Instruções Condicionais

Instrução “if”

Definição

- A instrução **if** permite o desvio do fluxo vertical de um programa conforme o teste de uma condição (valor lógico)
  - se a condição é verdadeira (true/sim) então o fluxo segue numa direção
  - se a condição é falsa (false/não) então o fluxo segue noutra direção
  - em qualquer um dos casos, após a execução da instrução **if**, o fluxo passa para o mesmo ponto

Sintaxe

- Na linguagem C

```
if (condição)
  instrução_1;
else
  instrução_2;
```

em que,

- **condição** é uma expressão lógica (o resultado é “verdadeiro/sim” ou “falso/não”), a qual tem **obrigatoriamente** que estar **entre parentesis**
- **instrução_1** e **instrução_2** são instruções de qualquer tipo, incluindo instruções “if”

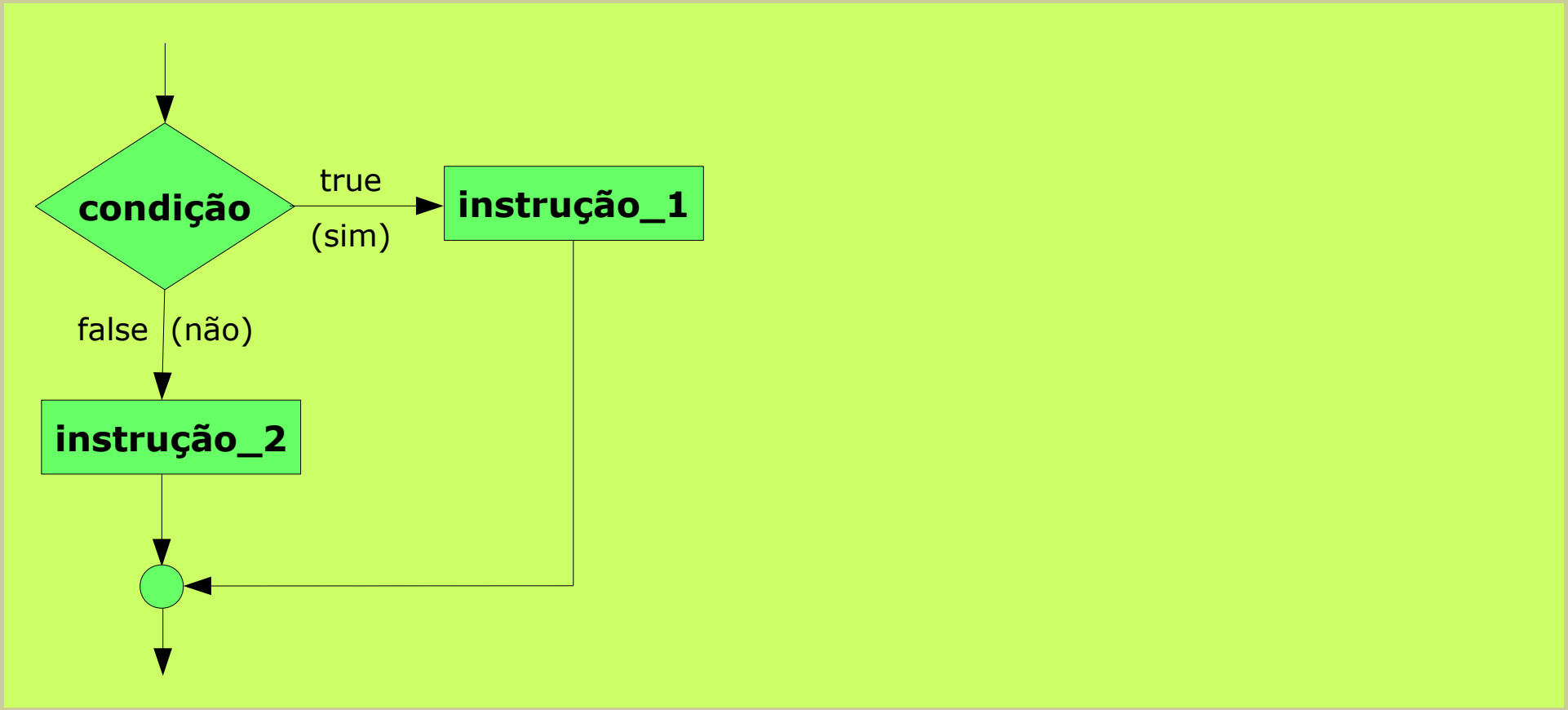
Sintaxe

- Em pseudo-código

```
se <condição> então
   <comandos_1>
senão
   <comandos_2>
fim_se
```

Sintaxe

- Em fluxograma



Funcionamento

- Se o valor da **condição** é *verdadeiro* (true/sim),
  então a **instrução_1** é executada
- Se o valor da **condição** é *falso* (false/não)
  então a **instrução_2** é executada

- Nunca são executadas as duas instruções
  (apenas uma das instruções é executada)

Instrução “if” simples (sem else)

Introdução

- Em determinadas situações, se o resultado (valor lógico) de uma condição for:
  - verdadeiro, então é necessário executar uma instrução,
  - falso, então nada deve ser feito (nenhuma instrução deve ser executada)

Sintaxe

- Na linguagem C

```
if (condição)
  instrução;
```

em que,

- **condição** é uma expressão lógica (o resultado é “verdadeiro/sim” ou “falso/não”), a qual tem **obrigatoriamente** que estar **entre parentesis**
- **instrução** é de qualquer tipo, incluindo uma instrução “if”

