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

- Linguagem C

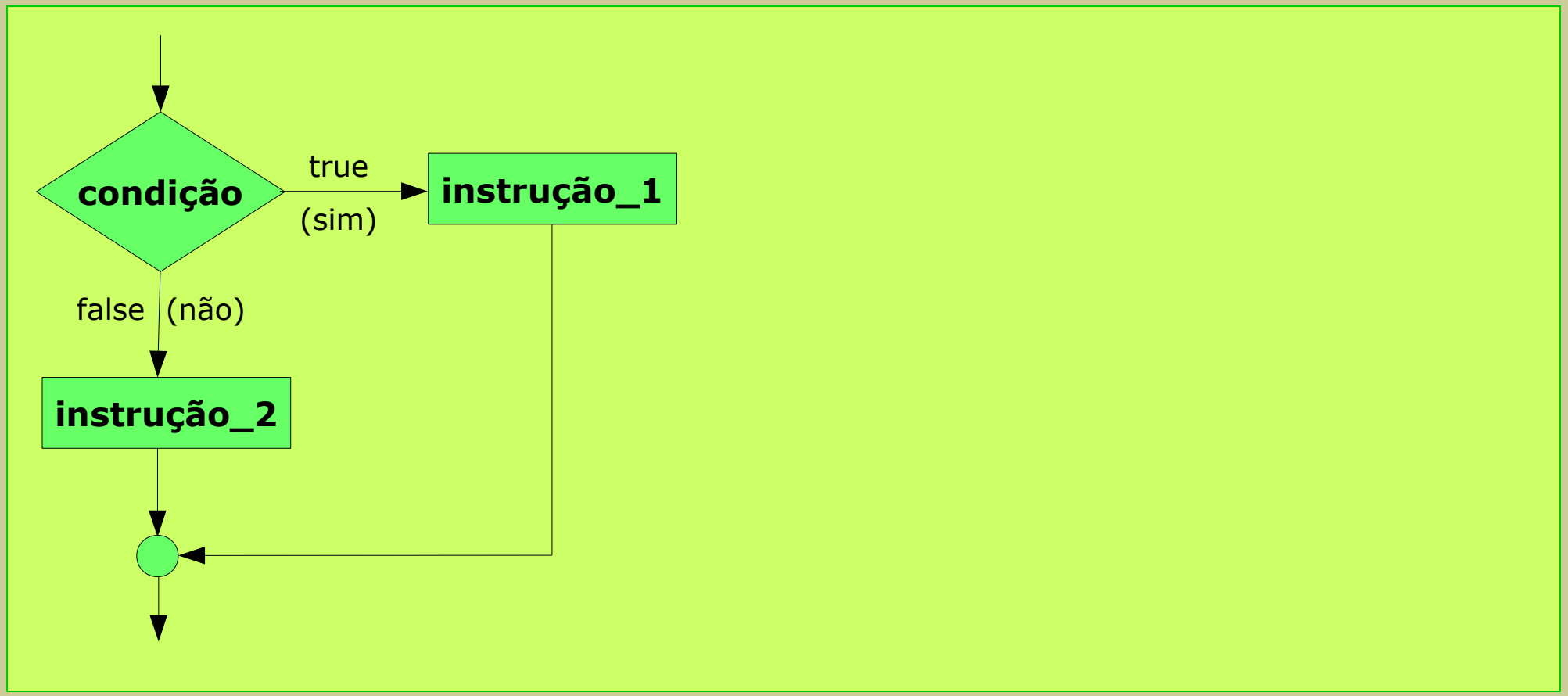
```
if (condição)
  instrução_1;
else
  instrução_2;
```

