

INF 1514

Introdução à Análise de Dados

Material 2

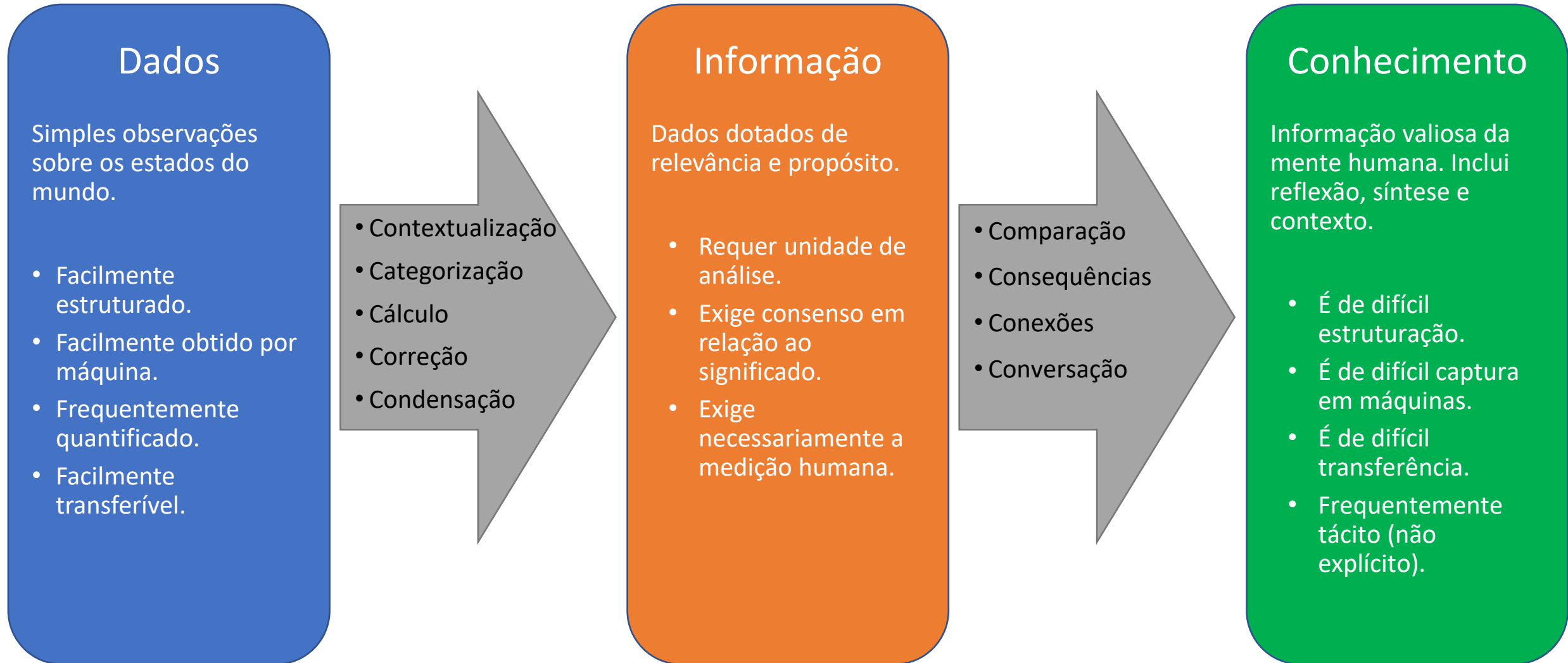


Este curso é idealizado para **ensinar programação** para **análises básicas** e **visualizações de dados**.

INF 1514



Dados, informação e conhecimento



Tipos de dados

Estruturados

Possuem formato definido e comprimento, fácil de armazenar e analisar com alto grau de organização.

Exemplos: bancos de dados relacionais, arquivos CSV, etc.

Semiestruturados

Representação estrutural heterogênea (não completamente desestruturados e nem fortemente tipados).

