INSTITUTO DE INFORMÁTICA
UFRGS

INF01040 – Introdução à Programação

Algoritmos com Seleção

Seleção condicional de fluxo de execução

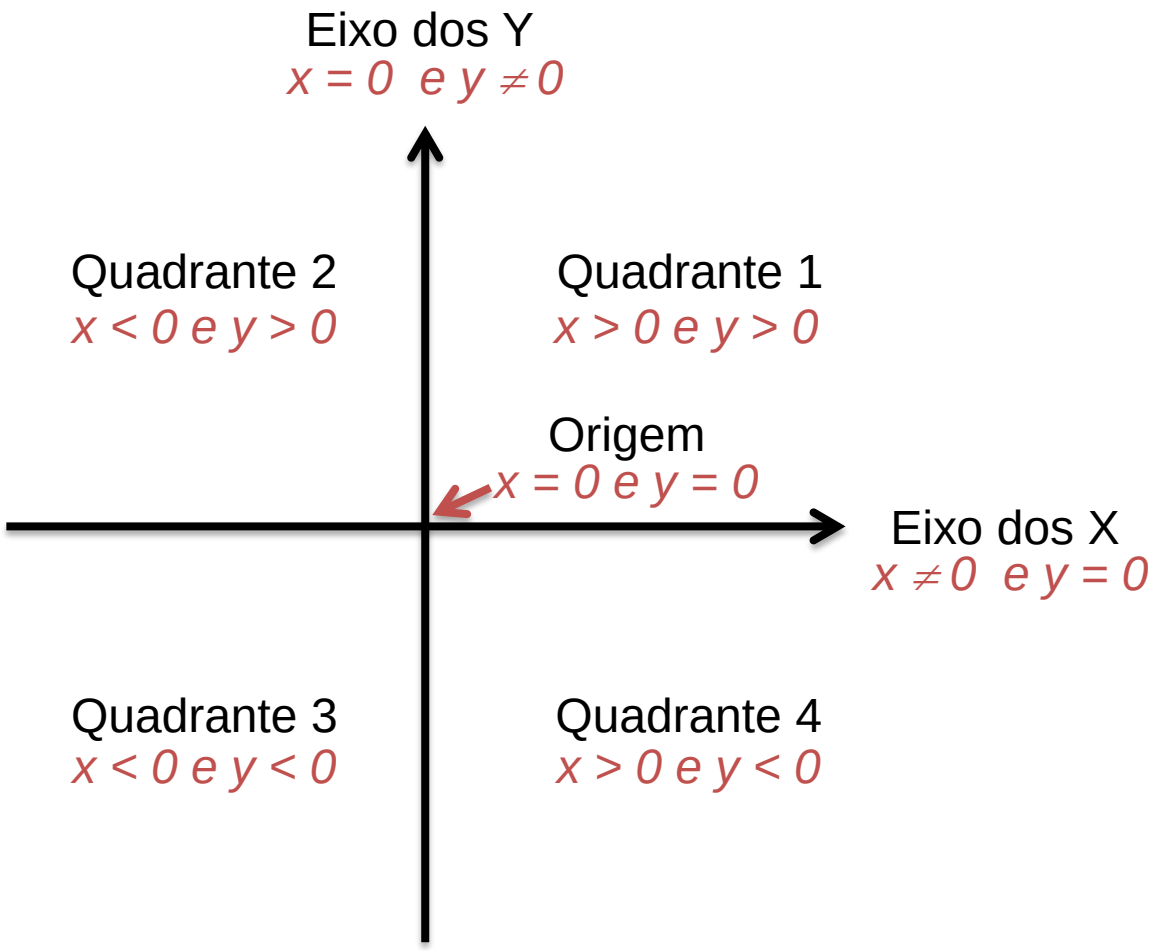
Motivação

- Considerando o seguinte problema:

> Dados um par de valores x e y, que representam as coordenadas de um ponto em um plano, determinar a localização do ponto: se em um quadrante, em um dos eixos ou na origem