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

- Fluxograma



Sintaxe

- Pseudocódigo

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

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

- Linguagem C

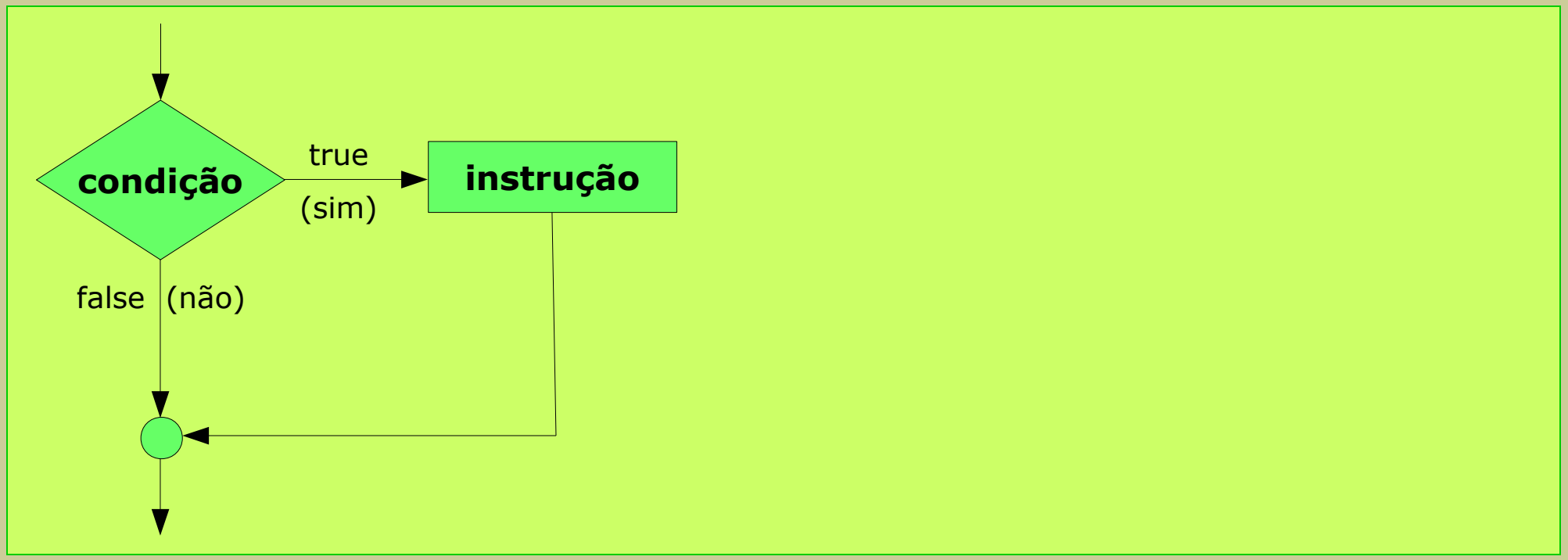
```
if (condição)
  instrução;
```

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

- Fluxograma



Sintaxe

- Pseudocódigo

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

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** 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;
  ...
}
```

Exemplo

Enunciado

Construir um programa que realize as seguintes ações (pela ordem indicada):

- insira/leia o código de um produto,
- insira/leia o preço base daquele produto (preço sem IVA)
- calcula e mostre o preço final (preço com IVA)

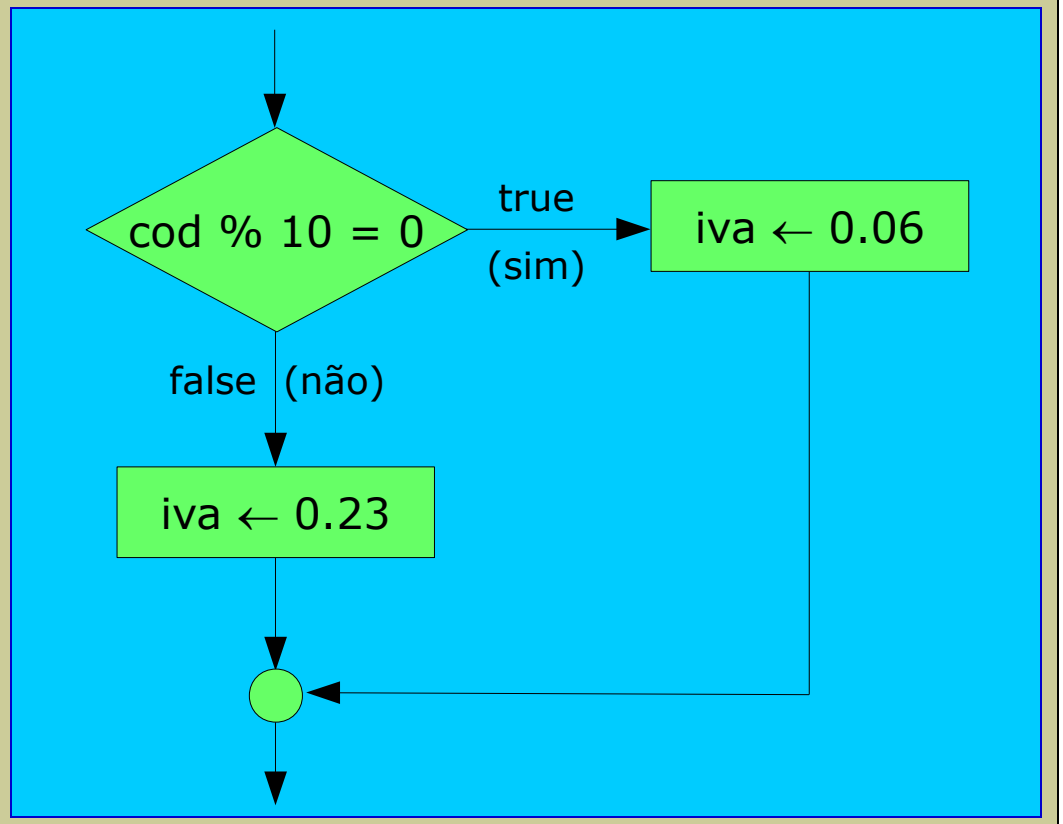
Qualquer produto está sujeito a uma taxa de IVA de 6% ou 23% (não há de 13%).

Assim, se o código do produto termina em

- 0 (zero) o IVA é de 6%
- 1 (um) o IVA é de 23%

Algoritmo (pseudocódigo e fluxograma) - parte principal

