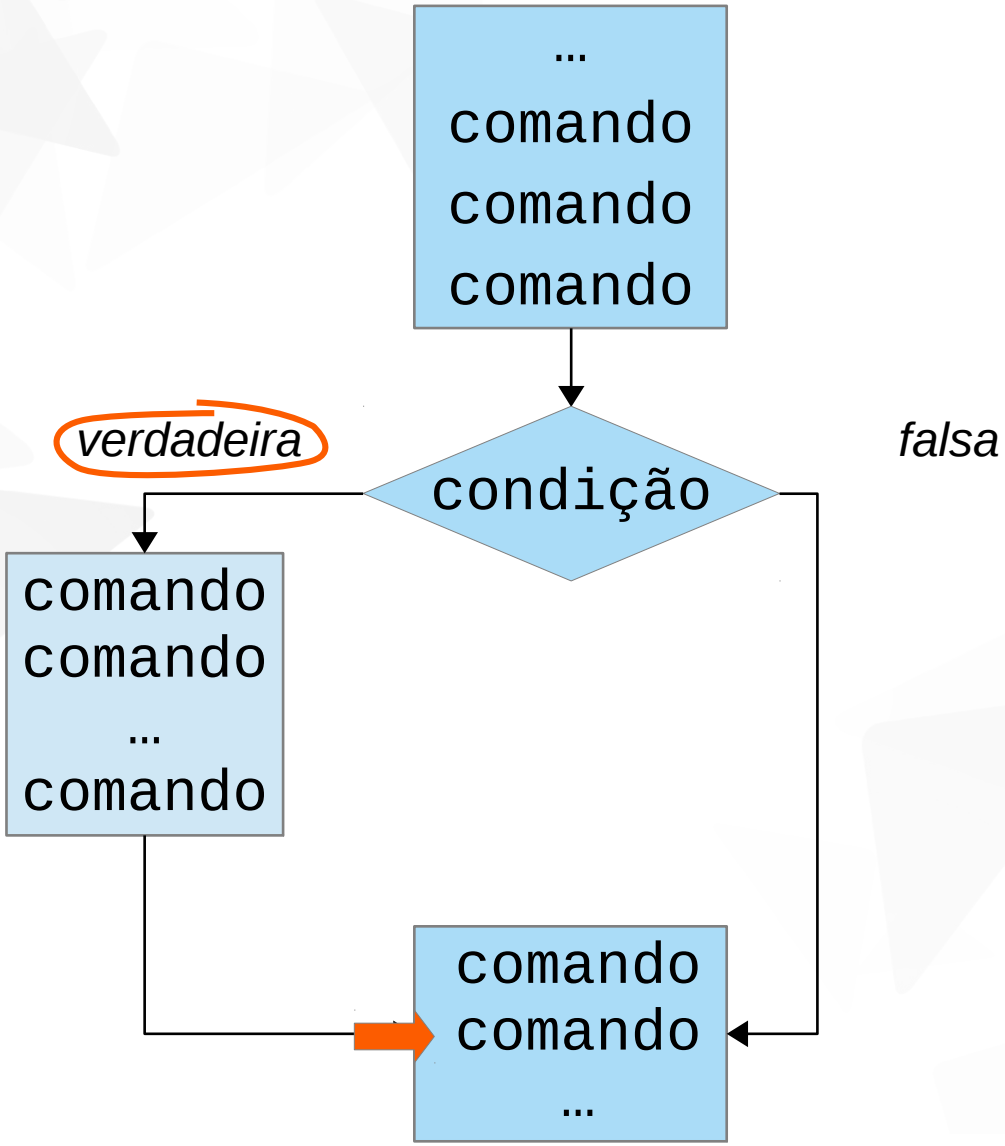
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



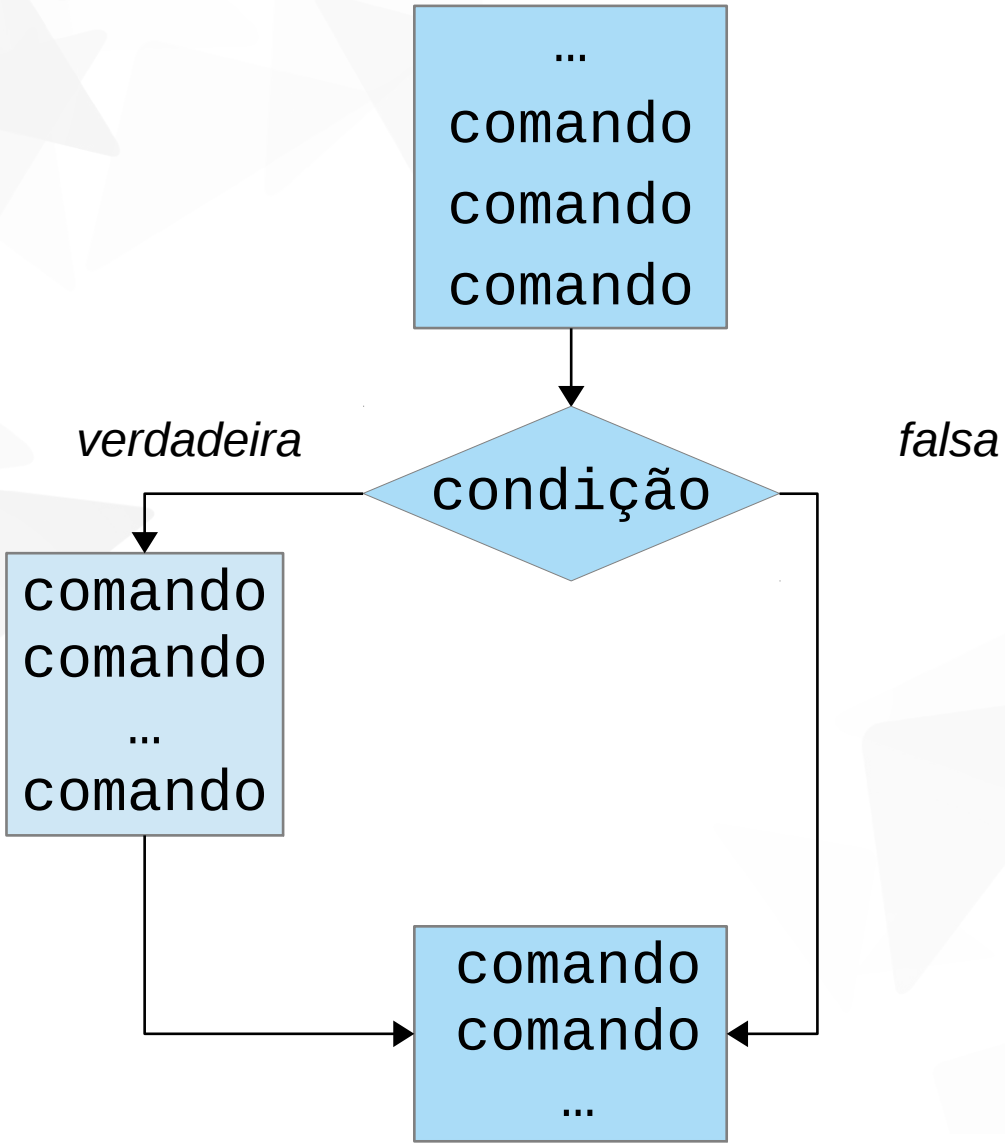
Estrutura condicional

Estrutura/desvio condicional **simples** (se-então)