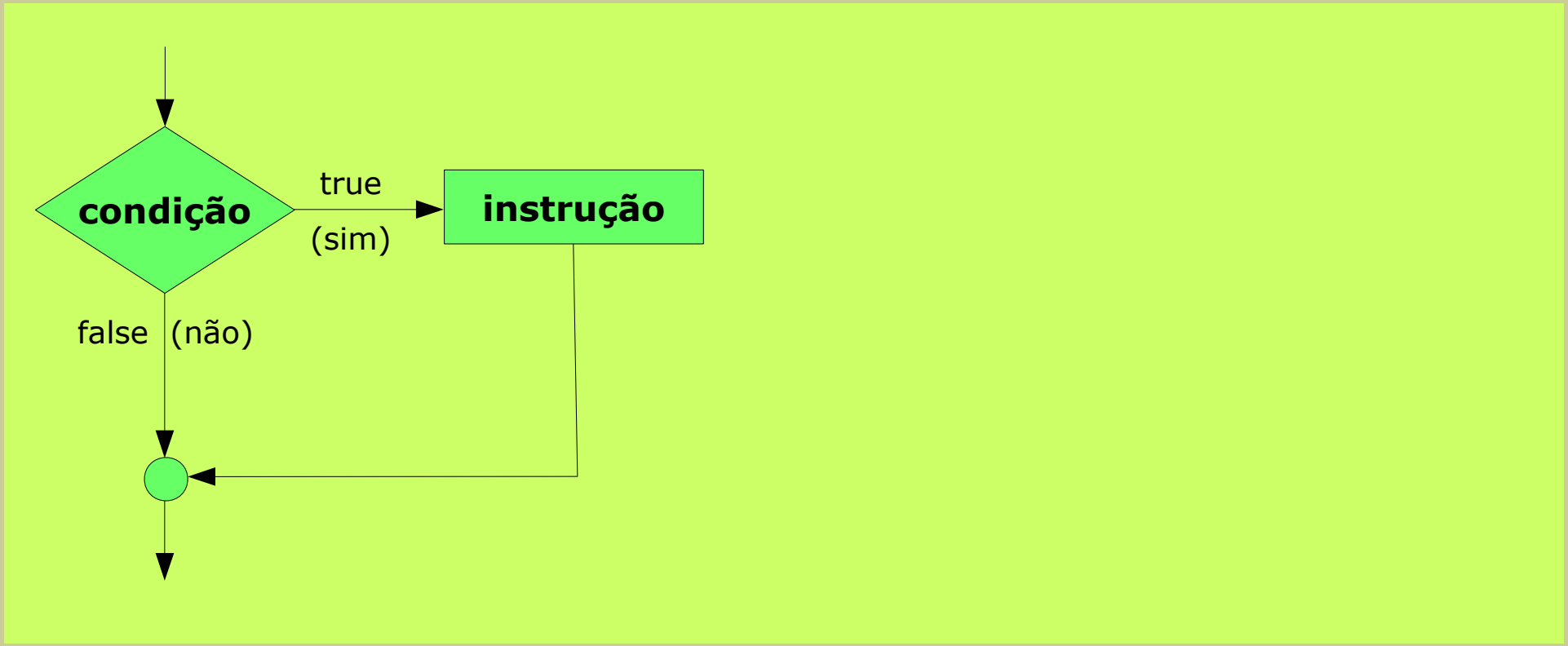
Sintaxe

- Em pseudo-código

```
se <condição> então
   <comandos>
fim_se
```

Sintaxe

- Em fluxograma



Funcionamento

- Se o valor da **condição** for verdadeiro (true/sim),
  então a **instrução** é executada
- Se o valor da **condição** for falso (false/não),
  então nenhuma instrução é executada

Bloco de instruções

Pergunta

- Considere-se a sintaxe da instrução **if** simples:

```
if (condição)
   instrução;
```

- Pretende-se que seja executada várias instruções (e não apenas uma), se a condição da instrução **if** for verdadeira. Como escrever o respetivo código?

Definição

- Um **bloco de instruções** (ou instrução composta), é um conjunto de instruções entre chavetas ({ … })

Sintaxe

```
{
  instrução_1;
  instrução_2;
  …
}
```

- Sintaticamente um *bloco de instruções* equivale a uma única *instrução*
- No *fim* de um *bloco de instruções* não é necessário colocar o ponto e vírgula (;)

Exemplificar para a instrução if

Considere-se a sintaxe da instrução **if** simples:

```
if (condição)
  instrução;
```

- Se na instrução **if** simples for necessário executar várias instruções no lugar da **instrução** deve-se usar um **bloco de instruções**

```
if (condição)
{
  instrução_1;
  instrução_2;
  …
}
```

- Também na instrução **if**, se for necessário executar várias instruções no lugar da **instrução_1** (ou **instrução_2**) deve-se usar um **bloco de instruções**