A estrutura geralmente implícita nos próprios dados.

Exemplos: XML, JSON, RDF, OWL, etc.

Não estruturados

Existem em seu estado natural, sem estrutura prévia definida.

Exemplos: arquivos Doc, e-mails, imagens, áudios, vídeos, etc.

Na disciplina utilizaremos fontes de dados como: Base de Dados, IBGE, Data Zoom, FRED e IEPS Data.

Tipos de dados

Estruturados

PR11835379
SC07762154
RS11088065

Exemplo de arquivo texto no qual as duas primeiras colunas identificam o estado e as oito demais colunas armazenam a população.

Semiestruturados

```
<sul>
  <estado>
    <sigla>PR</sigla>
    <populacao>11835379</populacao>
  </estado>
  <estado>
    <sigla>SC</sigla>
    <populacao>7762154</populacao>
  </estado>
  <estado>
    <sigla>RS</sigla>
    <populacao>11088065</populacao>
  </estado>
</sul>
```

Exemplo de arquivo XML com os mesmos dados do arquivo texto.

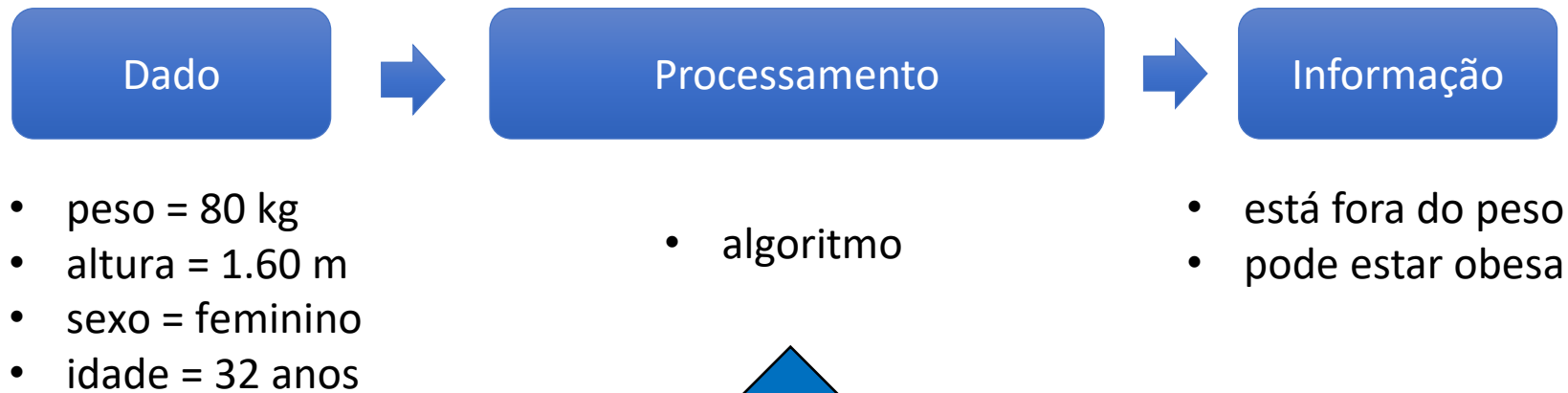
Não estruturados

A população dos estados PR, SC e RS é respectivamente 11835379, 7762154 e 11088065.

Exemplo de e-mail com os mesmos dados do arquivo texto.

Na disciplina utilizaremos fontes de dados como: Base de Dados, IBGE, Data Zoom, FRED e IEPS Data.

Processamento de dados consiste no **tratamento sistemático de dados**, através de **computadores** ou de outros **dispositivos eletrônicos**, com o objetivo de **ordenar, classificar** ou **efetuar quaisquer transformações** nos **dados**, segundo um **plano previamente programado**, visando a **obtenção de um determinado resultado**.