- **Entradas:** coordenadas x e y de um ponto
- **Saída:** mensagem adequada
- **Processamento:** série de **testes**, verificando em qual caso o par de valores se enquadra

Motivação

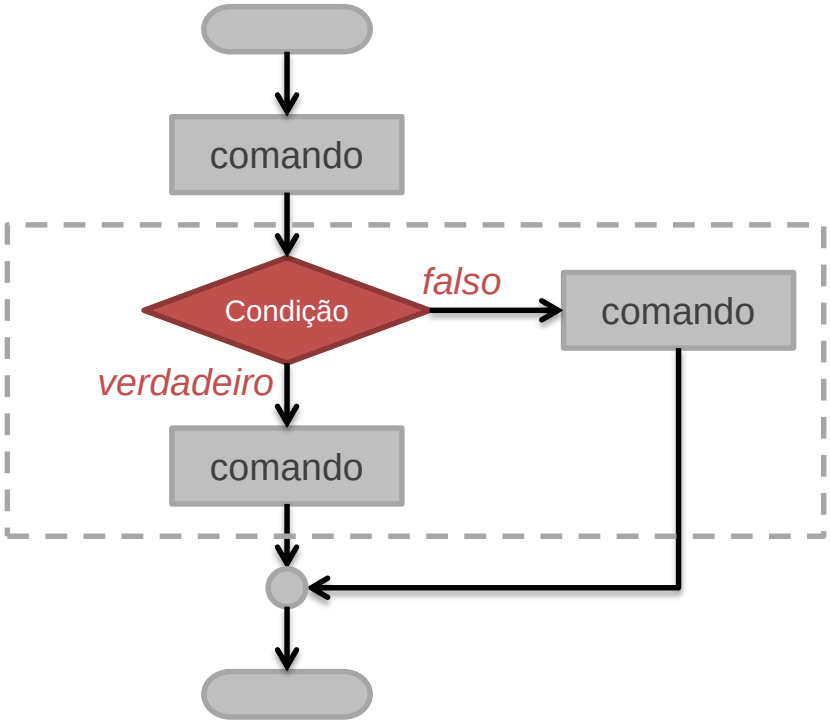

Eixo dos Y
x = 0 e y ≠ 0
Quadrante 2
x < 0 e y > 0
Quadrante 1
x > 0 e y > 0
Origem
x = 0 e y = 0
Eixo dos X
x ≠ 0 e y = 0
Quadrante 3
x < 0 e y < 0
Quadrante 4
x > 0 e y < 0

Passos do Algoritmo

- Leitura dos valores de x e y
- Determinação, pela avaliação de condições, de onde o ponto se encontra:
  - se em um quadrante;
  - se em um eixo;
  - ou se na origem.
- Escrita da mensagem, onde é indicada a localização do ponto