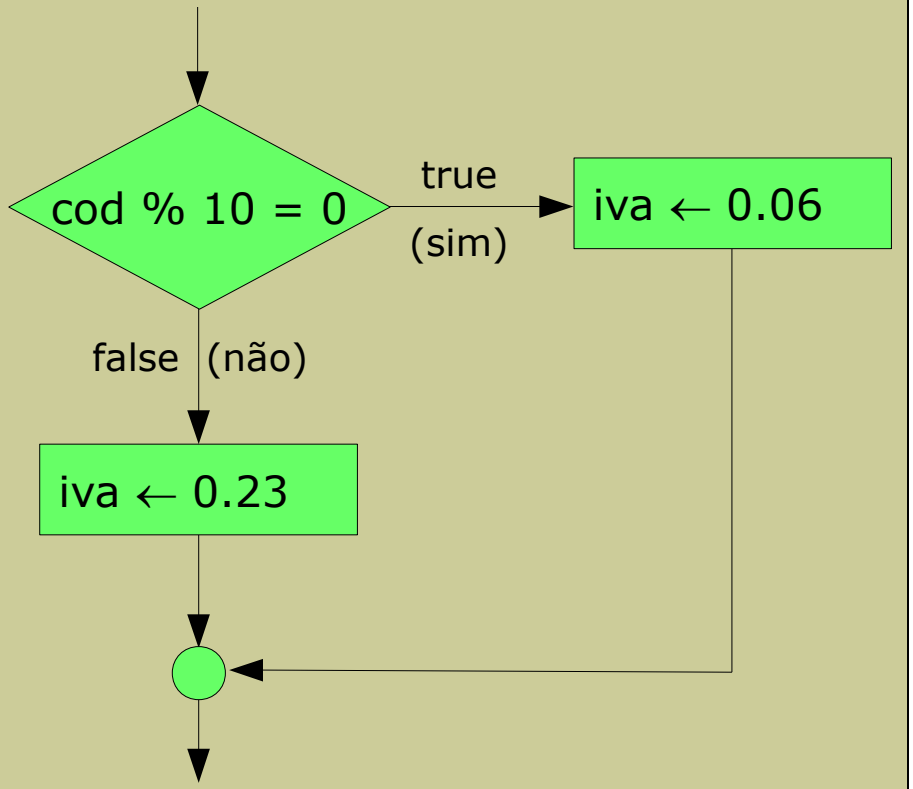
Exemplo

Enunciado

- Elaborar um programa que realize as seguintes ações, pela ordem indicada
  - leia o código de um produto,
  - leia o preço base daquele produto (preço sem iva) e
  - calcula e mostre o preço final (preço com iva)
- Se o código do produto terminar
  - em 0 (zero) o iva é de 6% (6/100 = 0.06)
  - em 1 (um) o iva é de 23% (23/100 = 0.23)

Pseudo-código e fluxograma (parte principal)

```
...
se (cod % 10 = 0) então
   iva ← 0.06
senão
   iva ← 0.23
fim_se
...
```



- De notar que:

  ao dividir um número inteiro positivo por 10,

  o resto da divisão inteira é igual ao

  algarismo das unidades daquele número

  (ex: 2543 % 10 = 3)

Programa

```
#include <stdio.h>
main (){
  int  cod;
  float  iva, pBase, pFinal;
  printf("Escreva o código:\n");
  scanf("%d", &cod);
  printf("Escreva o preço base:\n");
  scanf("%f", &pBase);
  if (cod % 10 == 0)
     iva = 0.06;
  else
     iva = 0.23;
  pFinal = pBase + (pBase * iva);
  printf("Preço final: %f\n", pFinal);
}
```

Programa

```
#include <stdio.h>
main (){
  int  cod;
  float  iva, pBase, pFinal;
  printf("Escreva o código:\n");
  scanf("%d", &cod);
  printf("Escreva o preço base:\n");
  scanf("%f", &pBase);
  if (cod % 10 == 0)
      iva = 0.06;
  else
      iva = 0.23;
  pFinal = pBase + (pBase * iva);
  printf("Preço final: %f\n", pFinal);
}
```

- Questão:

  O que acontece no programa se o código do produto não terminar em 0 nem em 1?

Programa

```
#include <stdio.h>
main () {
  int  cod;
  float  iva, pBase, pFinal;
  printf("Escreva o código:\n");
  scanf("%d", &cod);
  printf("Escreva o preço base:\n");
  scanf("%f", &pBase);
  if (cod % 10 == 0)
      iva = 0.06;
  else
      iva = 0.23;
  pFinal = pBase + (pBase * iva);
  printf("Preço final: %f\n", pFinal);
}
```

- Questão:

  O que acontece no programa se o código do produto não terminar em 0 nem em 1?

- Resposta:

  Para o programa, ser diferente de 0 é equivalente a ser igual a 1

- Comentário:

  Não sendo uma solução correta, como resolver este problema?