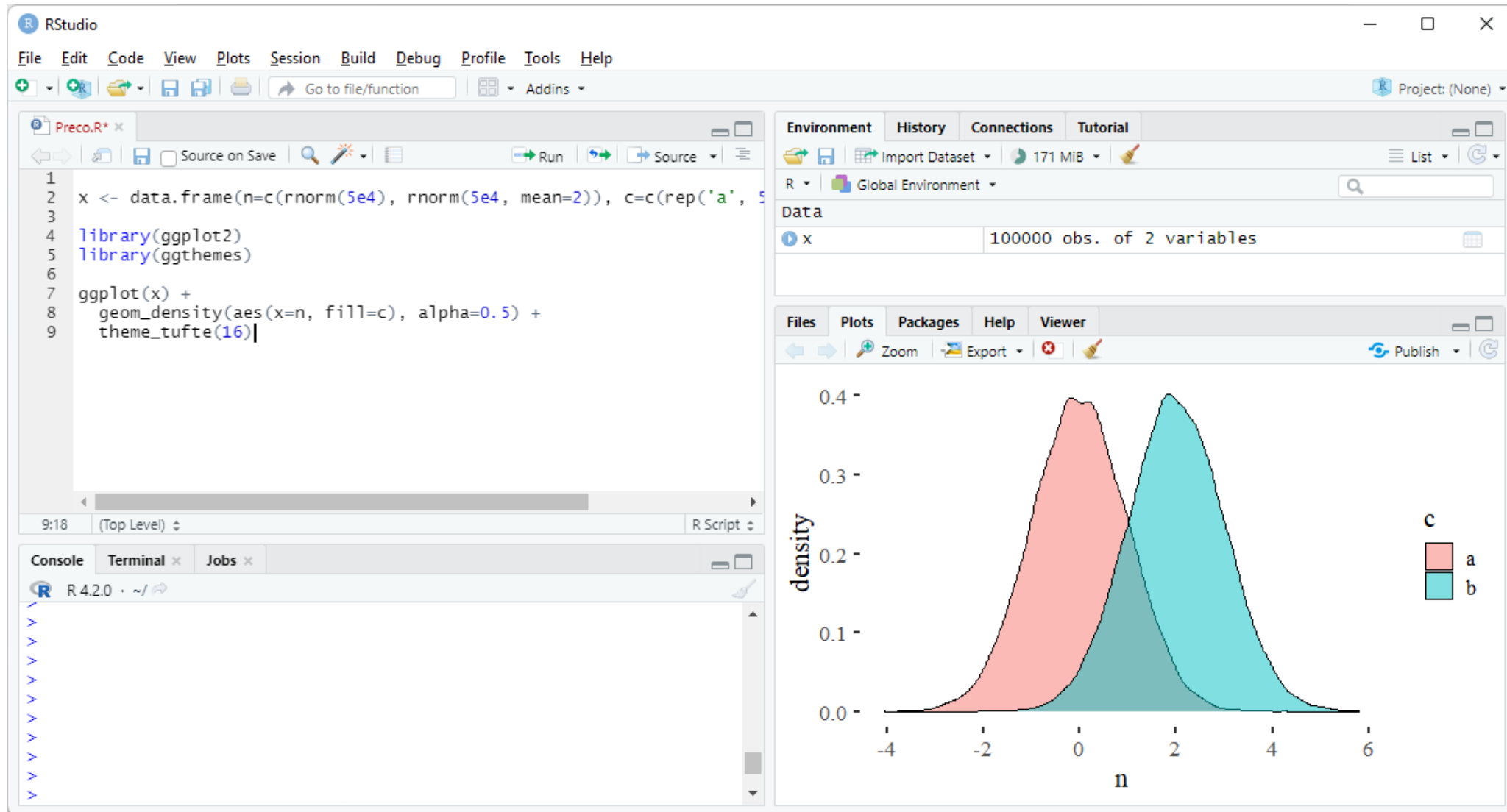


Segundo a Organização Mundial da Saúde se o índice de massa corporal para adultos ($imc = \frac{peso}{altura^2}$) for superior a 30 kg/m² então a pessoa está fora do peso.