Representando Seleções

- Fluxograma

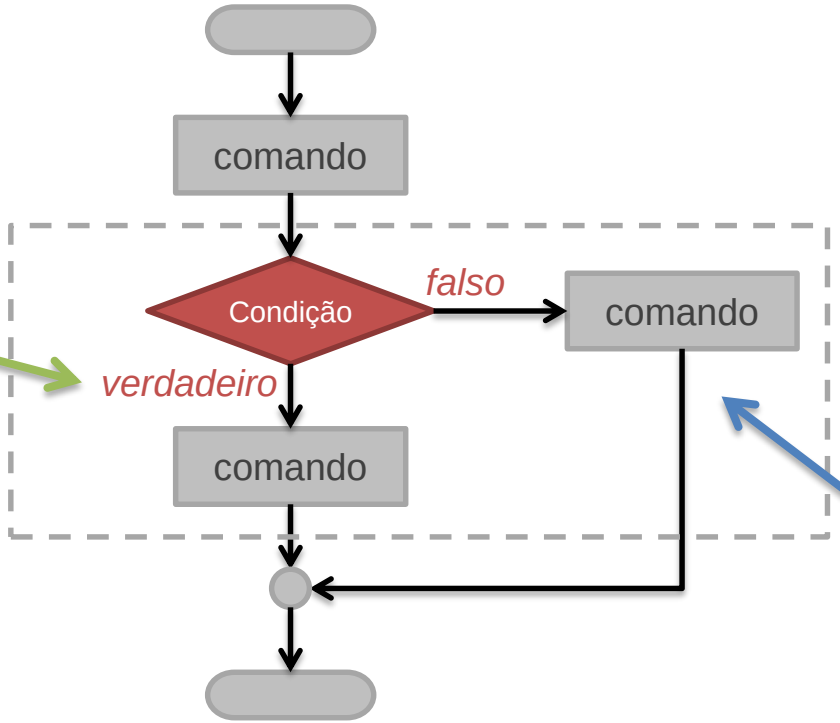


- Pseudo-código

```
1. Principal()
2. Início
3.    Se (condição)
4.       Então
5.          Comandos
6.       Senão
7.          Comandos
8.       Fim
9. Fim
```

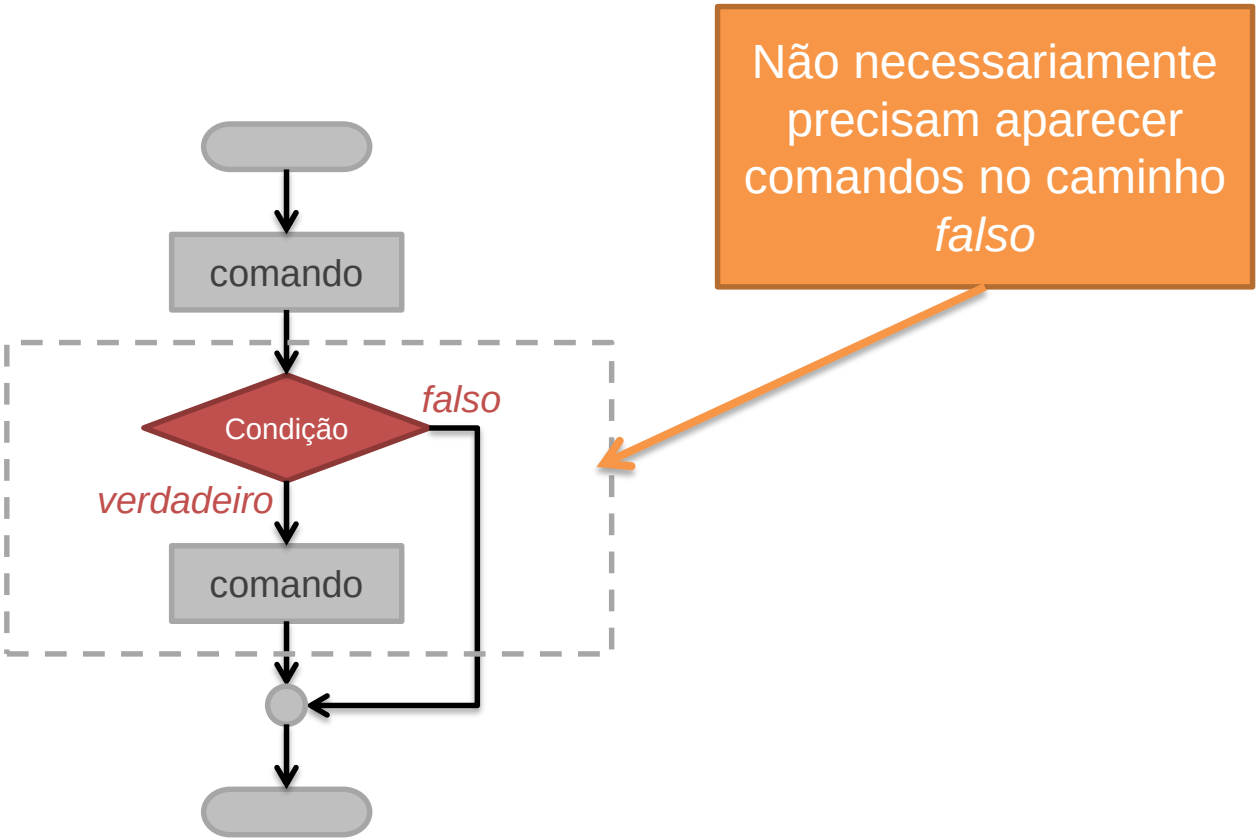
Representando Seleções

Um dos caminhos só executa caso a condição seja verdadeira



O outro caminho executa no caso contrário

Representando Seleções



Representando Seleções

- Com senão

```
1. Principal()
2. Início
3.    Se (condição)
4.       Então
5.          Comandos
6.       Senão
7.          Comandos
8.       Fim
9. Fim
```