Programa (solução 1)

```
#include <stdio.h>
main () {
  int  cod;
  float  iva, pBase, pFinal;
  ...    // leitura de código e preço base
  if (cod % 10 == 0)
     iva = 0.06;
  else
     iva = 0.23;
  pFinal = pBase + (pBase * iva);
  if (cod % 10 != 0 && cod % 10 != 1)
     printf("Código errado!\n");
  else
     printf("Preço final: %f\n", pFinal);
}
```

Programa (solução 2)

```
#include <stdio.h>
main () {
  int  cod;       float  iva, pBase, pFinal;
  ...     // leitura de código e preço base
  if (cod % 10 != 0 && cod % 10 != 1)
      printf("Código errado!\n");
  else {
      if (cod % 10 == 0)
          iva = 0.06;
      else
          iva = 0.23;
      pFinal = pBase + (pBase * iva);
      printf("Preço final: %f\n", pFinal);
  }
}
```

Programa (solução 2)

```
#include <stdio.h>
main () {
  int  cod;      float  iva, pBase, pFinal;
  ...    // leitura de código e preço base
  if (cod % 10 != 0 && cod % 10 != 1)
     printf("Código errado!\n");
  else {
     if (cod % 10 == 0)
        iva = 0.06;
     else
        iva = 0.23;
     pFinal = pBase + (pBase * iva);
     printf("Preço final: %f\n", pFinal);
  }
}
```

- Observar que:
  - a **instrução_2** da instrução **if** que aparece primeiro, é um **bloco de instruções**
  - o **bloco de instruções** contém uma instrução **if**
  - ou seja, uma instrução **if** contém uma instrução **if** – instruções **if encadeadas**

