

# Algoritmos e Lógica de Programação

Análise e Desenvolvimento de Sistemas






Prof. Dr. Lucas Baggio Figueira

# Objetivos

Compreender e aplicar os operadores aritméticos e lógicos.

Usar o comando de decisão para determinar fluxos de execução baseados em condições pré-determinadas.

# Operadores Aritméticos

Operador	Símbolo	Sintaxe
Adição	 +	$a + b$
Subtração	 -	$a - b$
Multiplicação	 *	$a * b$
Divisão	 /	$a / b$
Módulo (Resto)	 %	$a \% b$

# Exemplo

```
#include<iostream>

using namespace std;

int main() {

    int a = 13, b = 5;

    cout << "a + b = " << a+b << endl;
    cout << "a - b = " << a-b << endl;
    cout << "a * b = " << a*b << endl;
    cout << "a / b = " << a/b << endl;
    cout << "(float)a / b = " << (float)a/b << endl;
    cout << "a % b = " << a%b << endl;

    return 0;

}
```

```
a + b = 18
a - b = 8
a * b = 65
a / b = 2
(float)a / b = 2.600000
a % b = 3
```

# Atenção para o operador /

- O operador de divisão é polimórfico, ou seja, ele opera de acordo com o tipo do dividendo:

- Se o dividendo for do tipo float, então:  $14.0 / 3 = 4.66$

- Se o dividendo for do tipo int, então:  $14 / 3 = 4$

14 % 3

2

# Incremento e Decremento

- Incremento:

```
a = a + 1;
```

Operador de incremento:

```
a++; // pós-fixado
++a; // pré-fixado
```

- Decremento:

```
a = a - 1;
```

Operador de decremento

```
a--; // pós-fixado
--a; // pré-fixado
```

# Operadores Aritméticos com Atribuição

Símbolo	Sintaxe	Substitui
<code>+=</code>	<code>a += b</code>	<code>a = a + b</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a - b</code>
<code>*=</code>	<code>a *= b</code>	<code>a = a * b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a / b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a % b</code>

# Operadores Lógicos

Operadores que resultam é apenas dois valores:

**Verdadeiro / Falso**



# Operadores Lógicos

Operador	Símbolo	Sintaxe
Menor que	<	$a < b$
Menor ou igual que	<=	$a <= b$
Maior que	>	$a > b$
Maior ou igual que	>=	$a >= b$
Diferente de	!=	$a != b$
Igual a	==	$a == b$

# Exemplo

```
#include <iostream>

using namespace std;

int main() {

    int a = 4, b = 5;

    cout << "a < b : " << a<b <<endl;
    cout << "a <= b : " << a<=b <<endl;
    cout << "a > b : " << a>b <<endl;
    cout << "a >= b : " << a>=b <<endl;
    cout << "a != b : " << a!=b <<endl;
    cout << "a == b : " << a==b <<endl;
    cout << "a == 4 : " << a==4 <<endl;
    cout << "b >= 5 : " << b>=5 <<endl;

    return 0;

}
```

```
a < b : 1
a <= b : 1
a > b : 0
a >= b : 0
a != b : 1
a == b : 0
a == 4 : 1
b >= 5 : 1
```