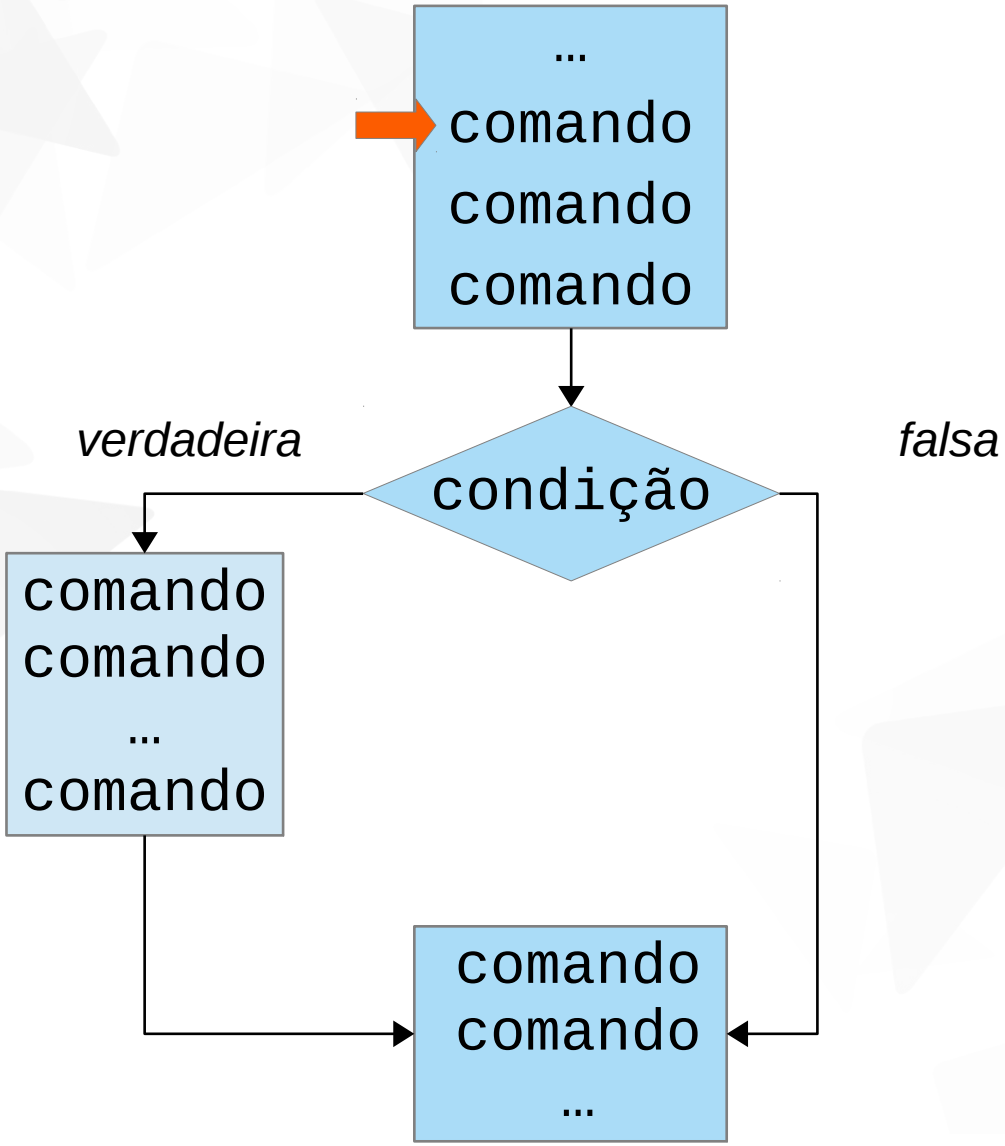
Estrutura condicional

E se a condição for **falsa?**



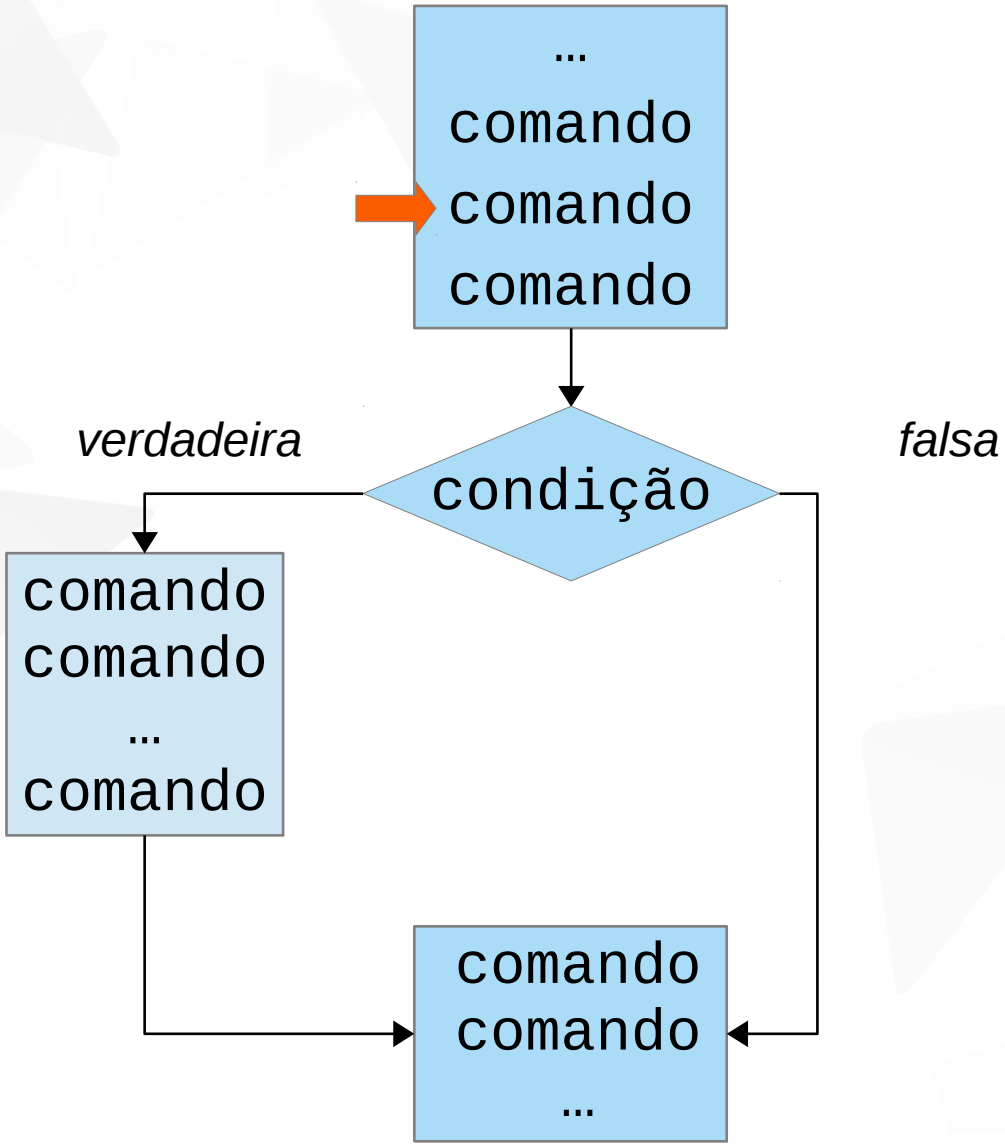
Estrutura condicional

E se a condição for **falsa?**



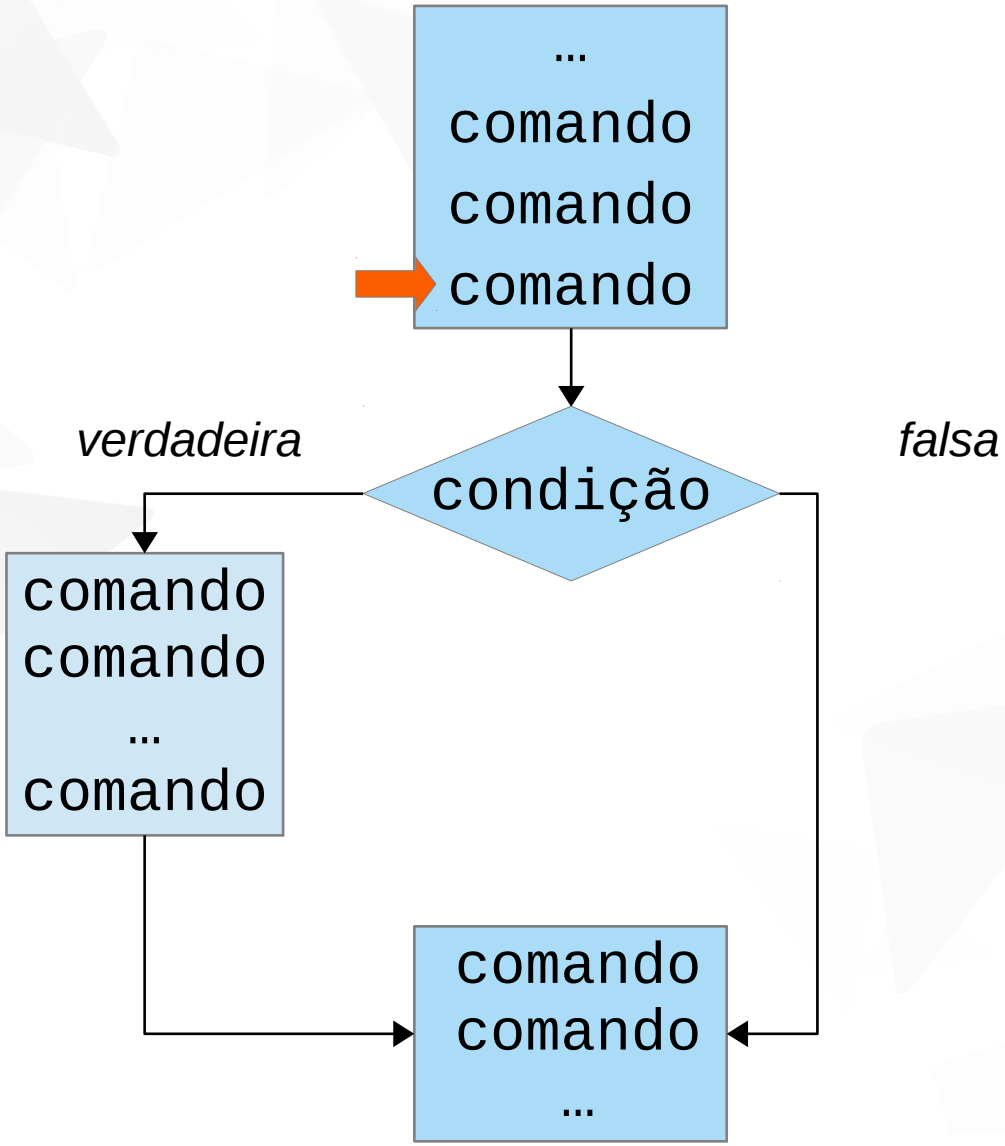
Estrutura condicional

E se a condição for **falsa?**



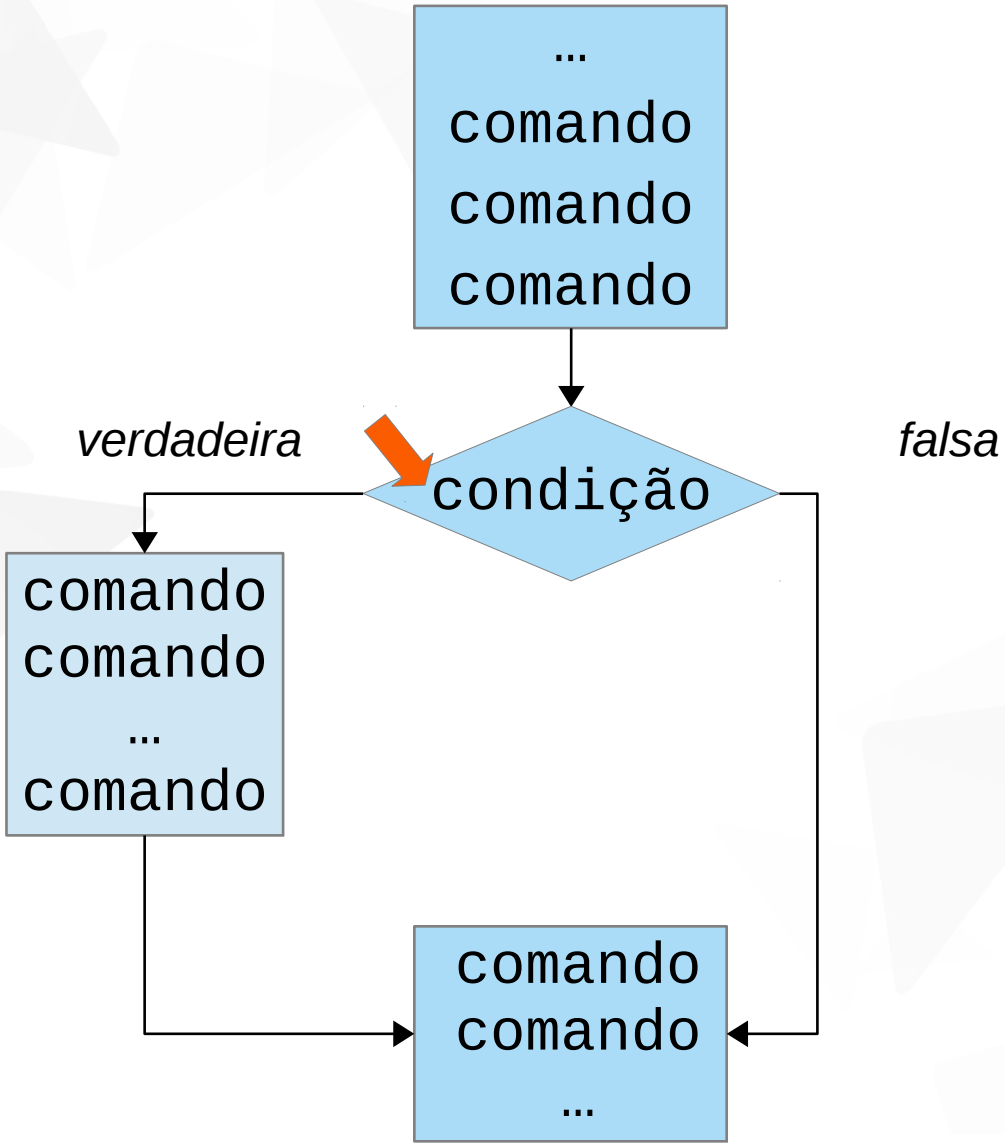
Estrutura condicional

E se a condição for **falsa?**



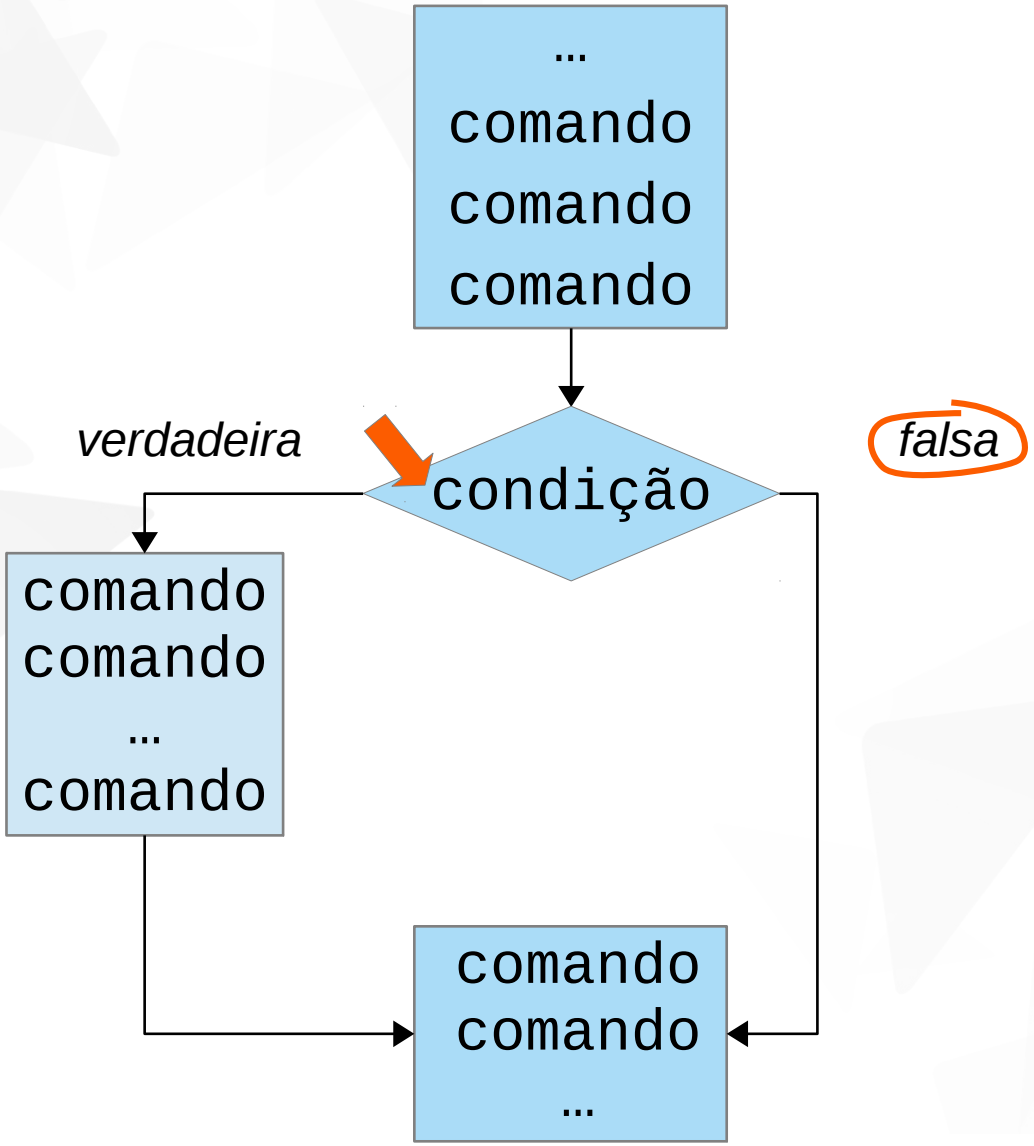
Estrutura condicional

E se a condição for **falsa?**



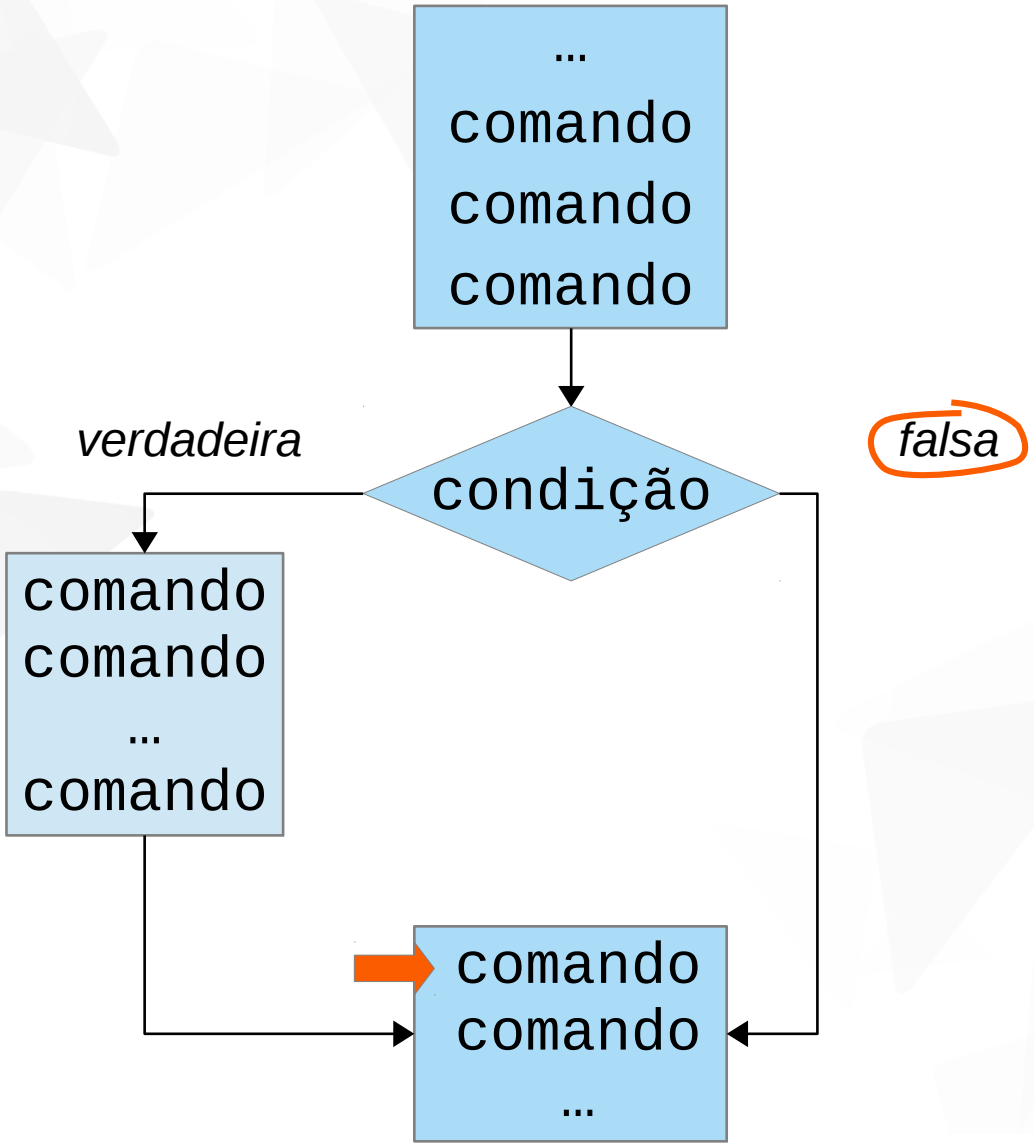