Instruções if encadeadas e Condições compostas

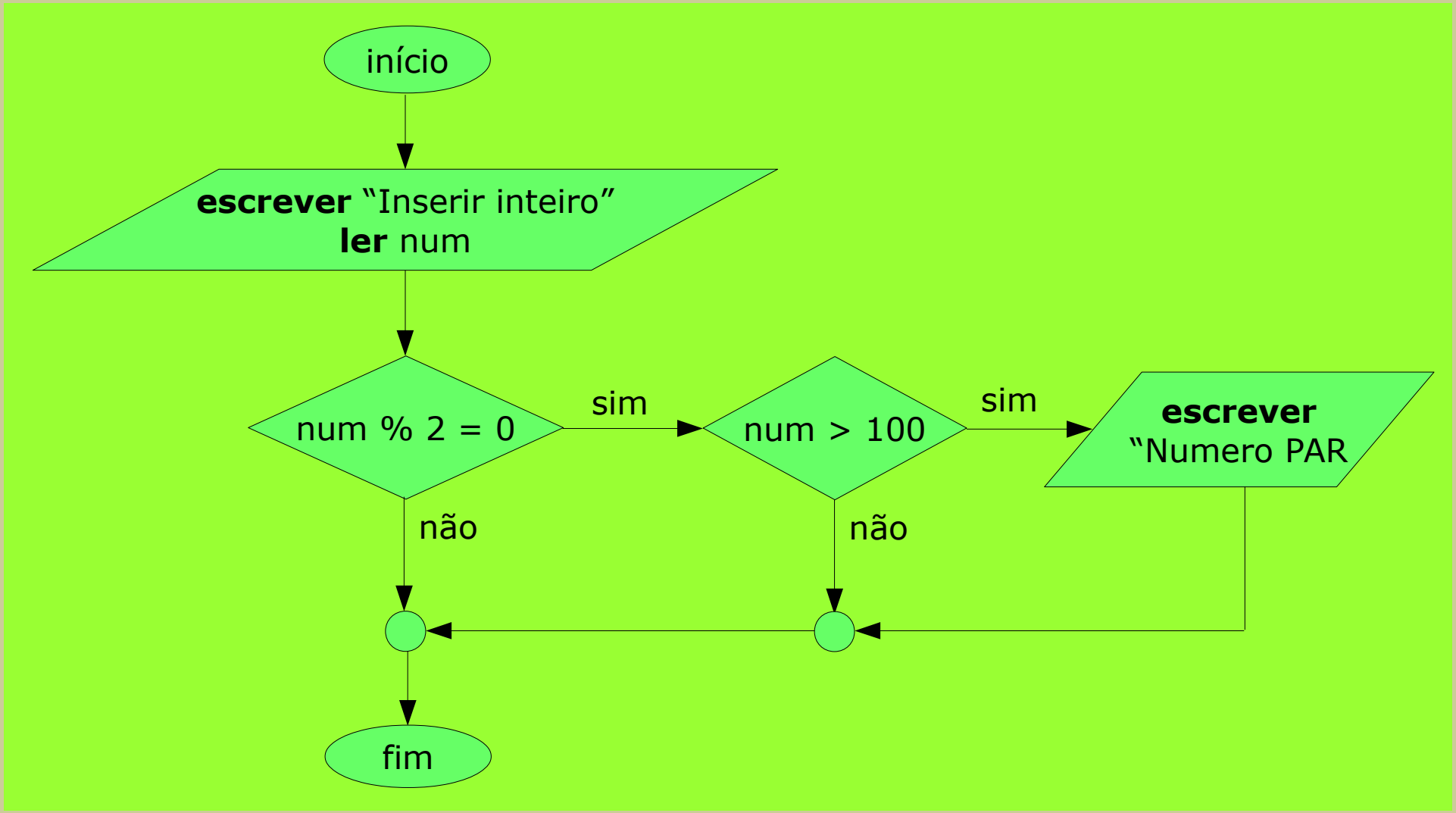
Problema

- Considere-se o seguinte enunciado:
  - inserir um número inteiro
  - apresentar uma mensagem se o número inserido for PAR e maior que 100

Pseudo-código

```
algoritmo NumeroPar
   escrever: "Insira um inteiro"
   ler: num
   se (num % 2 = 0) então
      se (n > 100) então
         escrever: "Numero PAR"