```
...
se (cod % 10 = 0) então
   iva ← 0.06     { 6/100 = 0.06 }
senão
   iva ← 0.23     { 23/100 = 0.23 }
fim_se
...
```



- Notar que
  - o resto da divisão inteira de um número inteiro positivo por 10 é igual ao algarismo das unidades daquele número (ex: 2543 % 10 = 3)

Programa

```
#include <stdio.h>
void main ()
{
  int  cod;
  float  iva, pBase, pFinal;
  printf("Insira o código do produto:\n");
  scanf("%d", &cod);
  printf("Insira 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>
void main ()
{
  int  cod;
  float  iva, pBase, pFinal;
  printf("Insira o código do produto:\n");
  scanf("%d", &cod);
  printf("Insira 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>
void main ()
{
  int  cod;
  float  iva, pBase, pFinal;
  printf("Insira o código do produto:\n");
  scanf("%d", &cod);
  printf("Insira 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>
void 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>
void 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>
void 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

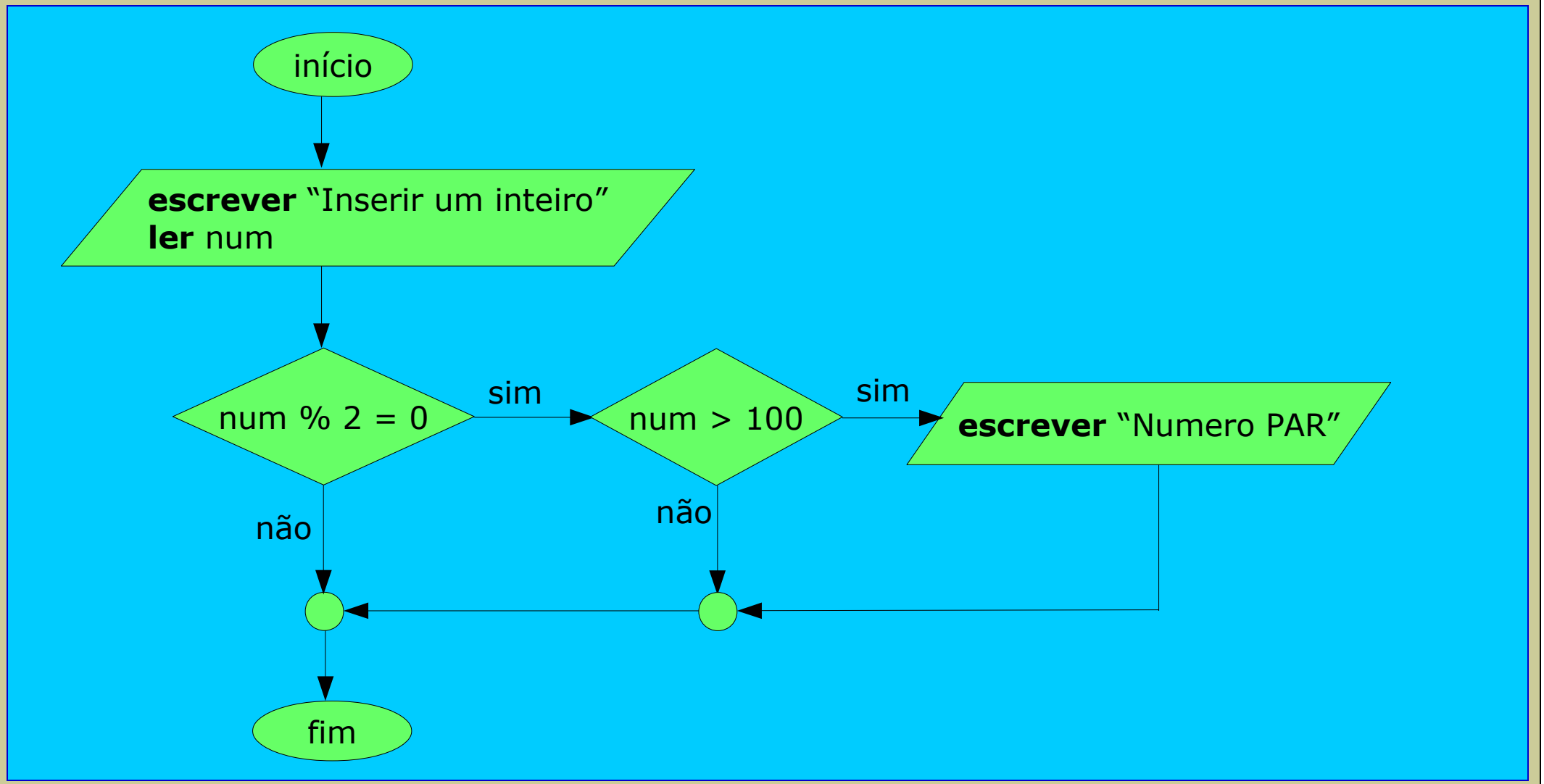
Instruções “if” encadeadas

Problema

- Enunciado

> Inserir um número inteiro.
>
> Apresentar uma mensagem se o número inserido for PAR e maior que 100.

Algoritmo (fluxograma)



Algoritmo (pseudocódigo)

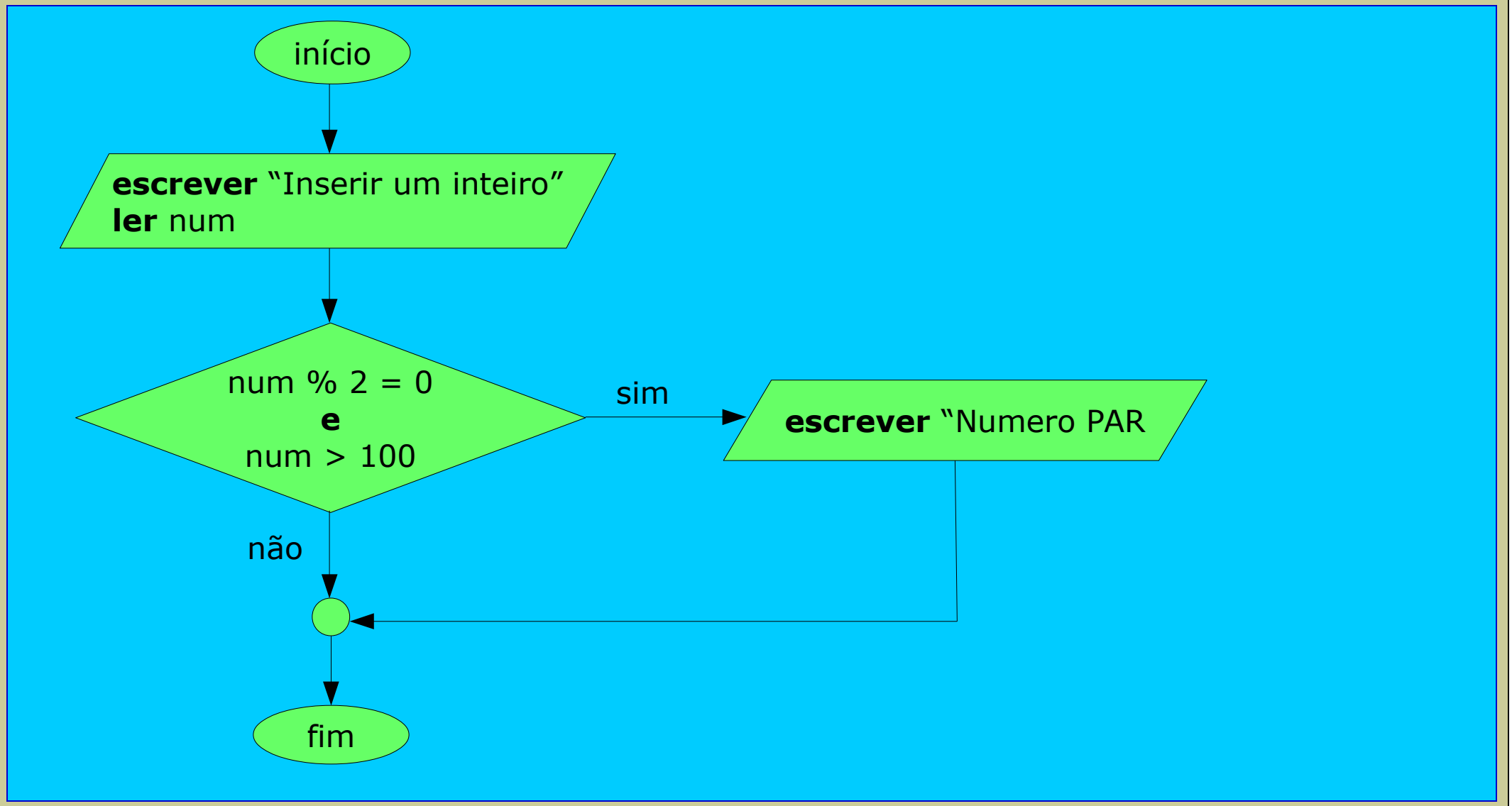
```
algoritmo numeroPar
  escrever: “Insira um inteiro”
  ler: num
  se (num % 2 = 0) então
     se (num > 100) então
        escrever: “Numero PAR”
     fim_se
  fim_se
fim_algoritmo
```