Estrutura condicional

E se a condição for **falsa?**



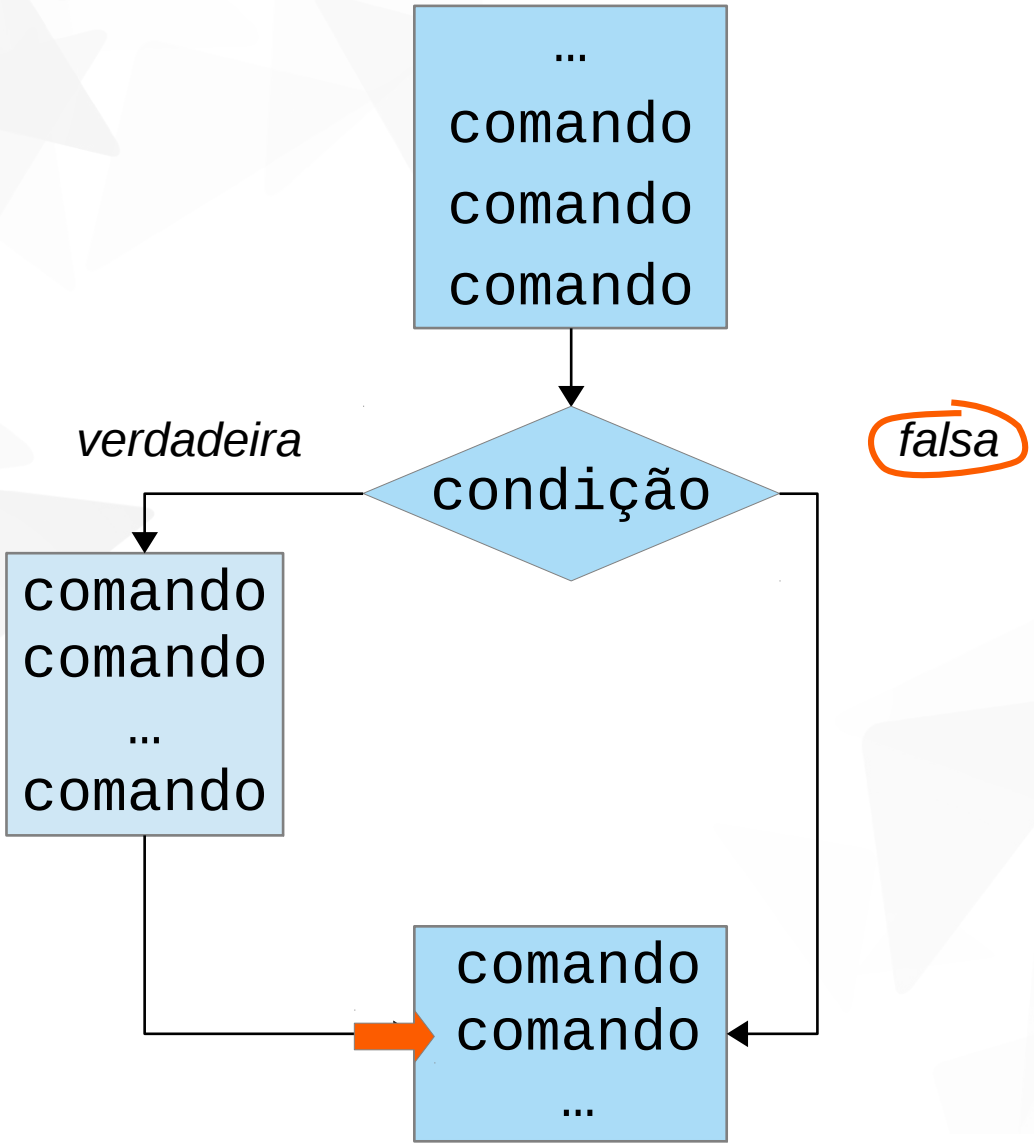
Estrutura condicional

E se a condição for **falsa?**



Estrutura condicional

E se a condição for **falsa?**

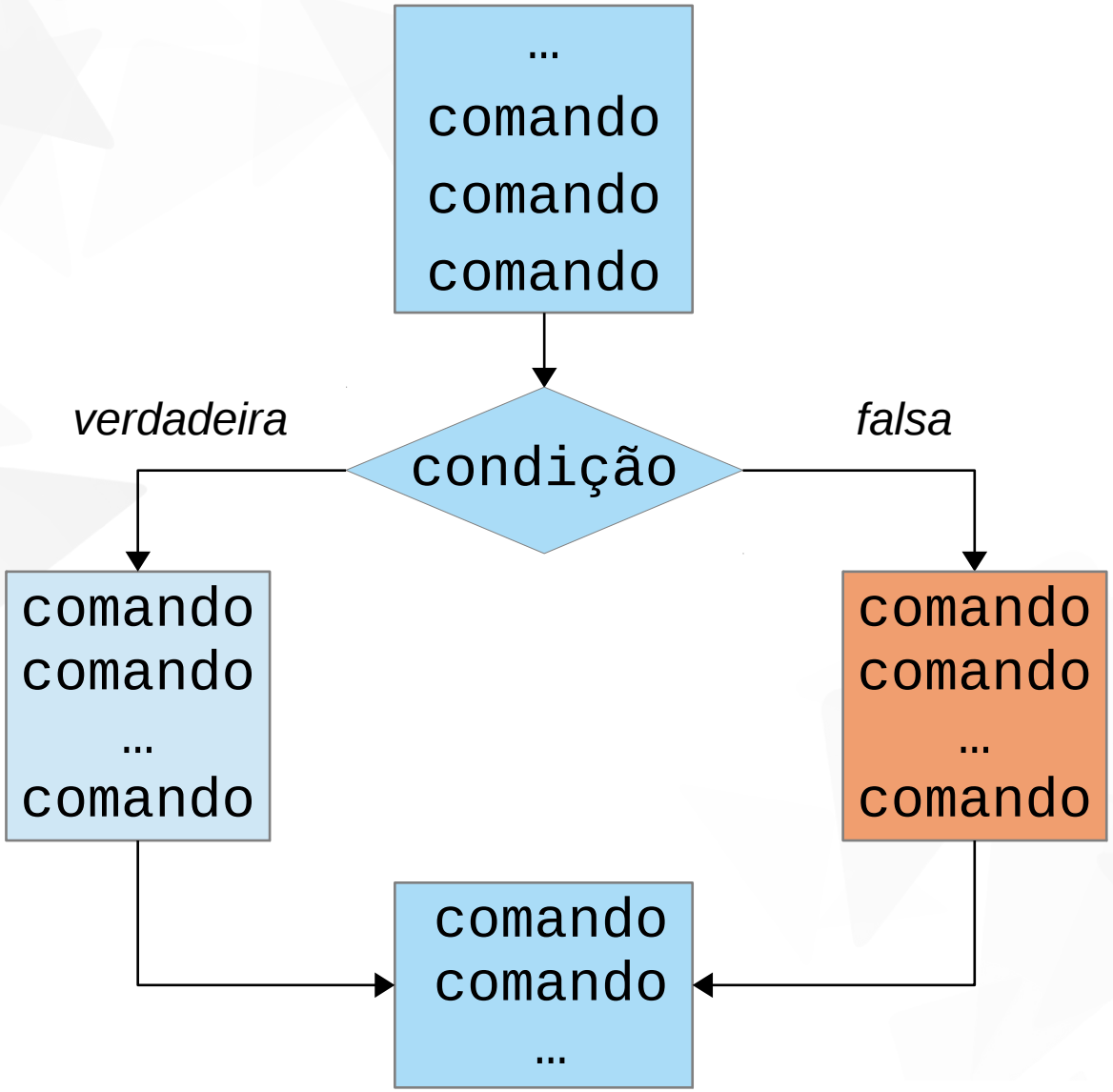


Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)

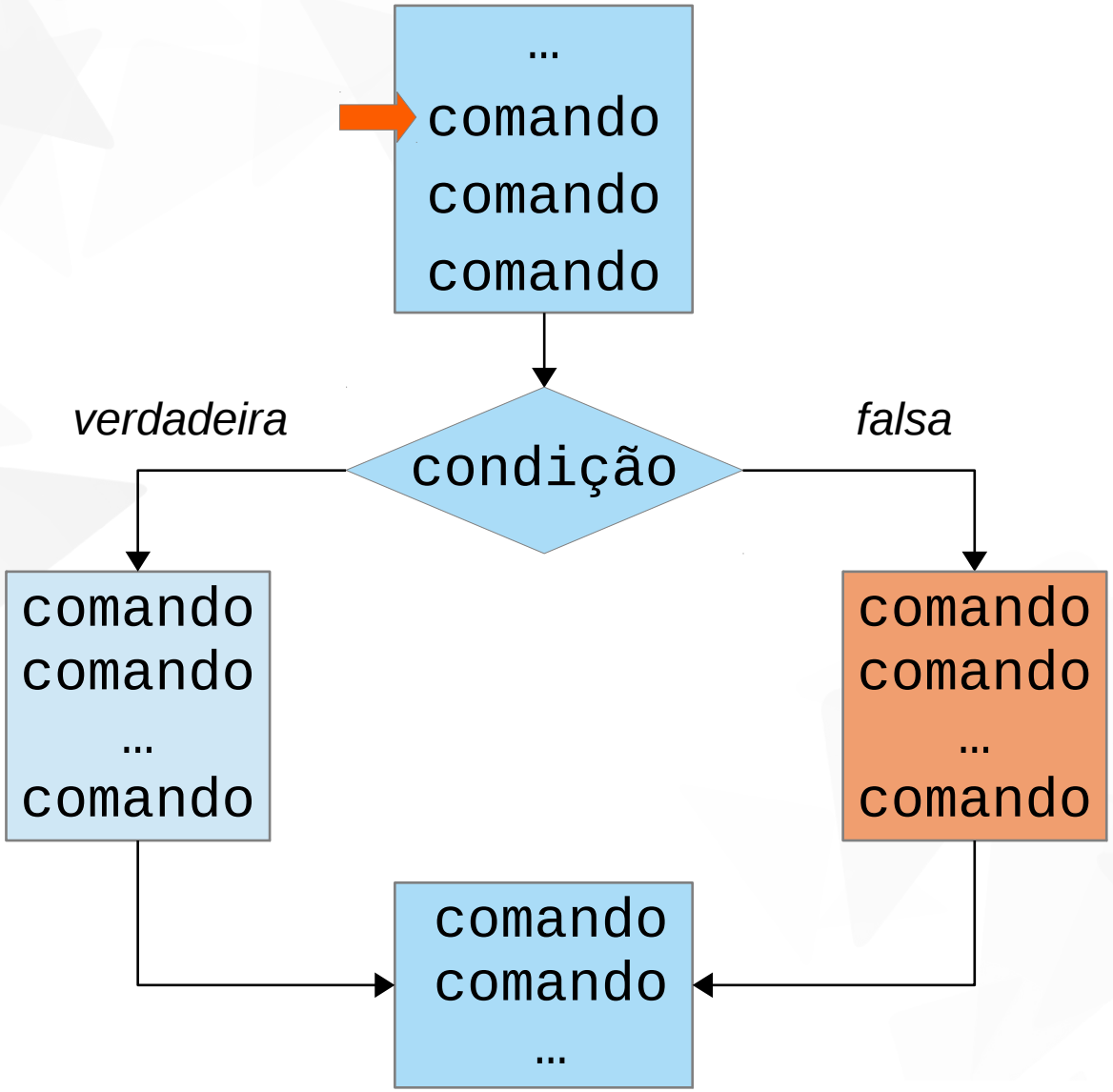
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



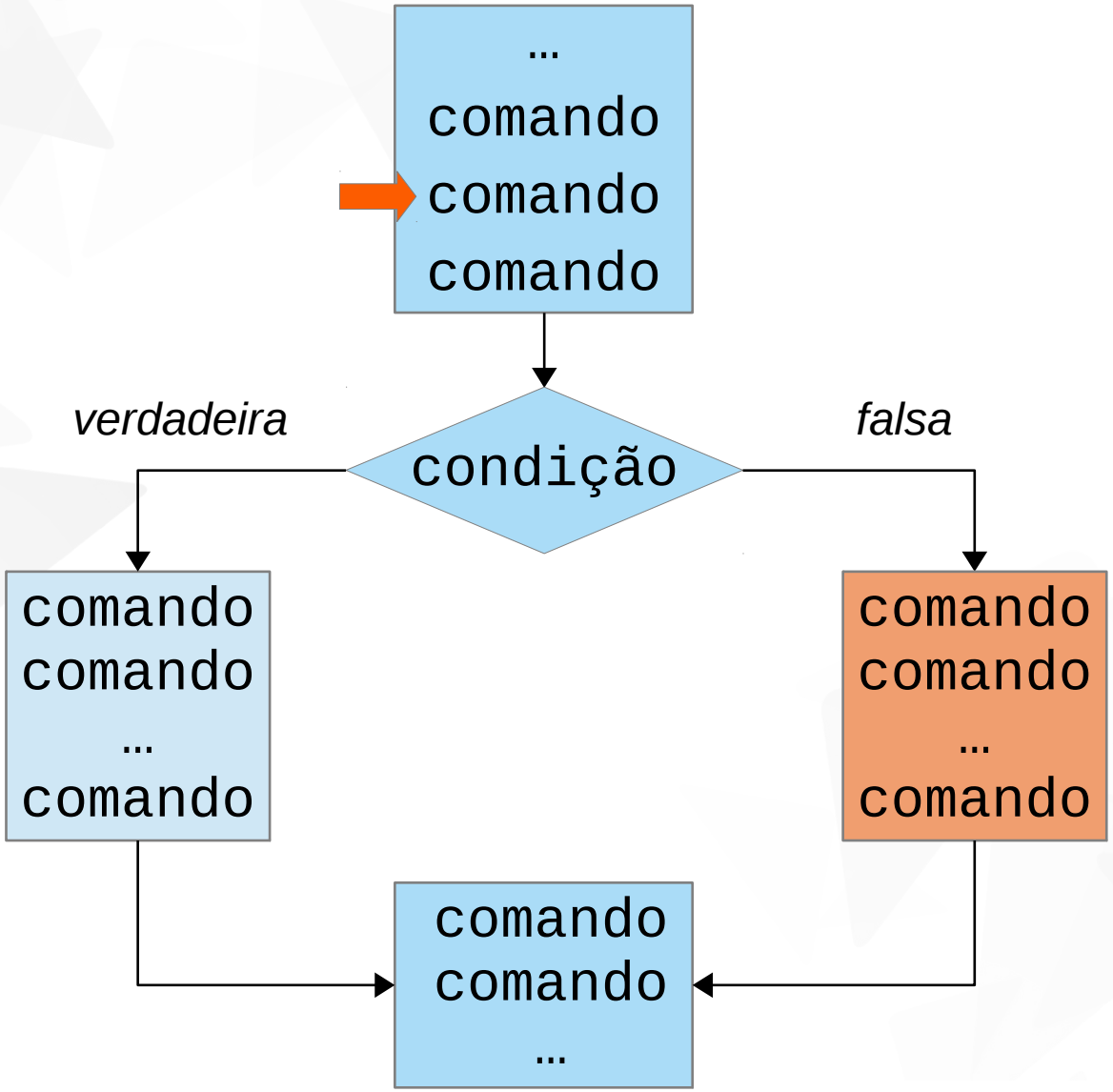
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