- Sem senão

```
1. Principal()
2. Início
3.    Se (condição)
4.       Então
5.          Comandos
6.       Fim
7. Fim
```

Caso Falso

Caso Verdadeiro

Voltando ao problema dos quadrantes...

- Inicialmente vamos considerar a versão simplificada do problema, como segue:

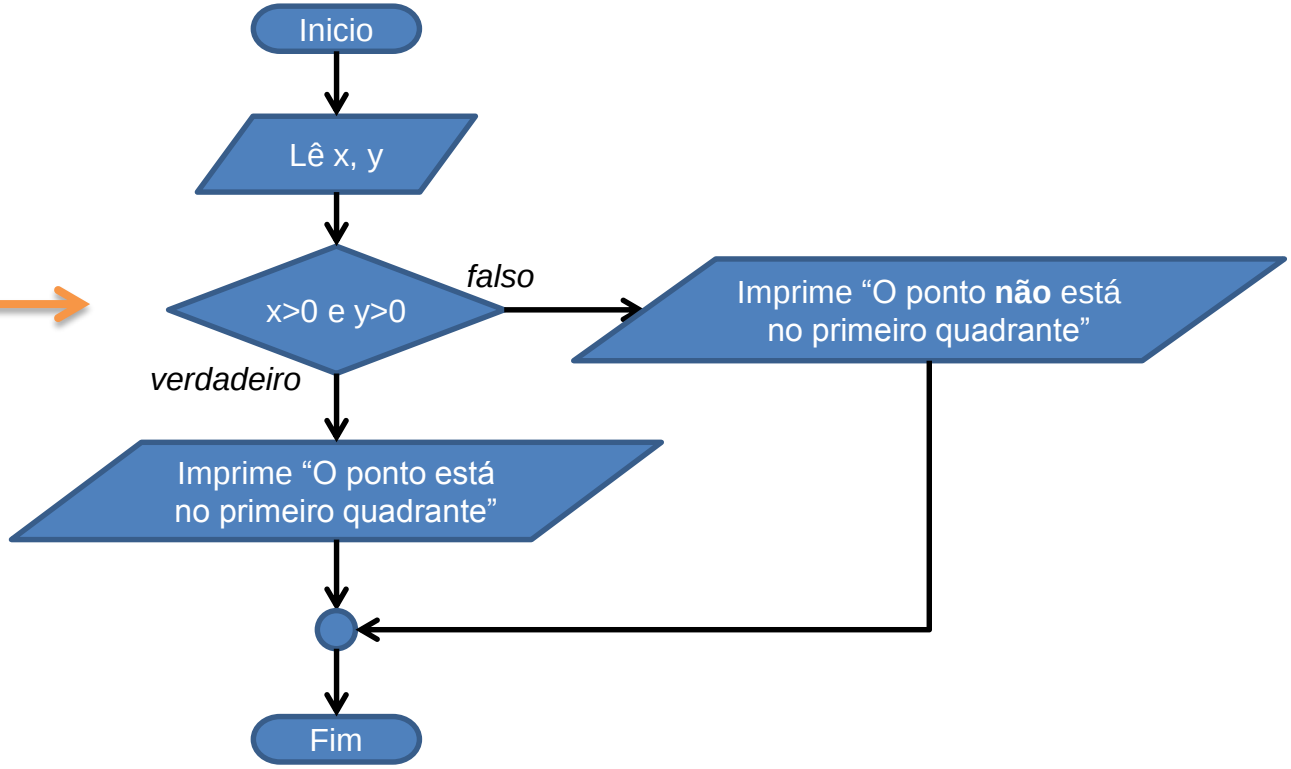
Dados um par de valores x e y, que representam as coordenadas de um ponto em um plano, determinar se o ponto informado **está no primeiro quadrante**