      fim_se
   fim_se
fim_algoritmo
```

Fluxograma


início
escrever "Inserir inteiro"
ler num
num % 2 = 0
sim
num > 100
sim
escrever
"Numero PAR
não
não
fim

Programa

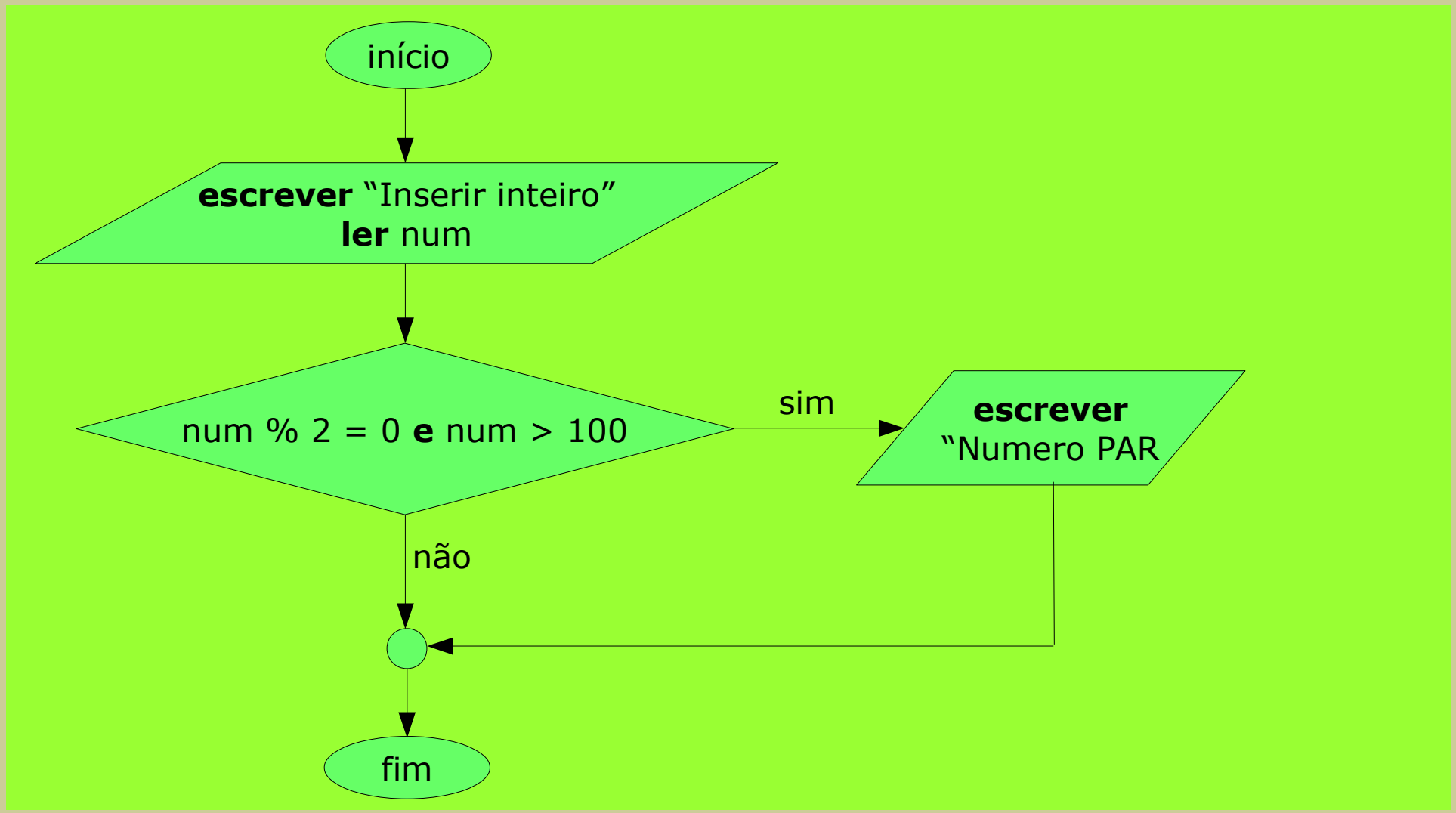
```
#include <stdio.h>
main() {
  int  num;
  printf("Insira um numero inteiro: ");
  scanf("%d", &num);
  if (num % 2 == 0)
     if (num > 100)
        printf("Numero PAR\n");
}
```