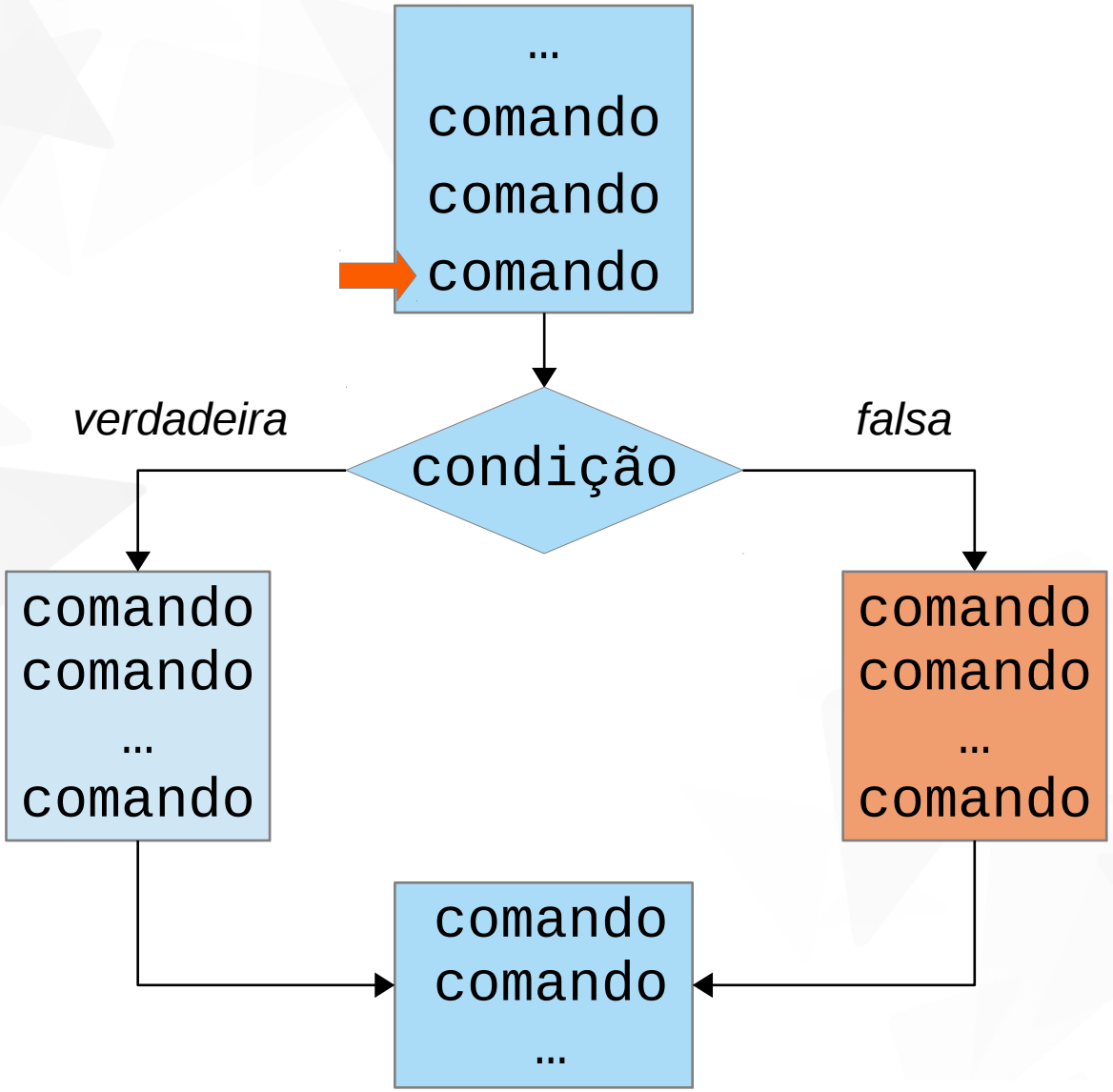
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



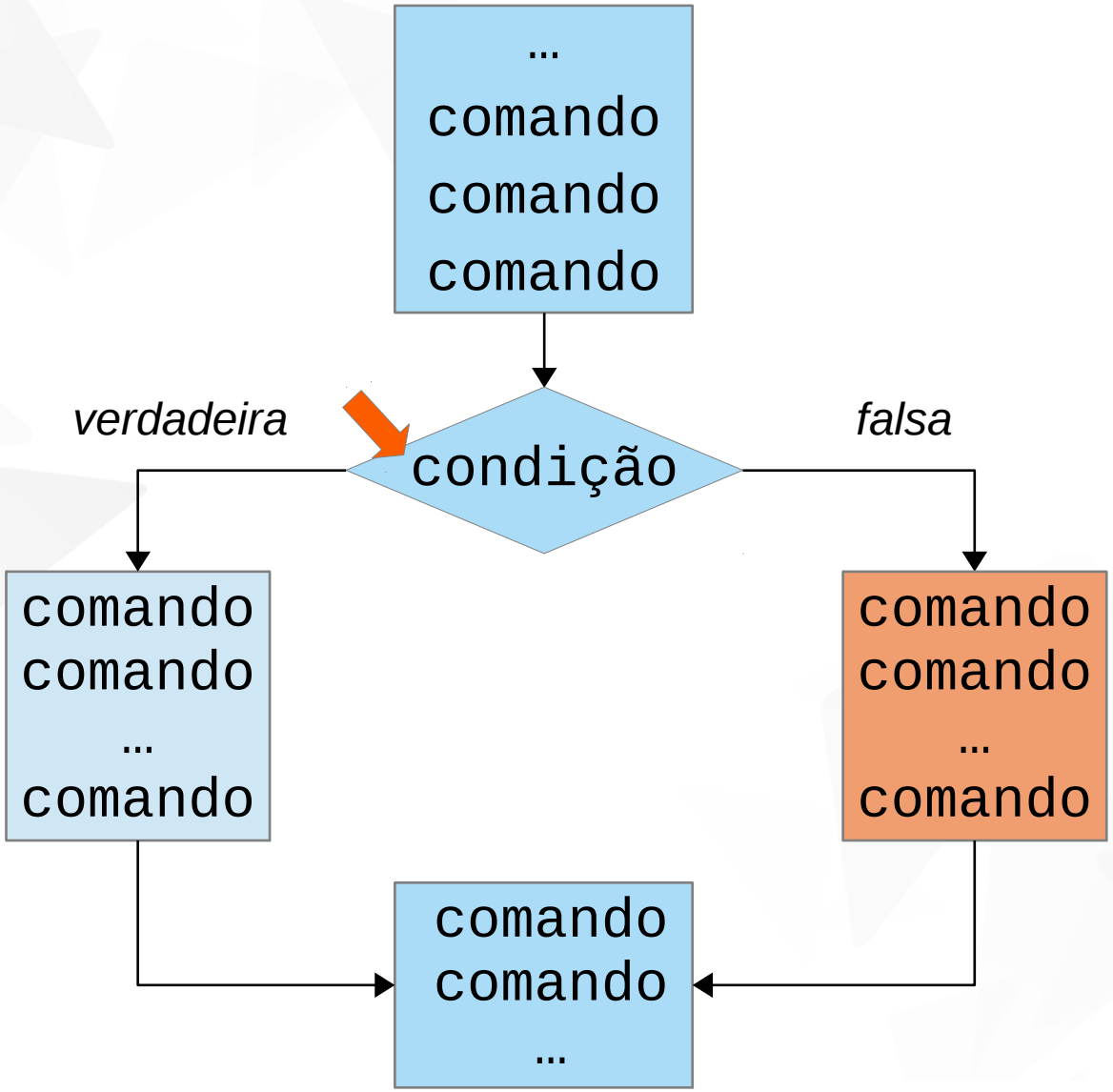
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



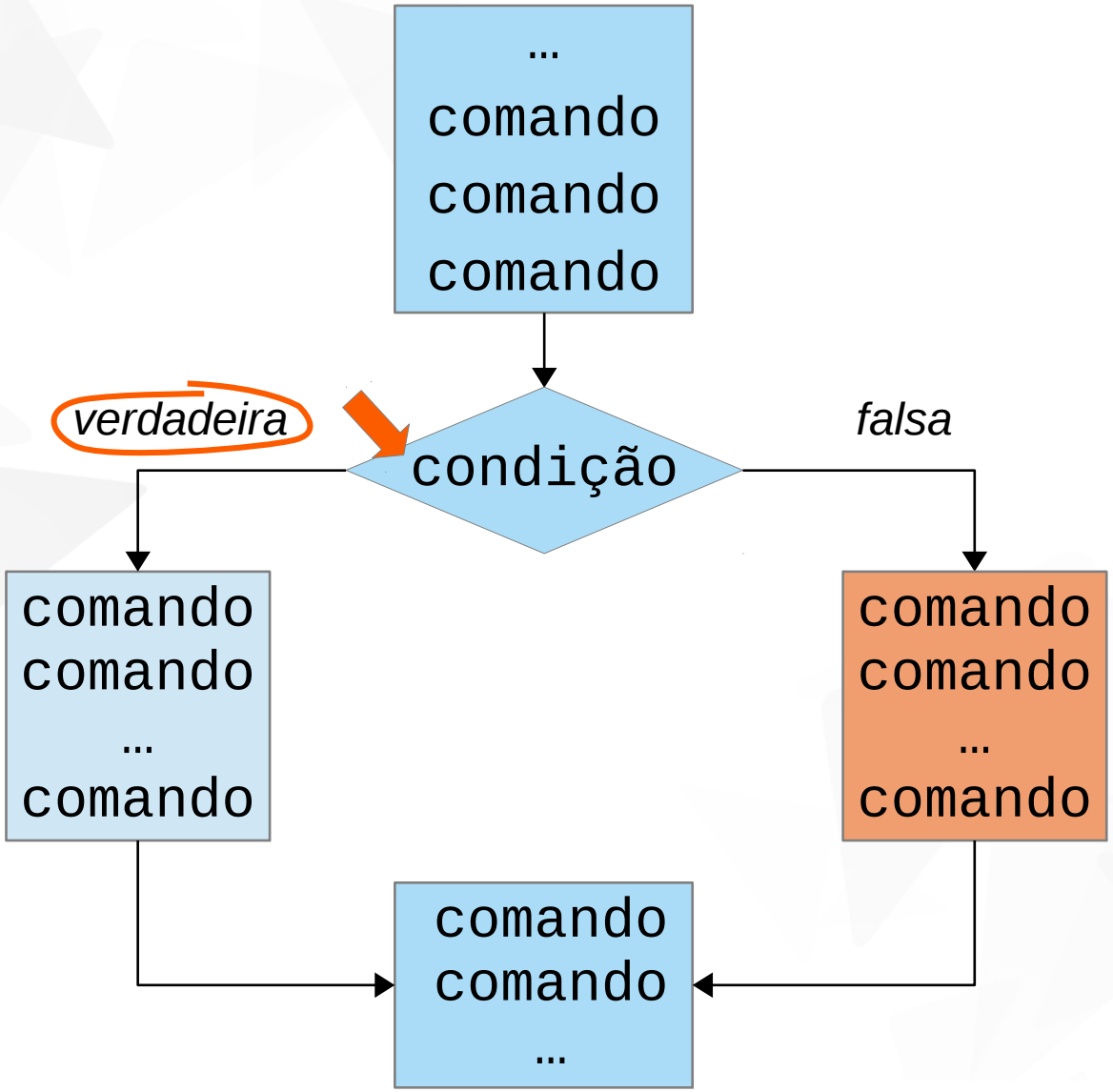
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



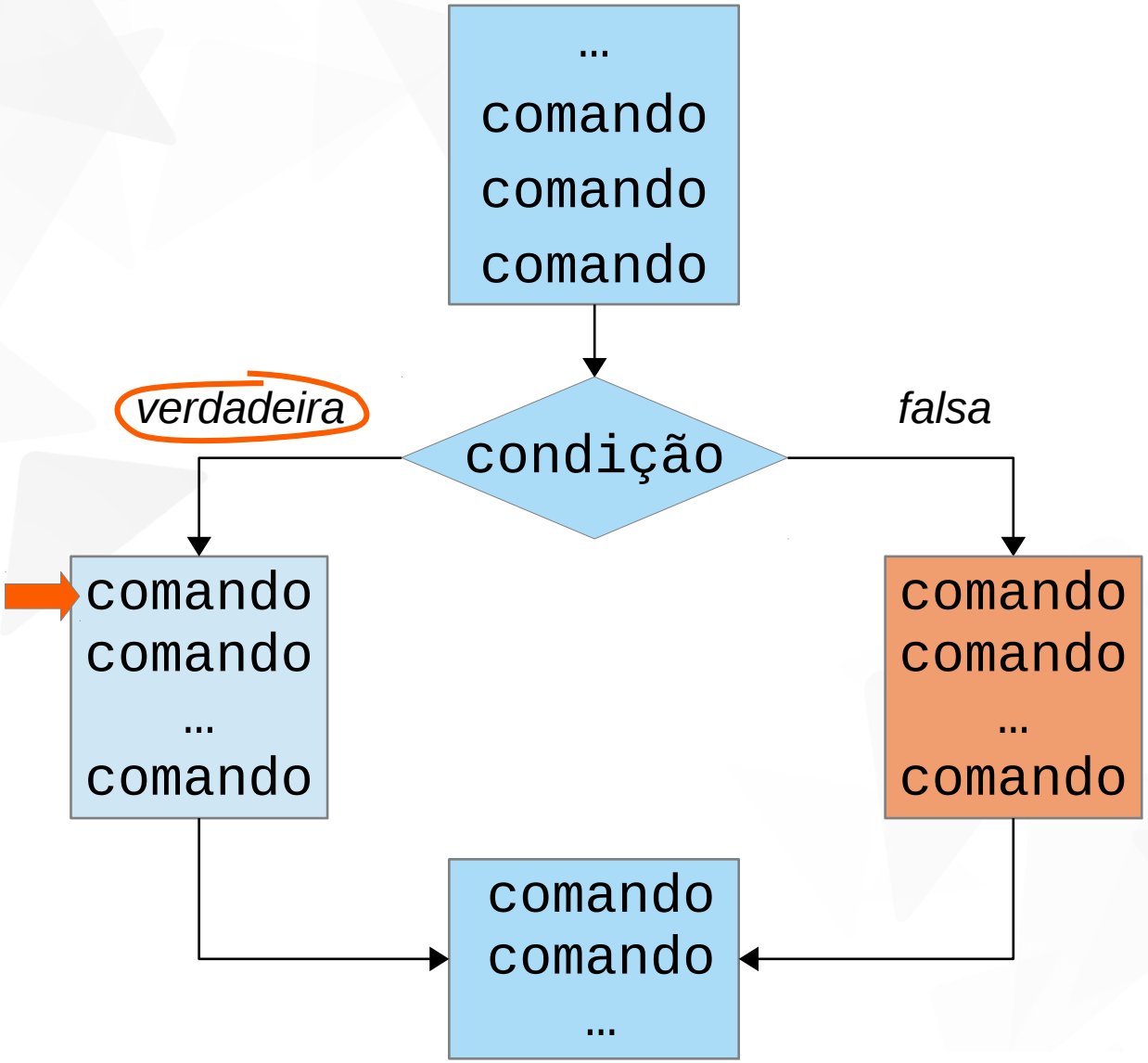
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



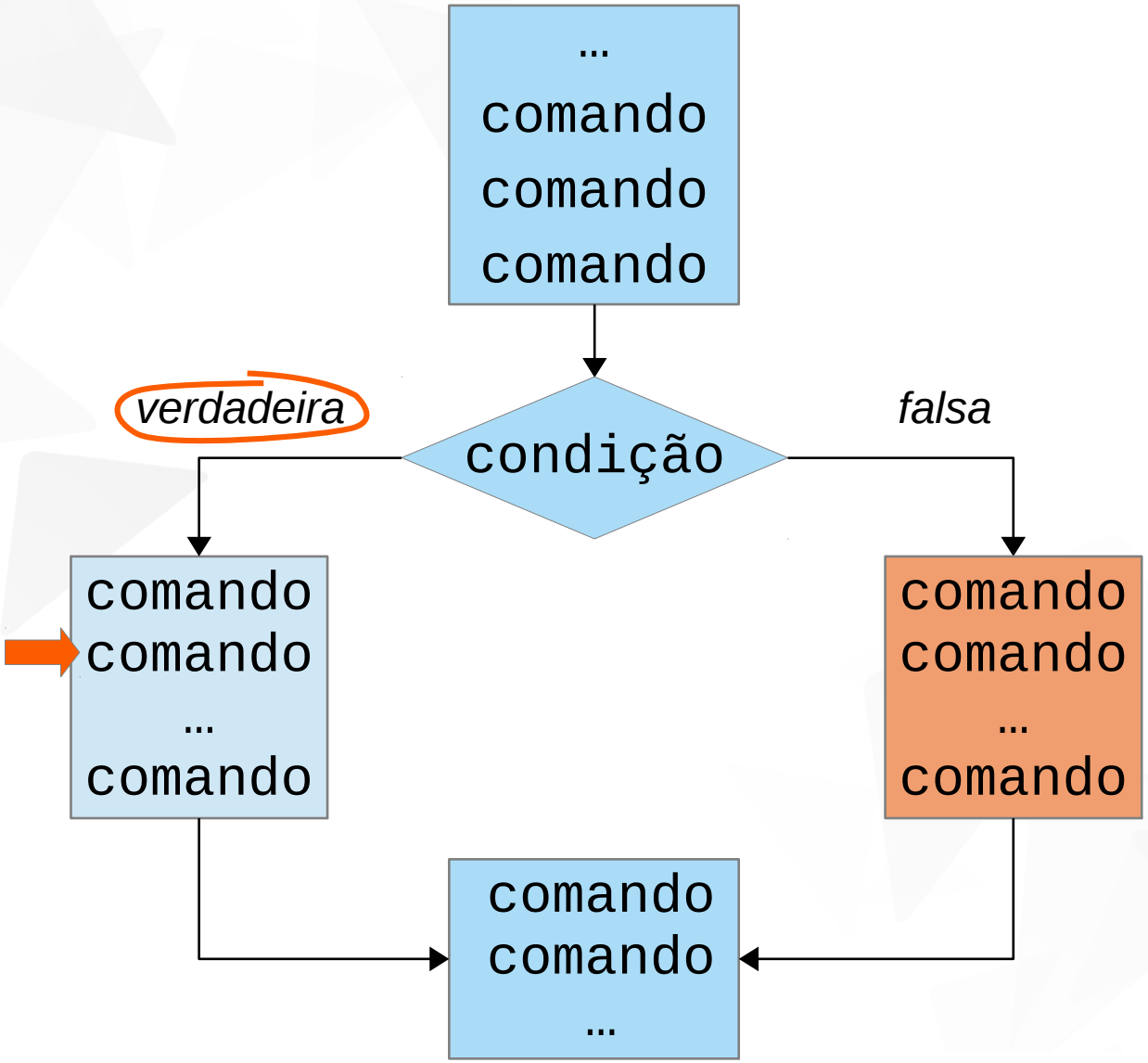
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