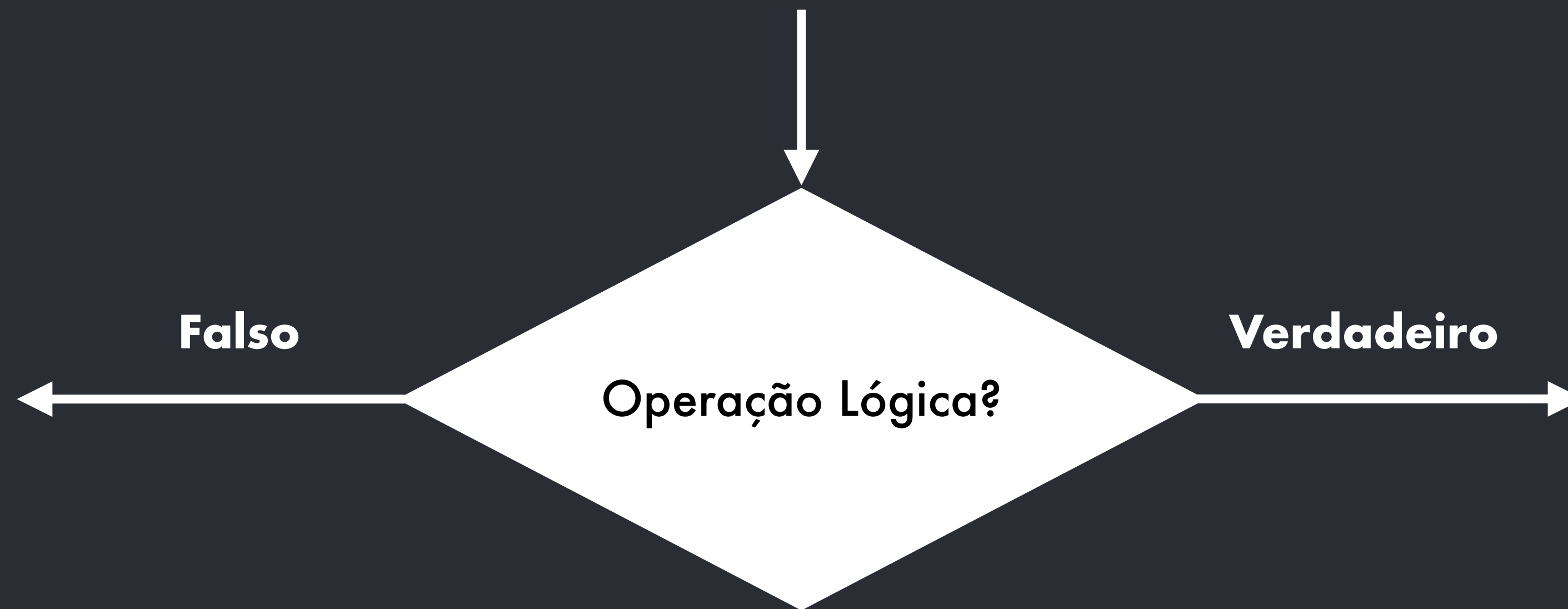
# Operadores Lógicos

P	Q	P && Q	P    Q	!Q
V	V	V	V	F
V	F	F	V	V
F	V	F	V	V
F	F	F	F	V

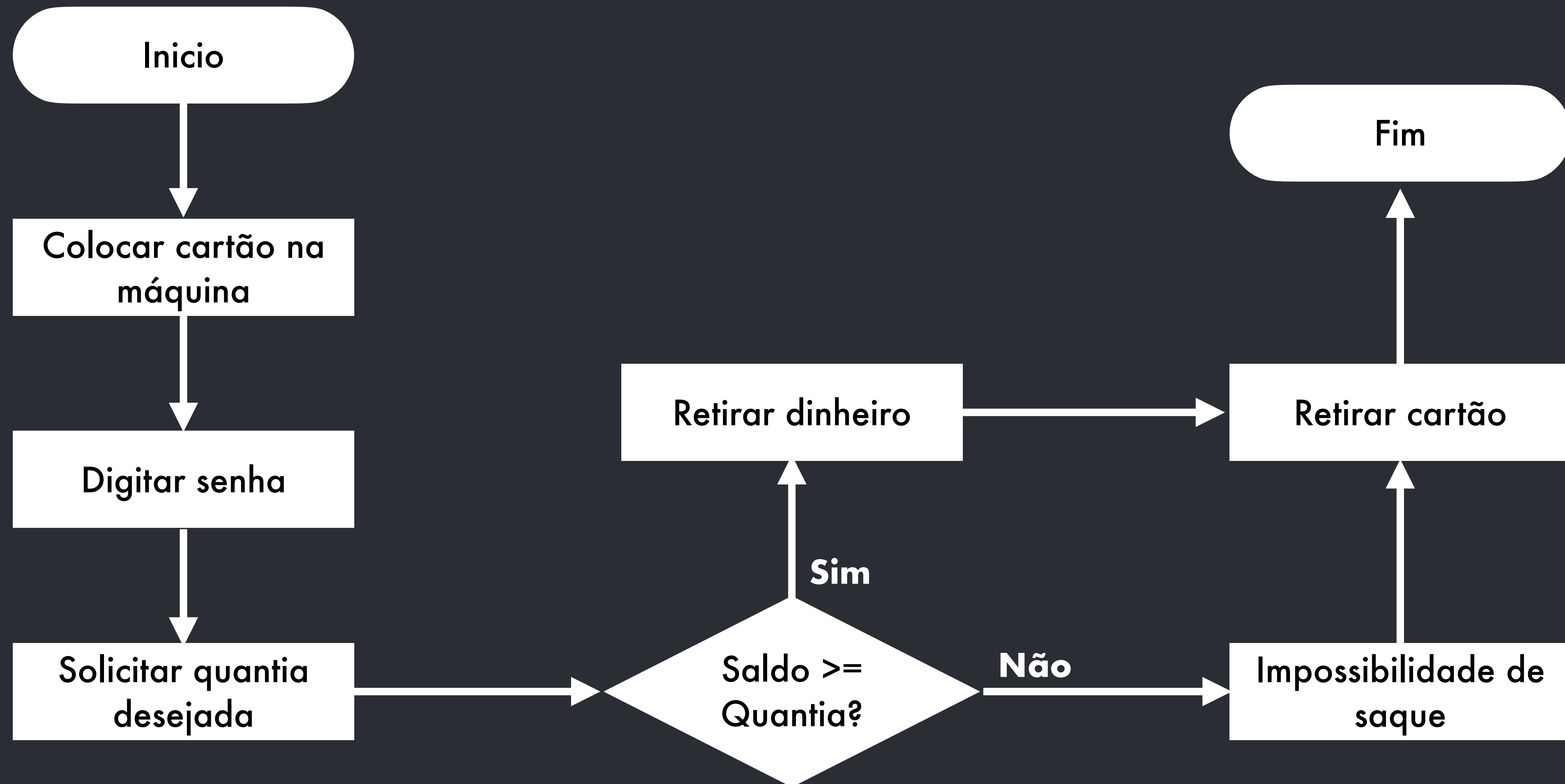
# Estruturas de Decisão

- Por diversas vezes os algoritmos devem se comportar de diferentes maneiras dependendo dos dados fornecidos pelos usuários e/ou gerados pelo processamento do mesmo
- São guiadas por operações lógicas, as quais determinam o fluxo a ser seguido.