- **Entradas:** coordenadas x e y de um ponto
- **Saída:** mensagem adequada
- **Processamento:** um teste para verificar se o ponto está no primeiro quadrante ou não

O fluxograma do algoritmo para o problema simplificado ficaria assim:

Condição

Se tanto x quando y forem valores positivos, então o ponto está no primeiro quadrante



Entendendo condições lógicas

- Algoritmos com seleção são aqueles em que a execução de determinados passos está subordinada a uma **condição lógica**

Uma **condição lógica** é uma expressão cujo resultado é **verdadeiro** ou **falso**

Exemplos de condições lógicas

- Expressões lógicas simples

se (x) ...

– Verdadeiro se o valor de x for verdadeiro

se (temperatura > 0) ...

– Verdadeiro se o valor de temperatura for positivo

se (h ≠ 5.5) ...

– Verdadeiro se o valor de h for diferente de 5.5

Exemplos de condições lógicas

- Expressões lógicas podem ser compostas de várias partes através do uso de **operadores lógicos**

se (temperatura > 0 e temperatura != 30) ...

– Verdadeiro se a temperatura for positiva e diferente de 30 simultâneamente

se (h = 12 ou p = 12 ou q = 12) ...

– Verdadeiro caso qualquer uma das variáveis h, p ou q seja igual a 12

Operadores Lógicos e Relacionais

- Uma condição lógica pode ser expressa através de expressões relacionais e/ou lógicas
- Essas expressões são descritas, respectivamente, através do uso dos operadores:

Relacionais

e

Lógicos

Operadores Relacionais

Em Fluxogramas e Pseudo-código	Em Linguagem C	Operação
<	<	menor que
>	>	maior que
<=	<=	menor ou igual a
>=	>=	maior ou igual a
=	==	igual a
≠	!=	diferente de

Exemplos:

a > b *x ≠ 0* *y >= 1* *3 <= x*

Operadores Lógicos

Em Fluxogramas e Pseudo-código	Em Linguagem C	Operação
e	&&	E lógico (ambos)
ou	||	Ou lógico (qualquer)
não	!	Negação

Exemplos:

(a < b e b < c) → o valor de *b* está entre *a* e *c*

(x < 0 ou x > 10) → ou o valor de *x* é negativo ou é maior que 10

((k <= 0 ou k >100) e k ≠ -1) → *k* pode ser zero, ou negativo, ou maior que *100*, contanto que não seja igual a *-1*