Programa

```
#include <stdio.h>
void 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**

Algoritmo (fluxograma)



Algoritmo (pseudocódigo)

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

Programa

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

Exercício 1

- 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 par e menor ou igual a 100, então escrever “AMARELO”
- se o número for apenas impar, então escrever “VERMELHO”

Programa 1

```
#include <stdio.h>
void main ()
{
  int   num;
  printf(“Insira um numero inteiro: “);
  scanf(“%d”, &num);
  if (num % 2 == 0)
     if (num > 100)
        printf(“VERDE\n”);
     else
        printf(“AMARELO\n”);
  else
     printf(“VERMELHO\n”);
}
```

- Para num = 46, o programa escreve **AMARELO**.

  Sim ou não?

Exercício 2

- 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 par e menor ou igual a 100, então escrever “AMARELO”~~

  (se o número for par e menor ou igual a 100, então não faz nada)
- se o número for apenas impar, então escrever “VERMELHO”

Programa 2

```
#include <stdio.h>
void main ()
{
  int   num;
  printf(“Insira um numero inteiro: “);
  scanf(“%d”, &num);
  if (num % 2 == 0)
     if (num > 100)
        printf(“VERDE\n”);
     else
        printf(“AMARELO\n”);
  else
     printf(“VERMELHO\n”);
}
```

- Este programa faz o pretendido?
- Para num = 46, o programa escreve o quê?

Regra importante

- Considerar o bloco

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

- Questão:

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

Regra importante

- 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;
...
```

- Perguntas:

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

  Como ficaria o bloco de código anterior?

Regra importante

- Resposta
  - deveria utilizar-se um bloco de instruções contendo a segunda instrução “**if**”
  - a situação anterior ficaria da seguinte forma:

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

Exercício 2

- 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 par e menor ou igual a 100, então escrever “AMARELO”~~

  (se o número for par e menor ou igual a 100, então não faz nada)
- se o número for apenas impar, então escrever “VERMELHO”

Exercício 2

- Programa (anterior)

```
#include <stdio.h>
void main ()
{
  int   num;
  printf(“Insira um numero inteiro: “);
  scanf(“%d”, &num);
  if (num % 2 == 0)
     if (num > 100)
        printf(“VERDE\n”);
  else
     printf(“VERMELHO\n”);
}
```

- **Este programa este incorreto (não faz o pretendido)**
- Para num = 46, o programa escreve **VERMELHO**

Exercício 2

- Programa (correto)

```
#include <stdio.h>
void main ()
{
  int   num;
  printf(“Insira um numero inteiro: “);
  scanf(“%d”, &num);
  if (num % 2 == 0)
  {
     if (num > 100)
        printf(“VERDE\n”);
  }
  else
     printf(“VERMELHO\n”);
}
```

- Para num = 46, o programa **não faz nada**