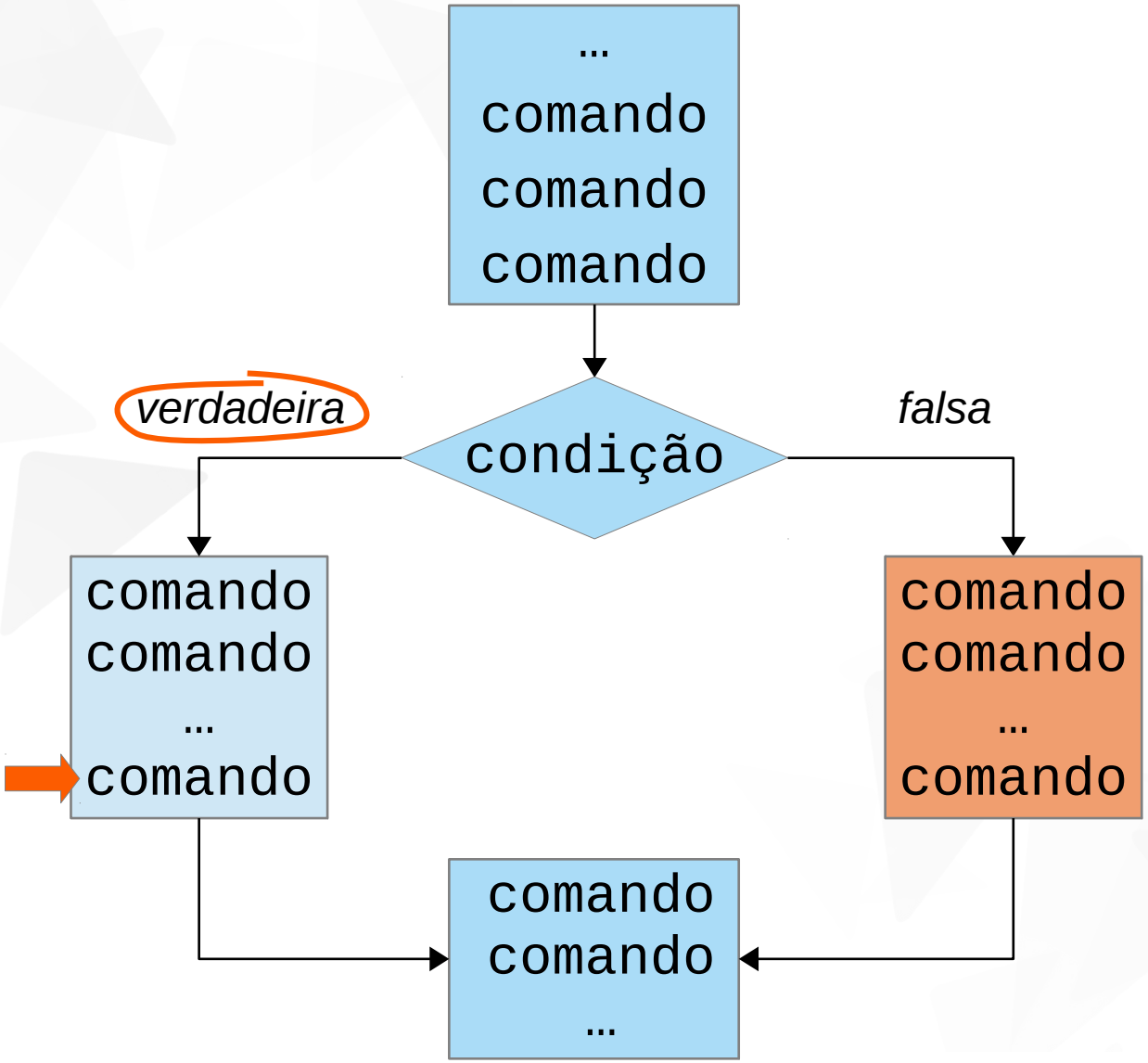
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



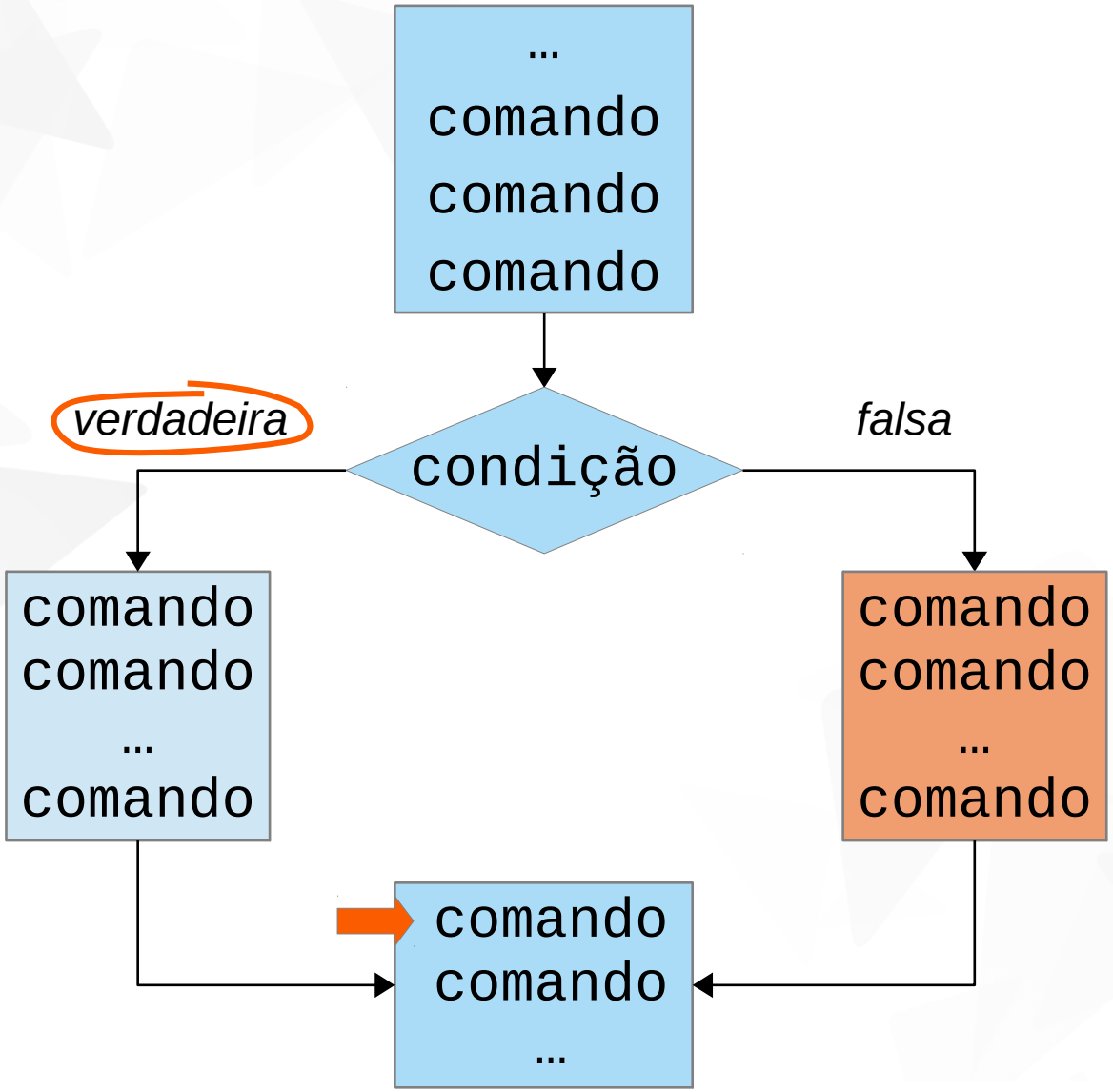
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



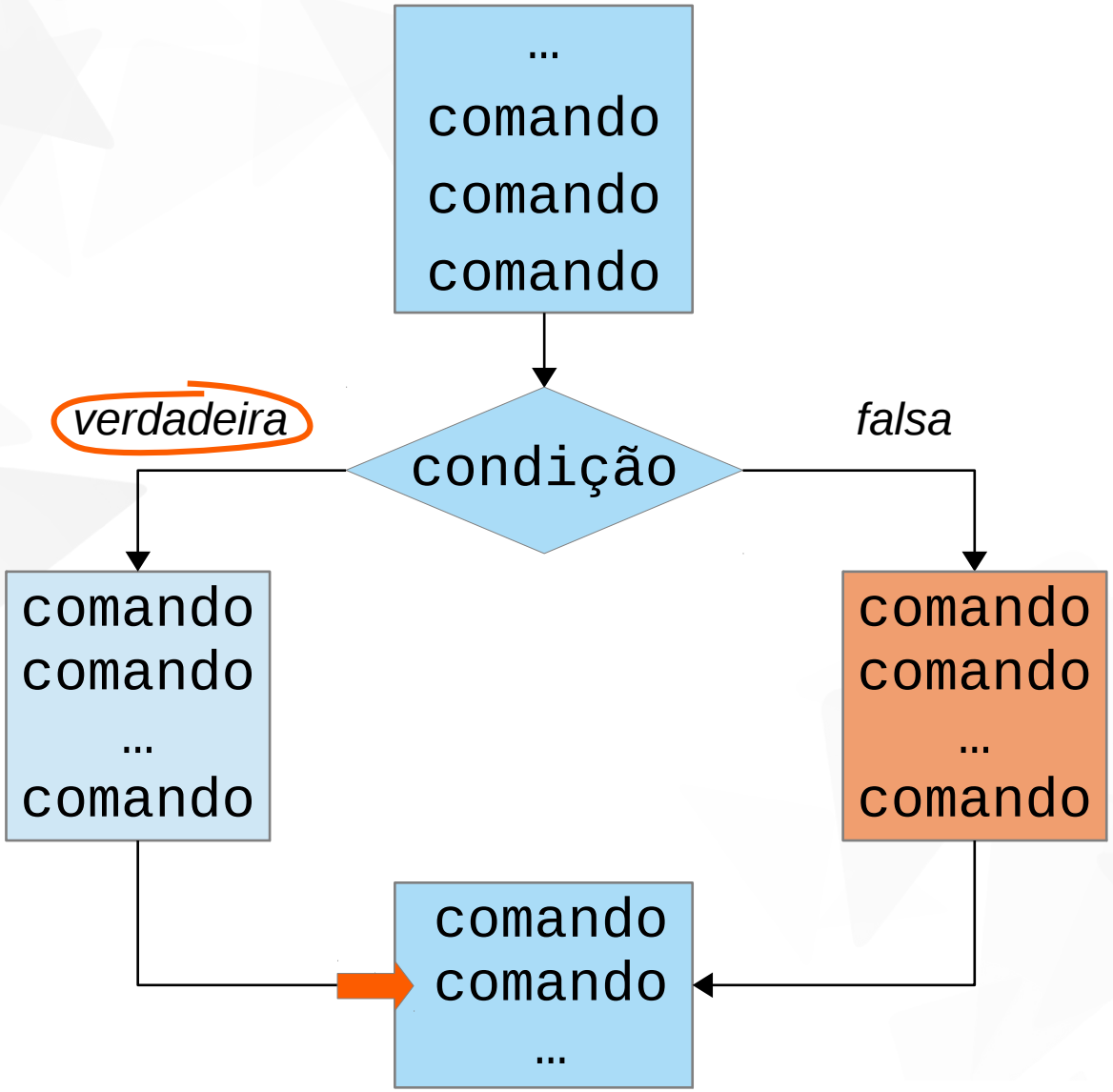
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



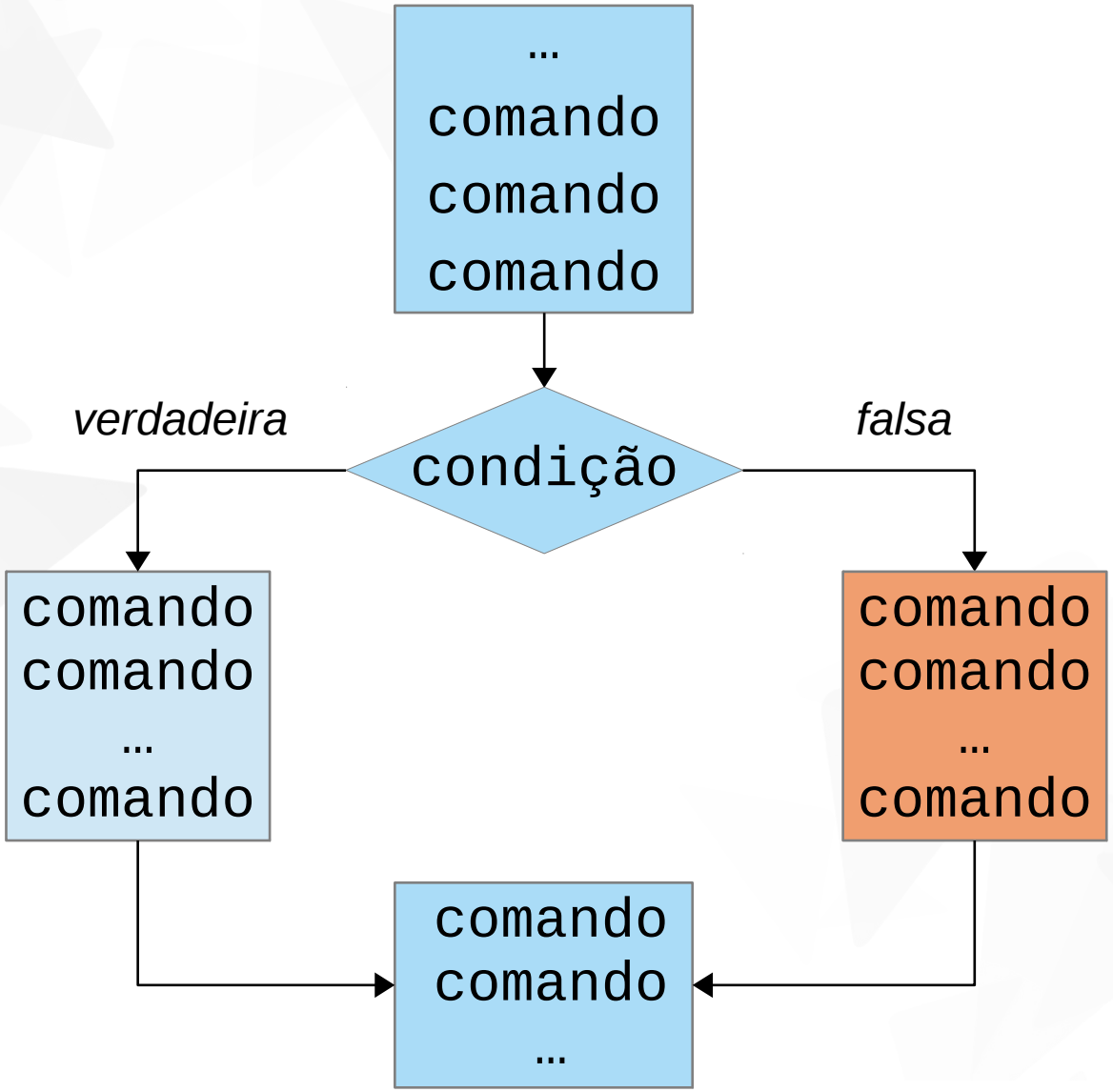
Estrutura condicional

Estrutura/desvio condicional **composta**(o) (se-então-senão)