- Note-se a utilização de uma instrução **if** como instrução de uma instrução **if**
- Neste caso, pode-se usar apenas uma instrução **if** com uma **condição composta**

Pseudo-código

```
algoritmo NumeroPar
   escrever: "Insira um inteiro"
   ler: num
   se (num % 2 = 0 e n > 100) então
      escrever: "Numero PAR"
   fim_se
fim_algoritmo
```

Fluxograma


início
escrever "Inserir inteiro"
ler num
num % 2 = 0 e num > 100
sim
escrever
"Numero PAR
não
fim

Programa

```
#include <stdio.h>
main() {
   int   num;
   printf("Insira um numero inteiro: ");
   scanf("%d", &num);
   if (num % 2 == 0  &&  num > 100)
        printf("Numero PAR\n");
}
```

Exercício

Enunciado

- Implementar um programa que:
  - leia um número inteiro
  - se o número for par e maior que 100, então escrever “verde”
  - se o número for apenas par e menor ou igual a 100, então escrever “amarelo”
  - se o número for apenas impar, escrever “vermelho”

Regra importante

Considerar o bloco

```
...
if (condição_1)
if (condição_2)
instrução_1;
else
instrução_2;
...
```

Pergunta

- A qual das instruções **if** (**azul** ou **vermelha**) pertence o único **else**?

Resposta

- Um dado **else** pertence sempre ao último **if** (em condições normais)
- A situação anterior ficaria da seguinte forma:

```
...
if (condição_1)
   if (condição_2)
      instrução_1;
   else
      instrução_2;
...
```

Pergunta

- E se, pela lógica do programa, o **else** devesse pertencer ao *primeiro* **if**?

  Como ficaria o bloco de código anterior?

Resposta

- Deveria utilizar-se um bloco de instruções (instrução composta), contendo a segunda instrução if
- A situação anterior ficaria da seguinte forma:

```
...
if (condição_1)
{
  if (condição_2)
      instrução_1;
}
else
  instrução_2;
...
```