R

- R é uma **linguagem** e um **ambiente de desenvolvimento** integrado.
- Foi criada originalmente por Ross Ihaka e por Robert Gentleman na universidade de Auckland, Nova Zelândia, e foi desenvolvida por um **esforço colaborativo** de pessoas em vários locais do mundo.
- A sua estrutura de **código aberto** e de **software público e gratuito** atraiu um grande número de desenvolvedores, sendo que há hoje inúmeros pacotes para o R.
- O R disponibiliza uma **ampla variedade de técnicas estatísticas e gráficas**, incluindo modelagem linear e não linear, testes estatísticos clássicos, análise de séries temporais, classificação, agrupamento e outras.
- Tem uma **história longa e confiável** e uma **forte comunidade** de suporte no setor de dados.

R Studio



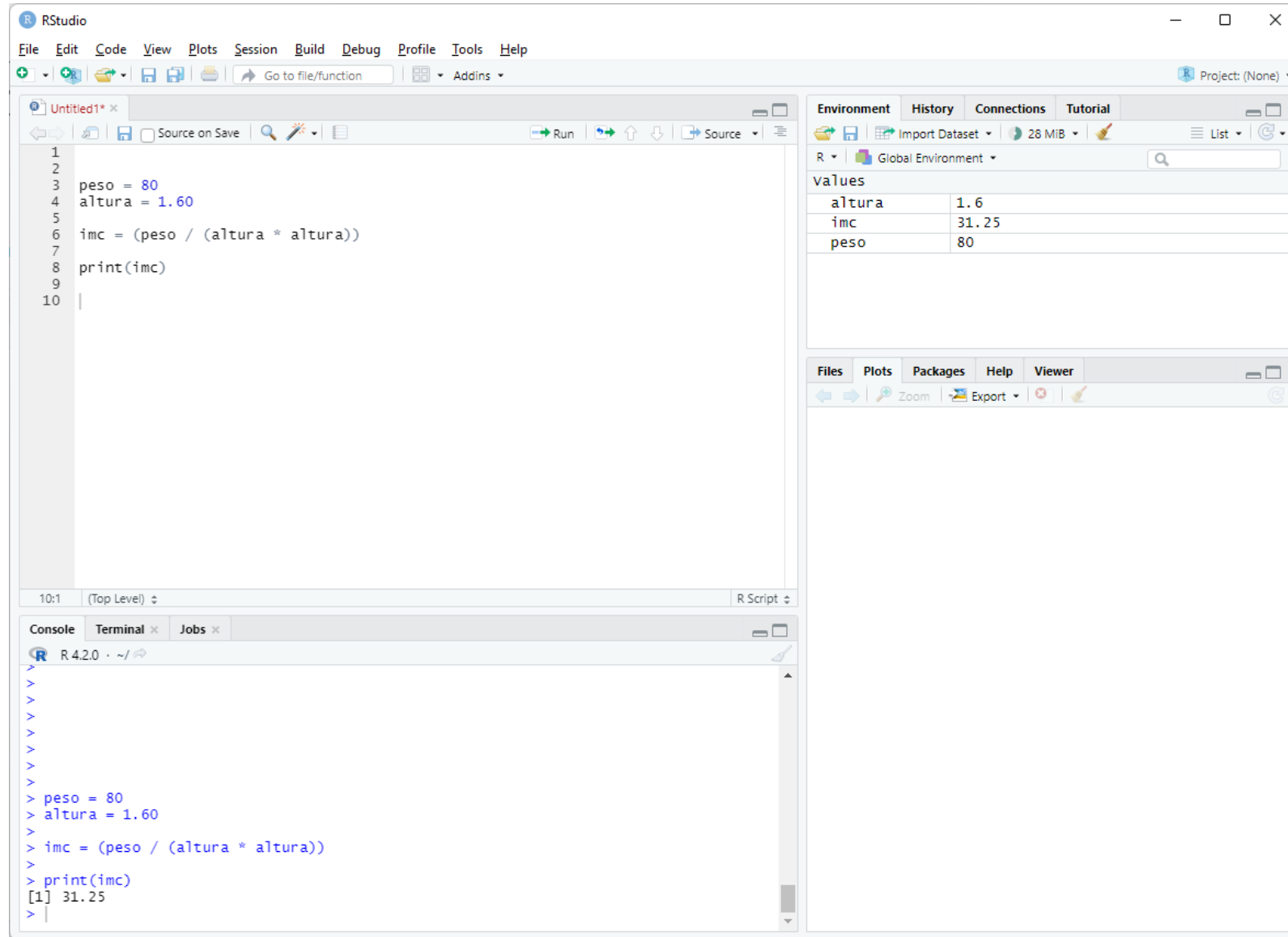
Instalação do ambiente R

- Realize o download do instalador a partir do site <http://www.r-project.org/> e o execute. Por questões de permissão, a instalação deverá ser feita utilizando um usuário administrador do computador no qual o R será instalado.
- Apesar de o R vir com uma interface gráfica, o RStudio torna a programação mais amigável.
- Entre no site <http://www.rstudio.com/> e faça o download do instalador da versão FREE e execute a instalação do produto. Por questões de permissão, a instalação deverá ser feita utilizando um usuário administrador do computador no qual o RStudio será instalado.
- Para computadores com Windows, após a instalação, use o ícone do RStudio que normalmente está disponível no *desktop* para carregar o ambiente.

Exemplo R

```
peso = 80  
altura = 1.60  
imc = (peso / (altura * altura))  
print(imc)
```

Exemplo R



The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains an R script with the following code:

```
1  
2  
3 peso = 80  
4 altura = 1.60  
5  
6 imc = (peso / (altura * altura))  
7  