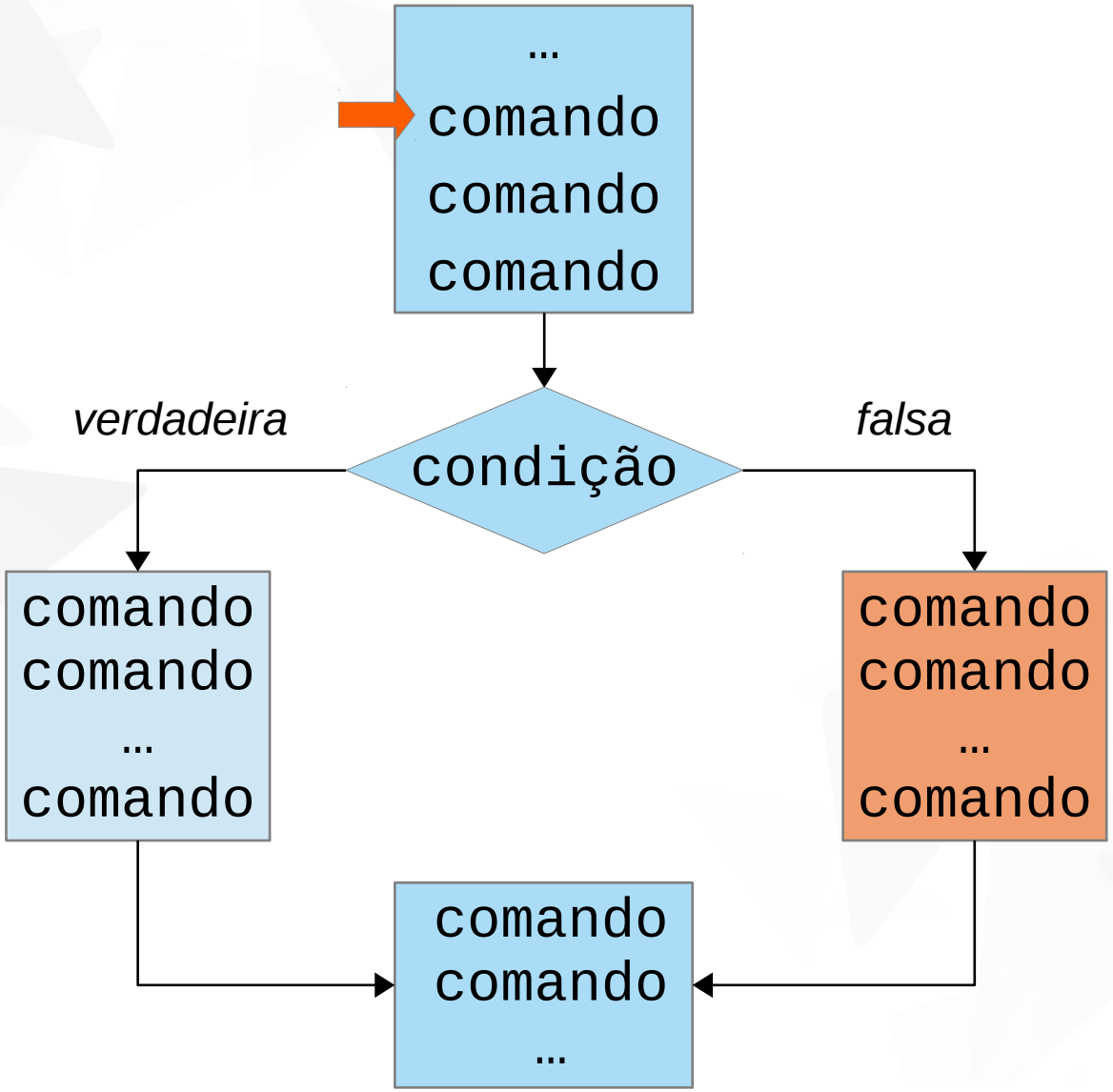
Estrutura condicional

E se a condição for **falsa?**



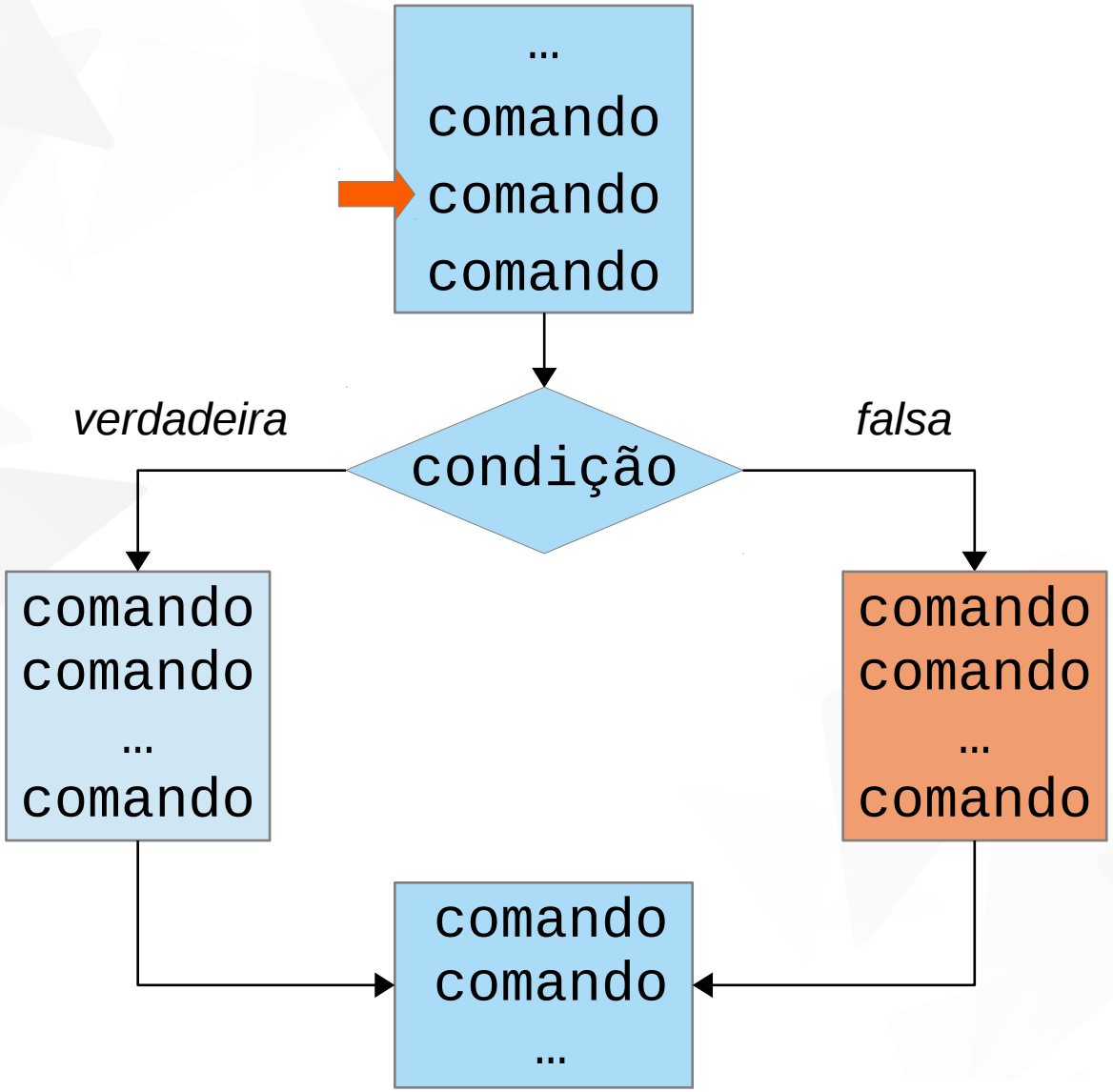
Estrutura condicional

E se a condição for **falsa?**



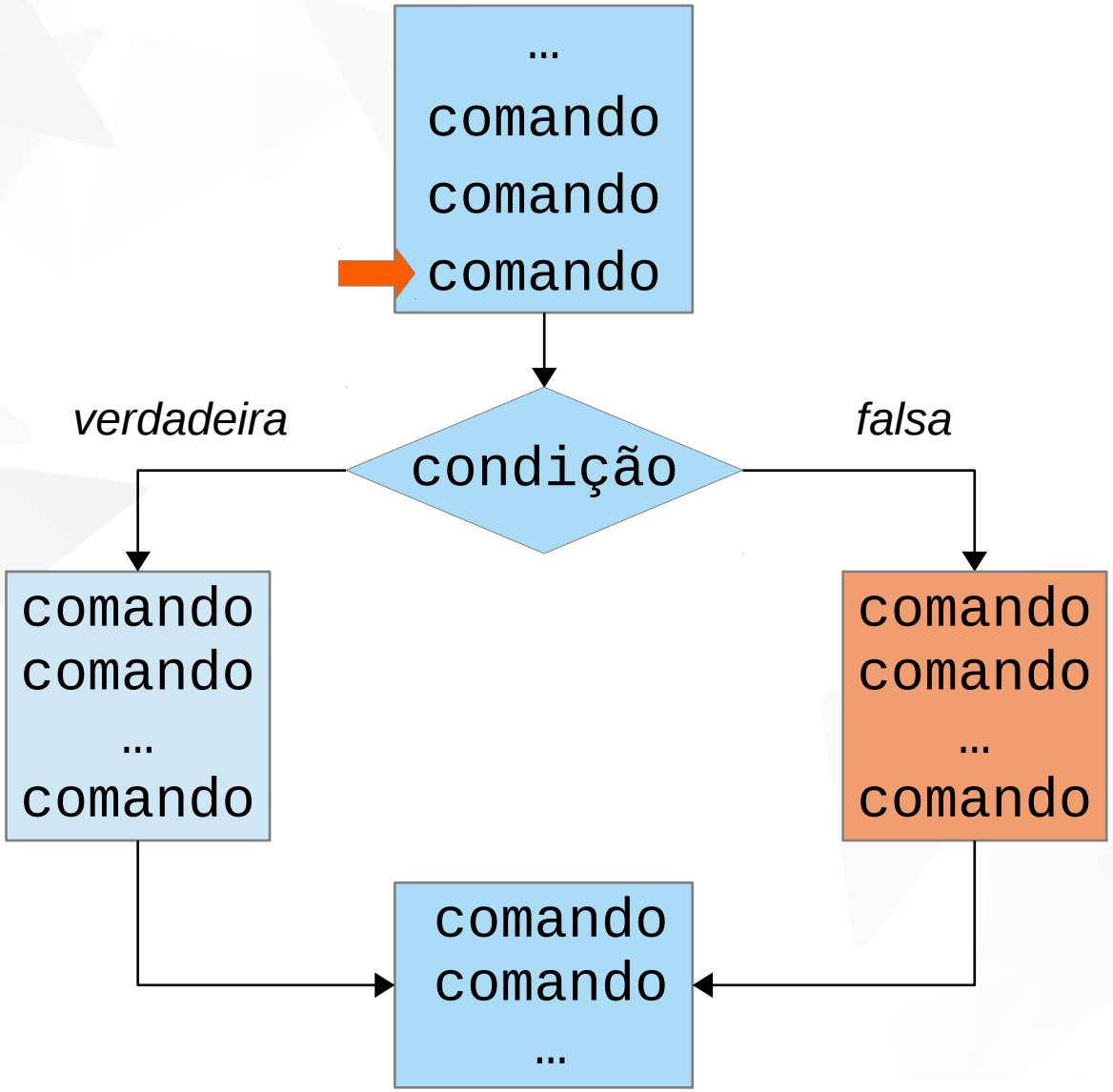
Estrutura condicional

E se a condição for **falsa?**



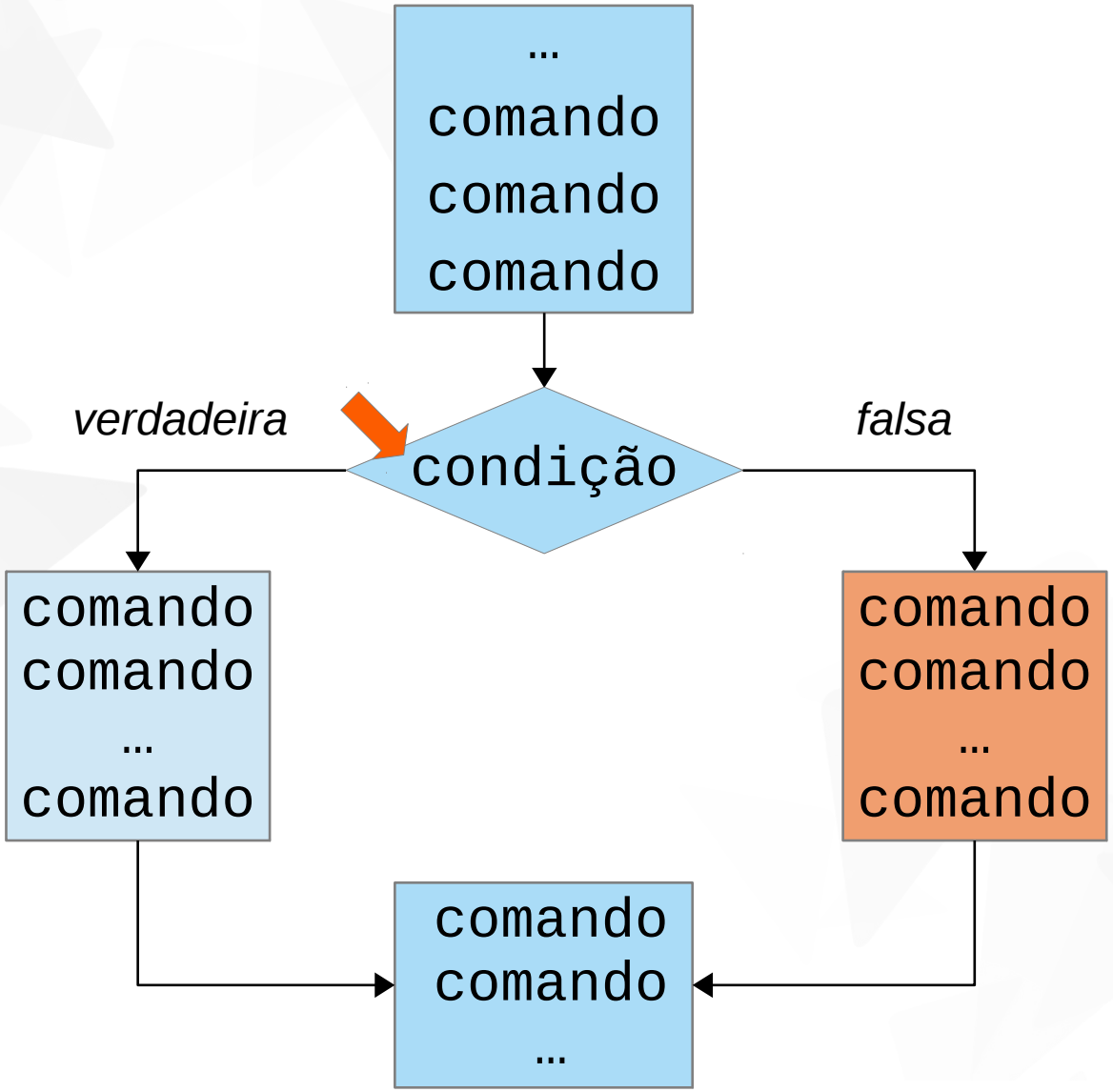
Estrutura condicional

E se a condição for **falsa?**



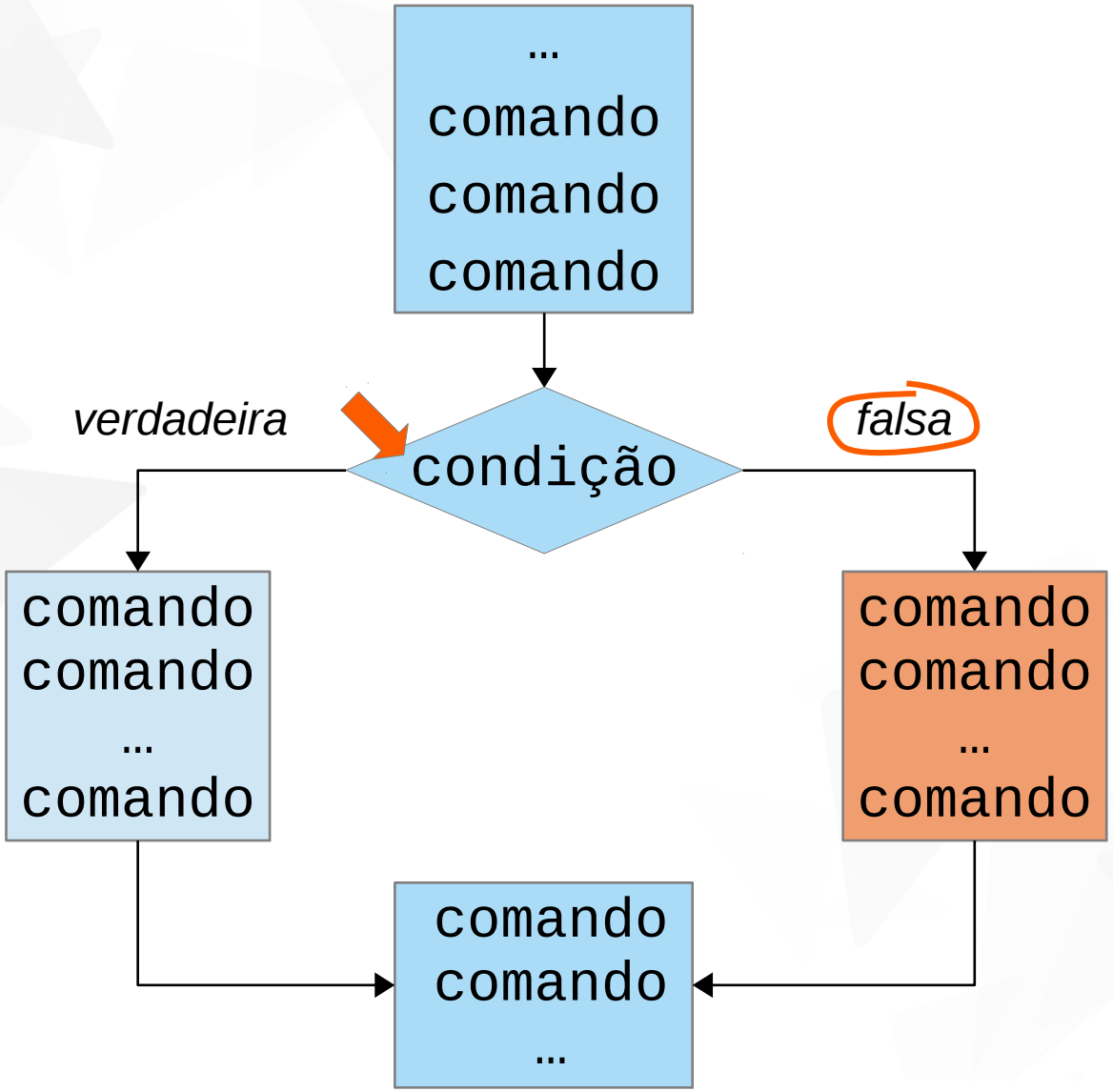
Estrutura condicional

E se a condição for **falsa?**



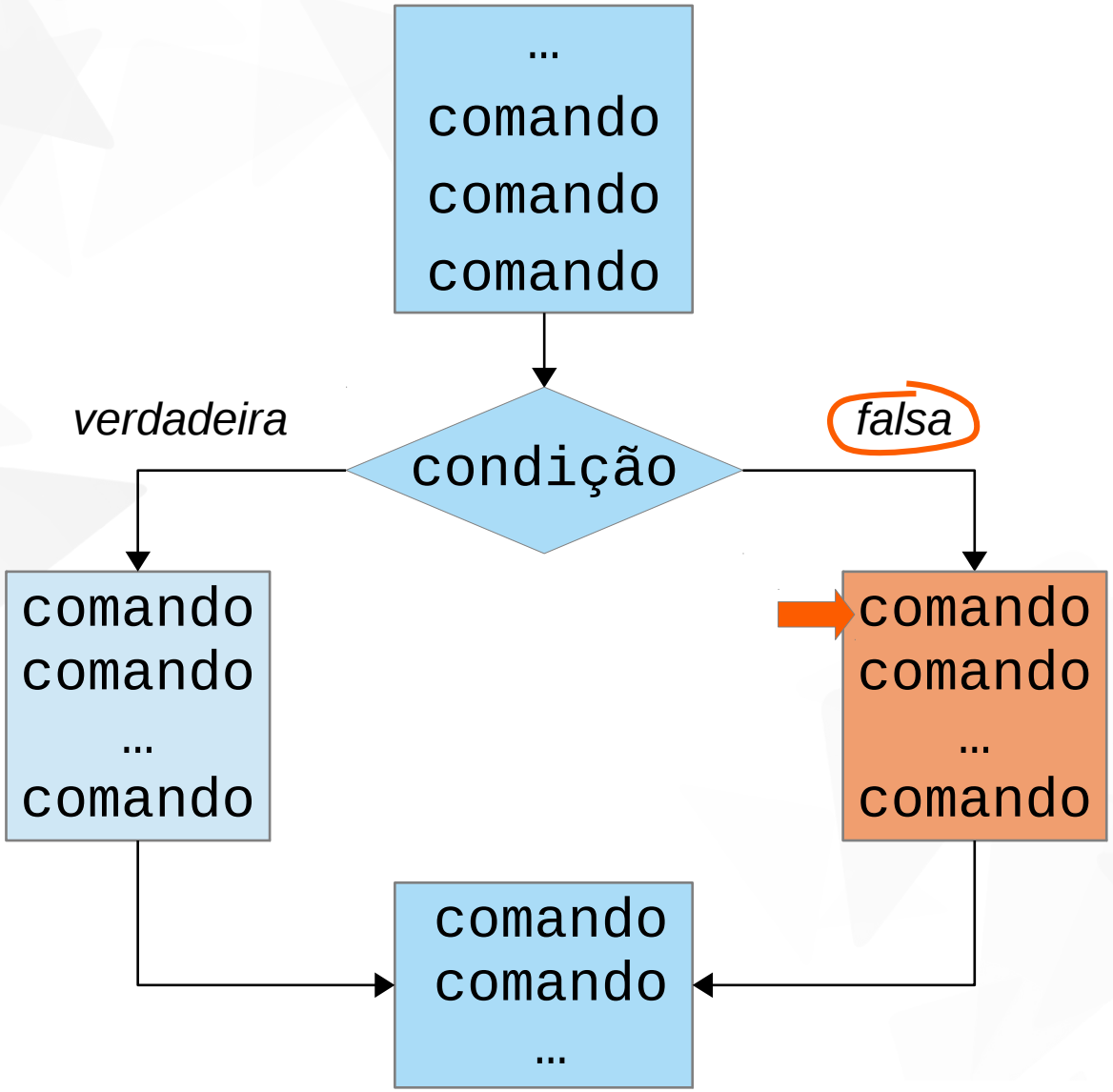
Estrutura condicional

E se a condição for **falsa?**



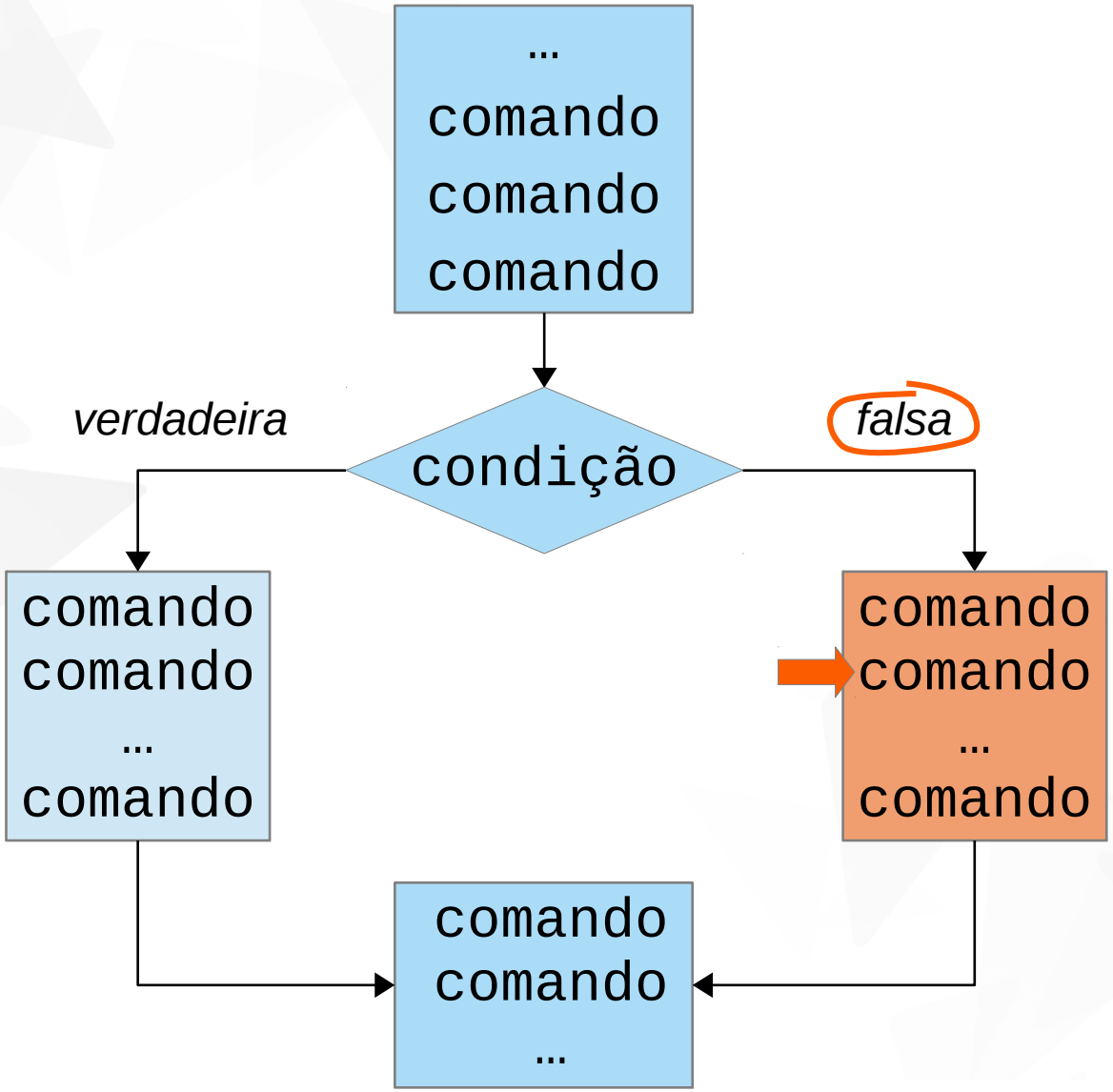
Estrutura condicional

E se a condição for **falsa?**



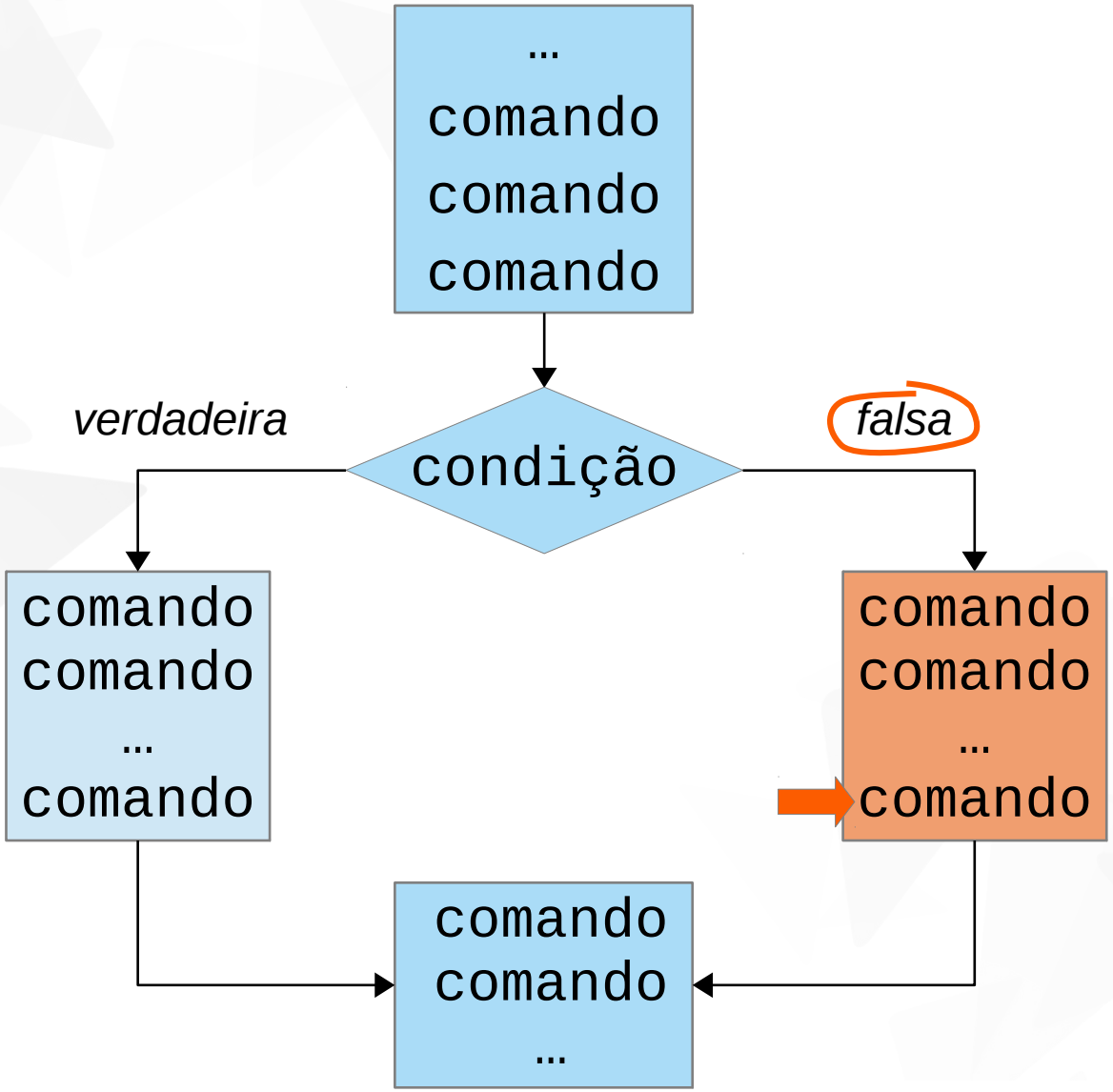
Estrutura condicional

E se a condição for **falsa?**



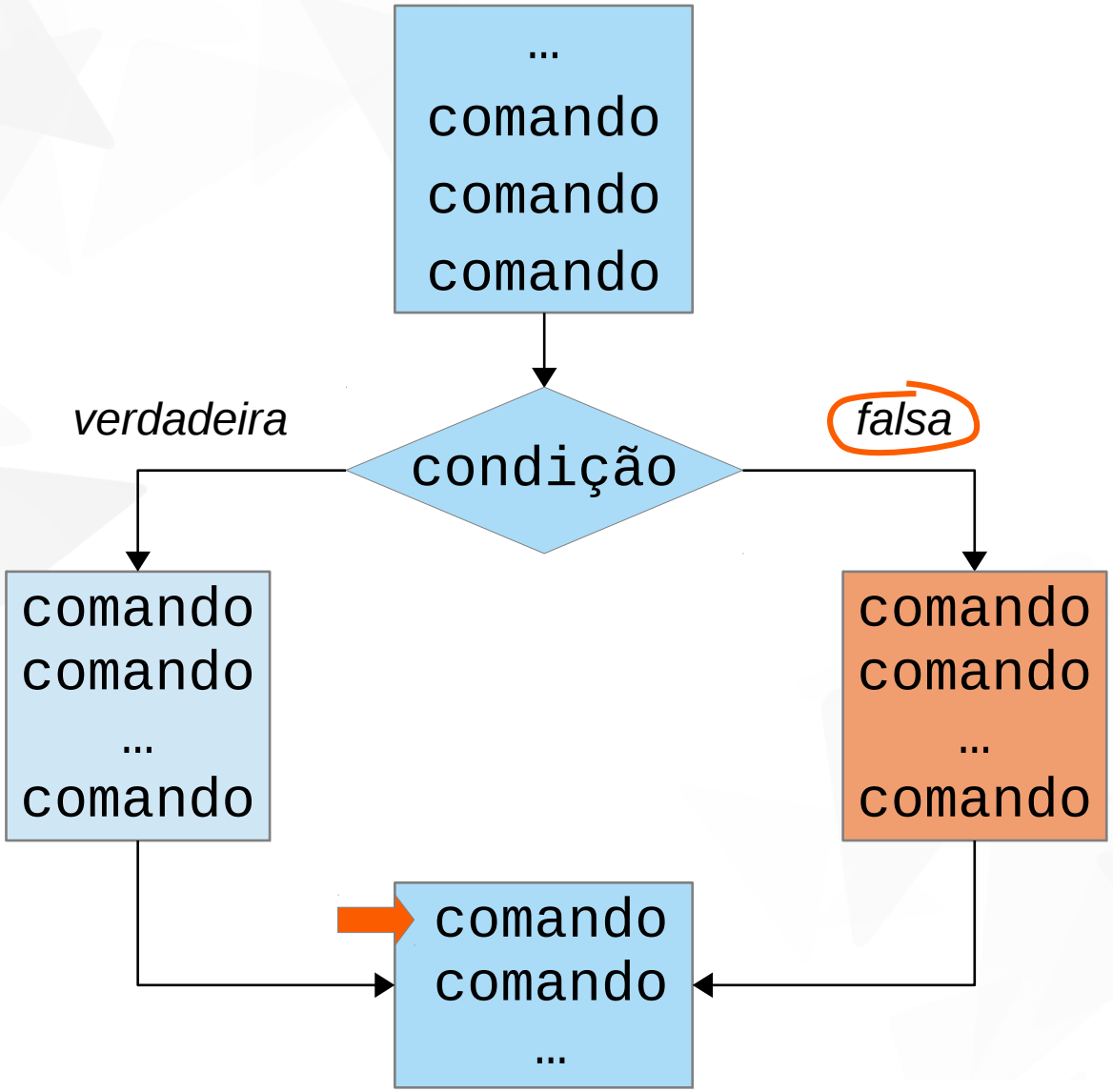
Estrutura condicional

E se a condição for **falsa?**



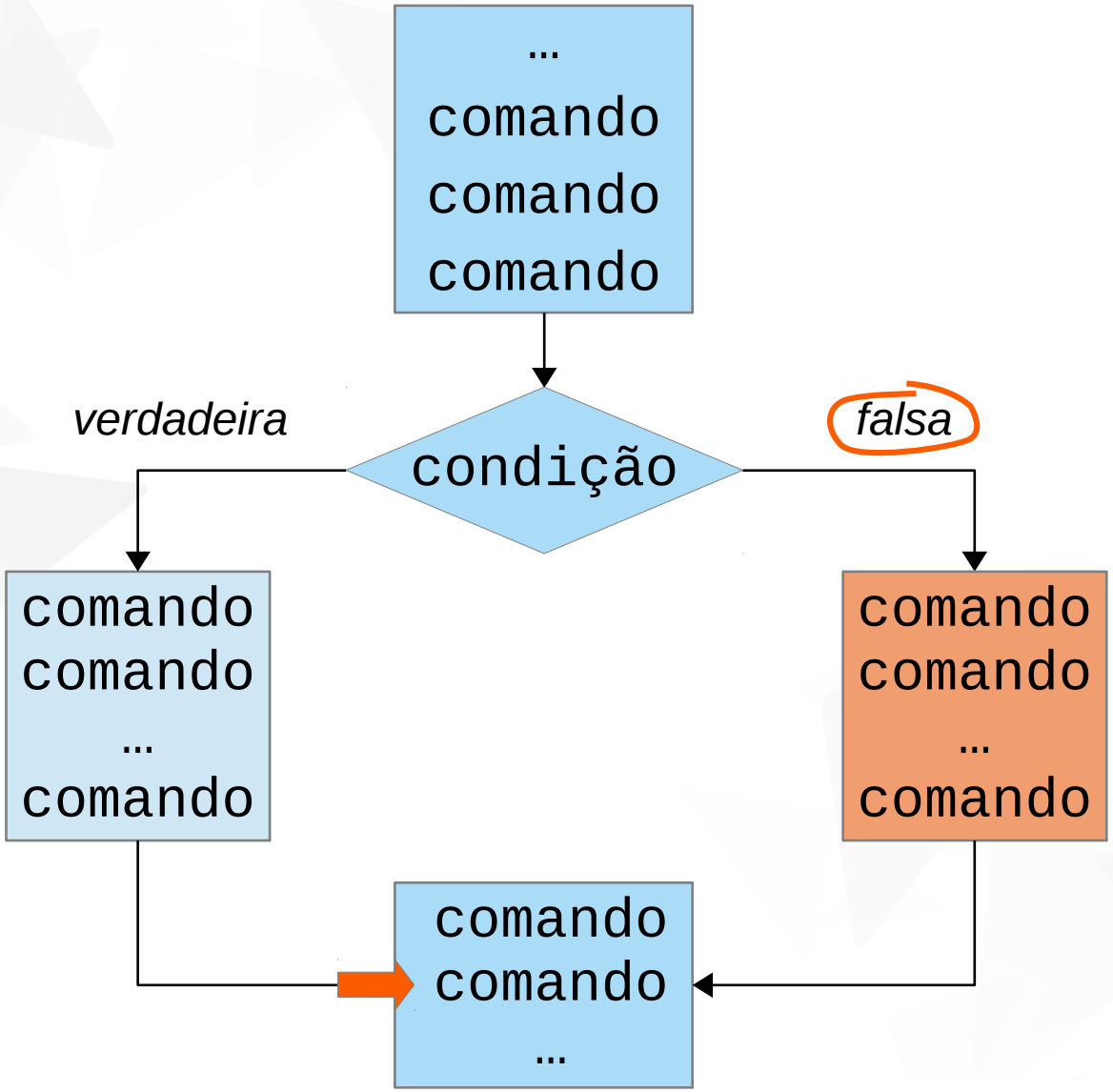
Estrutura condicional

E se a condição for **falsa?**



Estrutura condicional

E se a condição for **falsa?**



CONDICIONAIS

- desvio condicional simples em C:

```
if (expressão lógica) {
    _____
    _____
    _____
}
```

instruções a executar se expressão verdadeira

CONDICIONAIS

- desvio condicional composto em C:

```
if (expressão lógica) {
    ________
    ________
    ________
} else {
    ________
    ________
    ________

}
```

instruções a executar se expressão verdadeira

instruções a executar se expressão falsa

CONDICIONAIS

Exercício 1: implemente a função *conceito*, que retorna o conceito de um aluno de Programação Estruturada, com base nas notas da p1 e da p2 e no número de faltas.

CONDICIONAIS

Exercício 1: implemente a função *conceito*, que retorna o conceito de um aluno de Programação Estruturada, com base nas notas da p1 e da p2 e no número de faltas.

```
char conceito (double p1, double p2,
               int faltas) {
   ______
   ______
   ______
}
```

(solução na lousa)

CONDICIONAIS

Exercício 2: implemente a função *significadoConceito*, que imprime a interpretação de um conceito.

- O = reprovação por falta