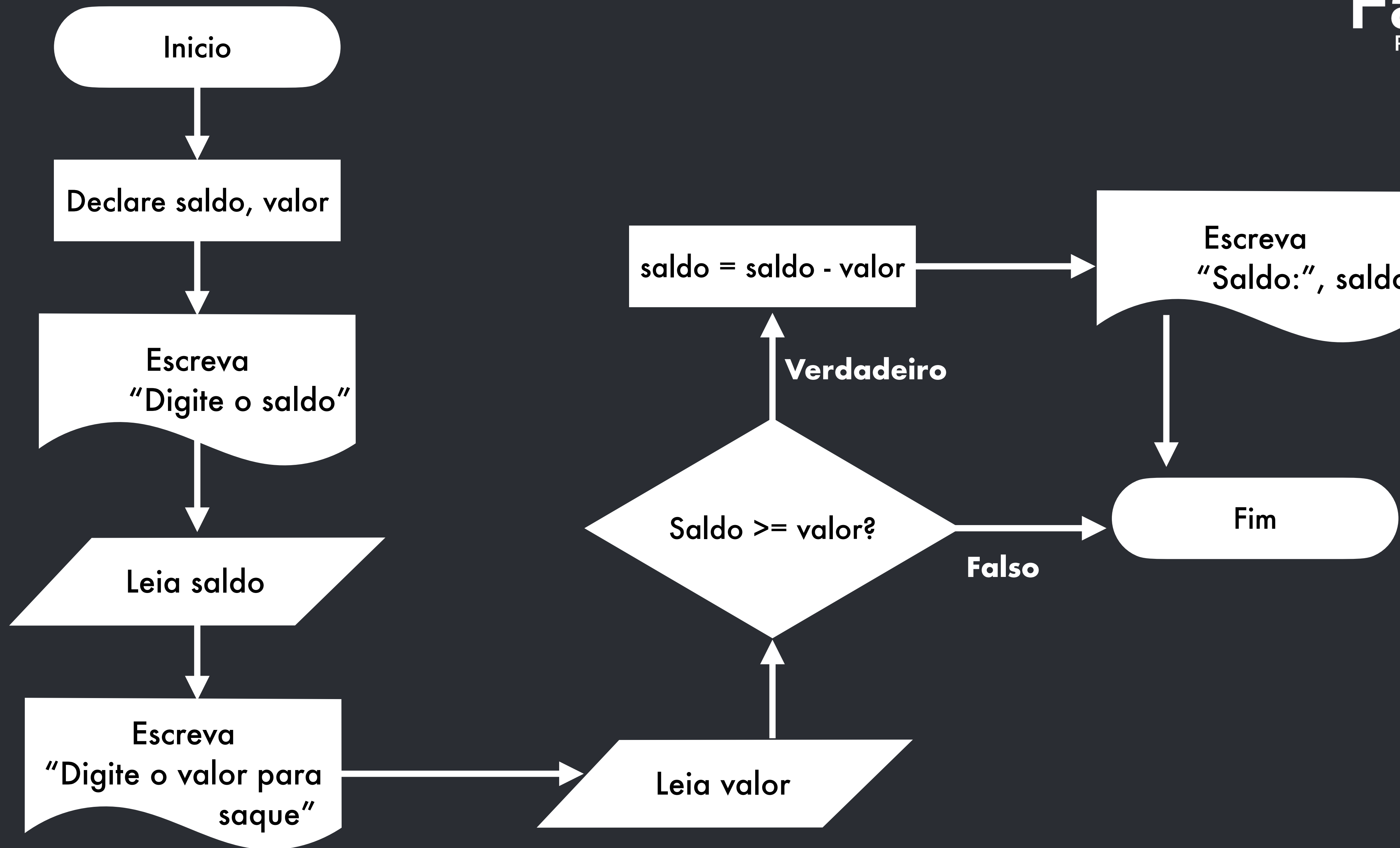
# Estruturas de Decisão



# Sacar \$ no caixa 24 horas



Faça um algoritmo que leia o saldo da conta e o valor que se deseja sacar, a seguir informe se o saque é possível juntamente com o saldo atualizado