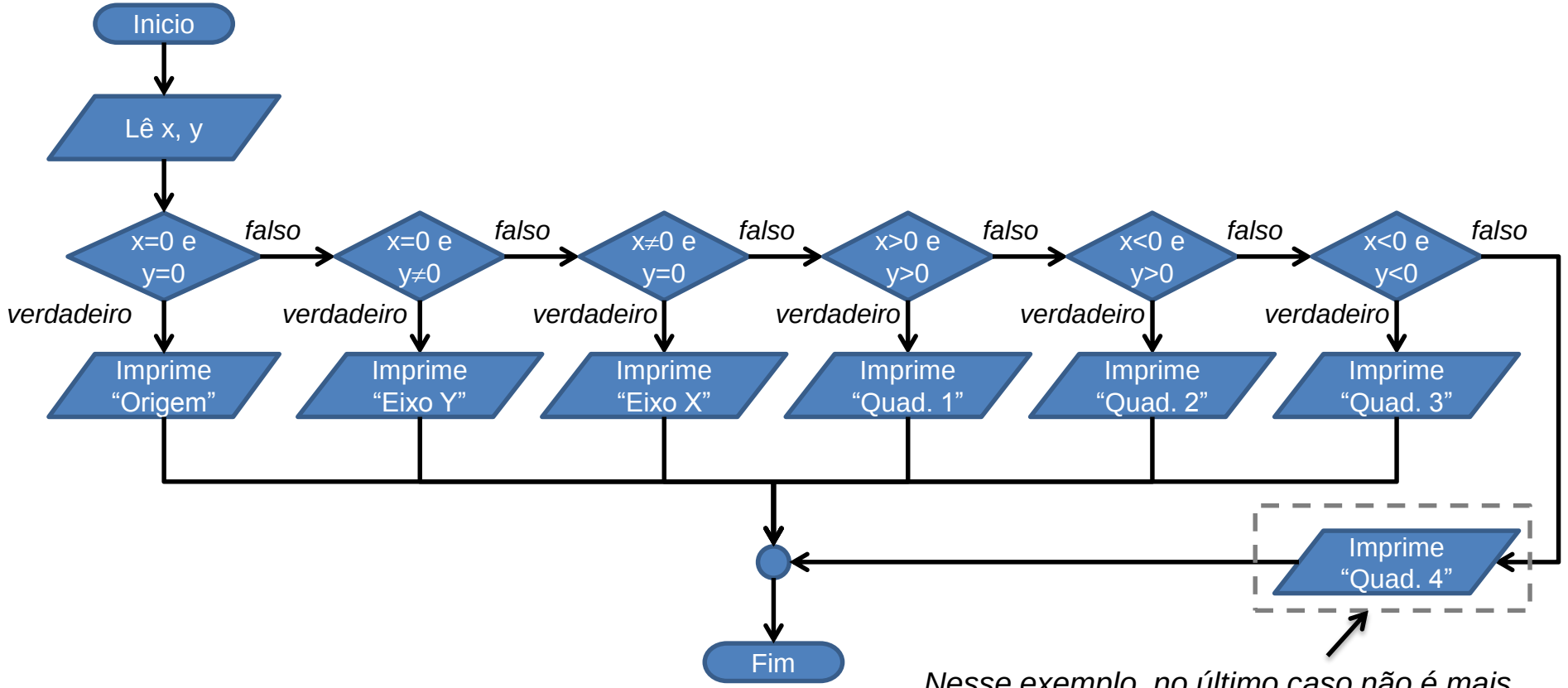
Retomando o problema inicial

- Considerando o seguinte problema:

> Dados um par de valores x e y, que representam as coordenadas de um ponto em um plano, determinar a localização do ponto: se em um quadrante, em um dos eixos ou na origem

- **Entradas:** coordenadas x e y de um ponto
- **Saída:** mensagem adequada
- **Processamento:** série de **testes**, verificando em qual caso o par de valores se enquadra

Fluxograma completo para solução do problema dos quadrantes



Nesse exemplo, no último caso não é mais necessário testar, uma vez que se não foi verdadeiro para nenhuma das outras opções o ponto só pode estar no quadrante.4

... em pseudo-código

```
1.  Principal()
2.  Início
3.    Lê x, y;
4.    Se (x=0 e y=0)
5.    Então
6.      Imprime "Origem";
7.    Senão
8.    Se (x=0 e y≠0)
9.    Então
10.     Imprime "Eixo Y";
11.   Senão
12.   Se (x≠0 e y=0)
13.   Então
14.     Imprime "Eixo X";
15.   Senão
16.   Se (x>0 e y>0)
17.   Então
18.     Imprime "Quad. 1";
19.   Senão
20.   Se (x<0 e y>0)
21.   Então
22.     Imprime "Quad. 2";
23.   Senão
24.   Se (x<0 e y<0)
25.   Então
26.     Imprime "Quad. 3";
27.   Senão
28.     Imprime "Quad. 4";
29.   Fim
30. Fim
```