- F = reprovação por aproveitamento insuficiente
- D = compreensão mínima, mas não satisfatória
- C = compreensão mínima satisfatória
- B = boa compreensão e aproveitamento
- A = compreensão e aproveitamento excelentes

CONDICIONAIS

Exercício 2: implemente a função *significadoConceito*, que imprime a interpretação de um conceito.

```
void significadoConceito (char conceito) {
    ______
    ______
    ______
}
```

(solução na lousa)

CONDICIONAIS

Sequência de testes de igualdade

```
if (conceito == 'O') {
   ...
} else if (conceito == 'F') {
   ...
} else if (conceito == 'D') {
   ...
   ...
} else if (conceito == 'A') {
   ...
} else {
   ...
}
```

CONDICIONAIS

switch – substitui sequência de testes de igualdade em uma variável inteira.

```
switch (var) {
  case num1:
    _____
  break;
  case num2:
    _____
  break;
  ...
  default:
    _____
}
```

CONDICIONAIS

Exercício 3: troque o conjunto de *ifs* no exercício 2 por um switch.

LAÇOS DE ITERAÇÃO

Repetição de código:

```
puts("Single Ladies – Beyoncé");

puts("All the single ladies!");
puts("All the single ladies!");
puts("All the single ladies!");
puts("All the single ladies!");
puts("All the single ladies!");
puts("All the single ladies!");
puts("All the single ladies!");
```

LAÇOS DE ITERAÇÃO

Repetição de código:

```
int i;
puts("Single Ladies – Beyoncé");

for (i = 1; i <= 7; ++i) {
  puts("All the single ladies!");
}
```

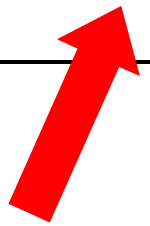
LAÇOS DE ITERAÇÃO

Repetição de código:

```
int i;
puts("Single Ladies – Beyoncé");

for (i = 1; i <= 7; ++i) {

}
```



inicialização
*executada **uma vez** logo antes do início do laço*

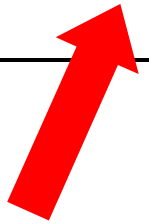
LAÇOS DE ITERAÇÃO

Repetição de código:

```
int i;
puts("Single Ladies – Beyoncé");

for (i = 1; i <= 7; ++i) {

}
```



condição de continuação
*verificada **antes** de inciar **cada** iteração*
se a expressão lógica for verdadeira, executa a iteração

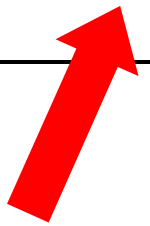
LAÇOS DE ITERAÇÃO

Repetição de código:

```
int i;
puts("Single Ladies – Beyoncé");