# Em Linguagem C++?

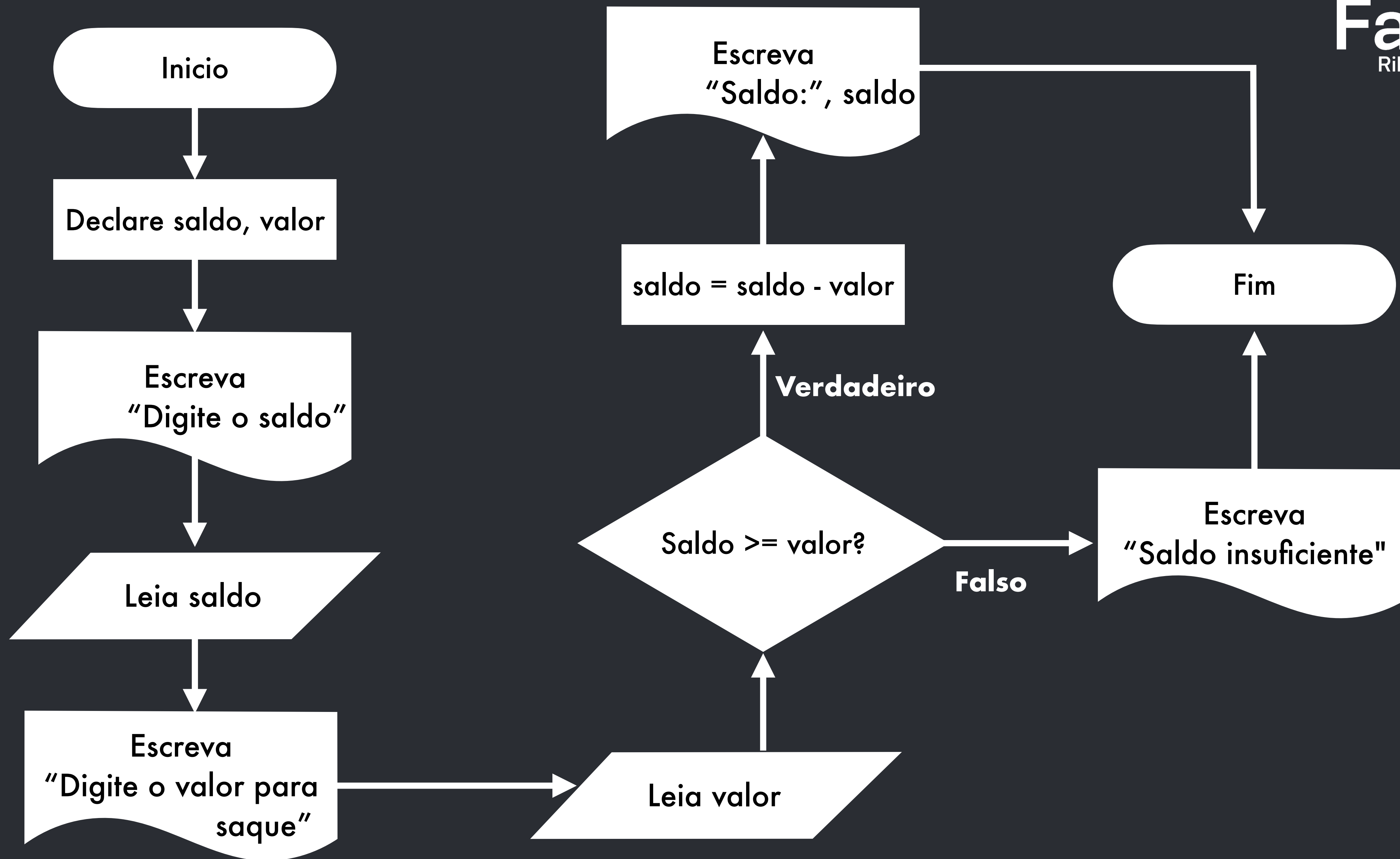
# Estrutura de Decisão - if

```
if (<operação lógica>)  
  <comando>;
```

```
if (<operação lógica>){  
  <comando>;  
  <comando>;  
  .  
  .  
  .  
  <comando>;  
}
```

LET'S CODING ....

**Seria interessante dar uma  
mensagem de negando o saque**



# Estrutura de Decisão - if/else

```
if (<operação lógica>)  
    <comando>;  
else  
    <comando>;
```

```
if (<operação lógica>) {  
    <comando>;  
    .  
    .  
    <comando>;  
}  
else {  
    <comando>;  
    .  
    .  
    <comando>;  
}
```

# Estrutura de Decisão - if/else

```
if (<operação lógica>)  
    <comando>;  
else {  
    <comando>;  
    .  
    .  
    <comando>;  
}
```

```
if (<operação lógica>) {  
    <comando>;  
    .  
    .  
    <comando>;  
}  
else  
    <comando>;
```

LET'S CODING ....