... em pseudo-código (sem usar senão)

```
1.  Principal()
2.  Início
3.    Lê x, y;
4.    Se (x=0 e y=0)
5.    Então
6.      Imprime "Origem";
7.    Fim
8.    Se (x=0 e y≠0)
9.    Então
10.     Imprime "Eixo Y";
11.   Fim
12.   Se (x≠0 e y=0)
13.   Então
14.     Imprime "Eixo X";
15.   Fim
16.   Se (x>0 e y>0)
17.   Então
18.     Imprime "Quad. 1";
19.   Fim
20.   Se (x<0 e y>0)
21.   Então
22.     Imprime "Quad. 2";
23.   Fim
24.   Se (x<0 e y<0)
25.   Então
26.     Imprime "Quad. 3";
27.   Fim
28.   Se (x>0 e y<0)
29.     Imprime "Quad. 4";
30.   Fim
31. Fim
```

Requer um teste a mais

O comando `if` na linguagem C

```
1. if (condição)
2.    comando;
```

Ou

```
1. if (condição)
2. {
3.    comando1;
4.    comando2;
5.    comando3;
6. }
```

- Funcionamento
  - Se `condição` resultar verdadeira, então o(s) comando(s) após o `if` será(ão) executado(s)
  - Se existe apenas um comando para executar não é necessário colocar chaves
  - Caso dois ou mais comandos sejam executados “dentro” do `if` se usa chaves para delimitar o escopo

Comando `if ... else ...`

```
1.  if (condição)
2.    comando1;
3.  else
4.    comando2;
```

Ou

```
1.  if (condição)
2.  {
3.    comando1;
4.    comando2;
5.  } else {
6.    comando3;
7.    comando4;
8.  }
```

- Funcionamento
  - Se `condição` resultar verdadeira, então o(s) comando(s) após o `if` será(ão) executado(s)
  - Senão, ou seja, se condição resultar em valor falso, serão executados os comandos “dentro” do `else`

Comparando com pseudo-código

```
1. if (condição)
2. {
3.    comandos;
4. }
```

```
1. Se (condição)
2.    Então
3.       Comandos
4.    Fim
```

```
1. if (condição)
2. {
3.    comandos;
4. } else {
5.    comandos;
6. }
```

```
1. Se (condição)
2.    Então
3.       Comandos
4.    Senão
5.       Comandos
6.    Fim
```

Implementado o problema dos quadrantes em C

Com **`if ... else ...`**

```c
#include <stdio.h>

int main(){
    float x,y;

    scanf("%f%f", &x, &y);

    if (x == 0 && y == 0){
        printf("Origem\n");
    }else if(x == 0 && y != 0){
        printf("Eixo Y\n");
    }else if(x != 0 && y == 0){
        printf("Eixo X\n");
    }else if(x > 0 && y > 0){
        printf("Quadrante 1\n");
    }else if(x < 0 && y > 0){
        printf("Quadrante 2\n");
    }else if(x < 0 && y < 0){
        printf("Quadrante 3\n");
    }else{
        printf("Quadrante 4\n");
    }

    return 0;
}
```

Apenas com **`if`**

```c
#include <stdio.h>

int main(){
    float x,y;

    scanf("%f%f", &x, &y);

    if (x == 0 && y == 0){
        printf("Origem\n");
    }
    if(x == 0 && y != 0){
        printf("Eixo Y\n");
    }
    if(x != 0 && y == 0){
        printf("Eixo X\n");
    }
    if(x > 0 && y > 0){
        printf("Quadrante 1\n");
    }
    if(x < 0 && y > 0){
        printf("Quadrante 2\n");
    }
    if(x < 0 && y < 0){
        printf("Quadrante 3\n");
    }
    if(x > 0 && y < 0){
        printf("Quadrante 4\n");
    }

    return 0;
}
```

Algumas particularidades de seleção em C

- Não é permitido "encurtar" expressões lógicas em C, por exemplo:

  ~~`(a < b < c)`~~ ao invés de `(a < b && b < c)`

- O valor **falso** não existe formalmente em C
  - Os valores `0` (zero numérico), `'\0'` (caractere "barra zero") e `NULL` são considerados falsos
  - Qualquer outro valor é **verdadeiro**
- Cuidado! Em C, o operador **=** representa **atribuição**, o **operador lógico** para comparação é **==**
- **Para pensar:** qual das soluções para o problema dos quadrantes é mais **eficiente**, com `if ... else ...` ou apenas com `if`?