for (i = 1; i <= 7; ++i) {

}
```



atualização

*declaração executada **logo após cada** iteração*

LAÇOS DE ITERAÇÃO

Exercício 4: implemente a função *fatorial*, que dado um número *n*, calcula o número *n*! tal que

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$$

LAÇOS DE ITERAÇÃO

Exercício 5: implemente a função *arcotangente*, que calcula uma aproximação para arctan pelo somatório dos *n* primeiros termos abaixo

$$\text{atan}(x) \approx x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots$$

LAÇOS DE ITERAÇÃO ANINHADOS

Exercício 6: Dizemos que um inteiro positivo n pode ser escrito como a soma de dois quadrados se existem a, b inteiros tais que:

$$n = a^2 + b^2$$

Crie uma função somaDeDoisQuadrados(n) que retorna verdadeiro caso n possa ser escrito como a soma de dois quadrados, ou falso caso não possa.

Para os inteiros a e b, precisamos testar todas as possibilidades entre 1 e $\sqrt{n-1}$

LAÇOS DE ITERAÇÃO ANINHADOS

```
int somaDeDoisQuadrados(int n) {
  int a, b;
  for (a = 1; a<=sqrt(n-1); ++a) {
    for (b = 1; b<=sqrt(n-1); ++b {

      if (n == a*a + b*b) {
        return 1;
      } // fim if

    } // fim for (b = ...
  } // fim for (a = ..
  return 0;
}
```

LAÇOS DE ITERAÇÃO – **while** E **do...while**

Além do laço **for**, temos duas outras estruturas de iteração em C:

```
while (cond) {
   ______
   ______
   ______
}
```

expressão é verificada ***logo antes do início de cada iteração***

```
do {
   ______
   ______
   ______
} while (cond);
```

LAÇOS DE ITERAÇÃO – **while** E **do...while**

Além do laço **for**, temos duas outras estruturas de iteração em C:

```
while (cond) {
   ______
   ______
   ______
}
```

```
do {
   ______
   ______
   ______
} while (cond);
```

expressão é verificada
após cada iteração

LAÇOS DE ITERAÇÃO – **while** E **do...while**

Exemplo: algoritmo de euclides para o máximo divisor comum de inteiros positivos.

```
unsigned int mdc(unsigned int a,
                 unsigned int b) {
  while (b != 0) {
    int resto;
    resto = a % b;
    a = b;
    b = resto;
  }
  return a;
}
```