# SWITCH-CASE

# SWITCH

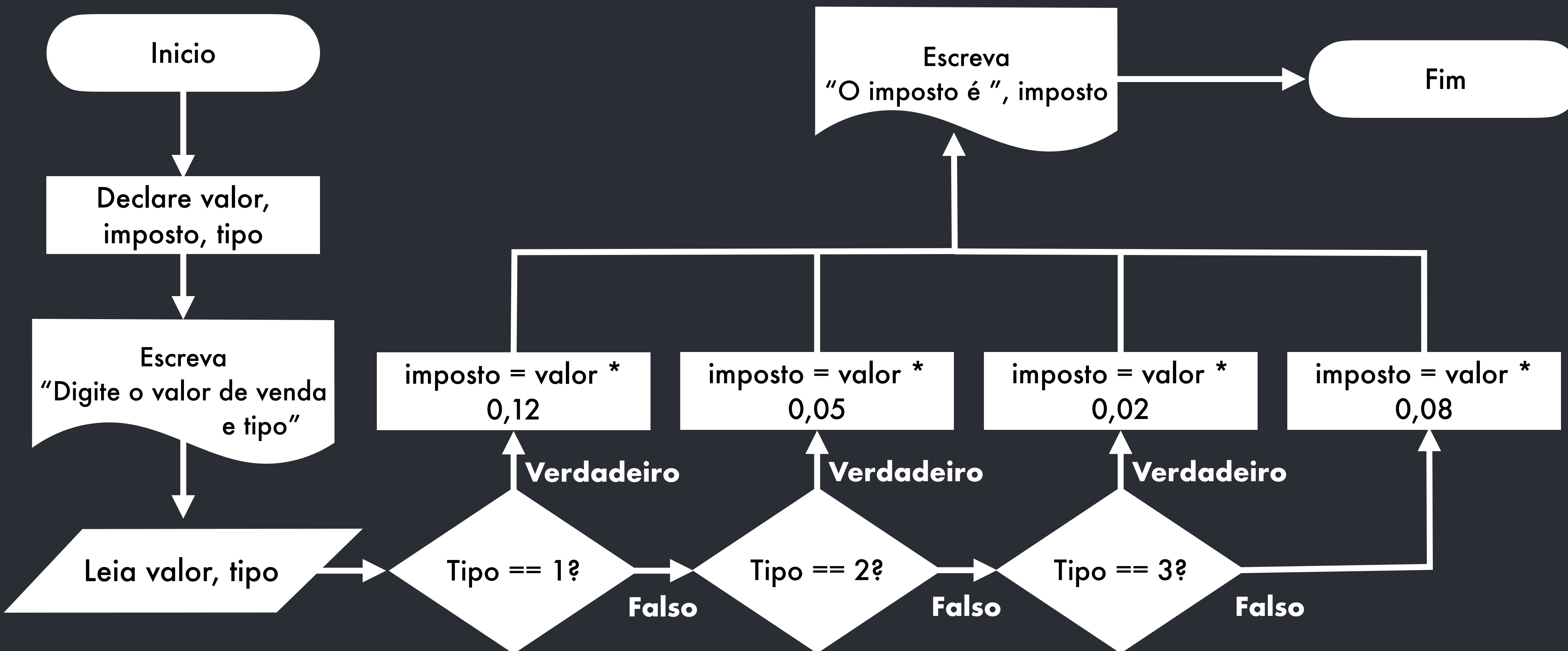
Usado em casos onde uma variável pode assumir um conjunto de valores exatos e disjuntos entre si.

```
switch (<variável>) {  
    case <valor>:<comando>;  
        ...  
        <comando>;  
        break;  
    ...  
    case <valor>:<comando>;  
        ...  
        <comando>;  
        break;  
}
```

```
switch (<variável>) {  
    case <valor>:<comando>;  
        ...  
        <comando>;  
        break;  
    ...  
    case <valor>:<comando>;  
        ...  
        <comando>;  
        break;  
    default:<comando>;  
        ...  
        <comando>;  
}
```

Faça um programa que calcule o imposto devido sobre um produto (valor de venda) de acordo com sua classe:

- (1) Eletrônico: 12%
- (2) Alimentação: 5%
- (3) Vestuário: 2%
- (4) Outros: 8%



LET'S CODING ....