8 print(imc)  
9  
10 |
```
- Environment Pane:** Shows the current environment with the following values:

Variable	Value
altura	1.6
imc	31.25
peso	80
- Console:** Shows the execution output:

```
>  
>  
>  
>  
>  
>  
>  
> peso = 80  
> altura = 1.60  
> imc = (peso / (altura * altura))  
>  
> print(imc)  
[1] 31.25  
>
```

Algoritmo

- Em termos mais simples, um algoritmo é um **passo a passo** bem **objetivo** para se fazer alguma coisa, usando de recursos limitados para tal.
- Um algoritmo **não representa**, necessariamente, um **programa de computador**, e sim os **passos** necessários para **realizar uma tarefa**. Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano.
- **Diferentes algoritmos** podem realizar a **mesma tarefa**, usando um **conjunto diferenciado de instruções**, em mais ou menos tempo, espaço ou esforço do que outros. Tal diferença pode ser reflexo da complexidade computacional aplicada, que depende de estruturas de dados adequadas ao algoritmo.
- Por exemplo, um algoritmo para se vestir pode especificar que você vista primeiro as meias e os sapatos antes de vestir a calça, enquanto outro algoritmo especifica que você deve primeiro vestir a calça e depois as meias e os sapatos. Fica claro que o primeiro algoritmo é mais difícil de executar que o segundo apesar de ambos levarem ao mesmo resultado.

Algoritmo

- Os algoritmos podem **repetir passos** (fazer iterações) ou **necessitar de decisões** (tais como comparações ou lógica) até que a tarefa seja completada.
- Um algoritmo corretamente executado não irá resolver um problema se estiver **implementado incorretamente** ou se **não for apropriado** ao problema.
- Um algoritmo é uma sequência finita de instruções **bem definidas** e **não ambíguas**, cada uma das quais pode ser executada mecanicamente num período de tempo finito e com uma quantidade de esforço finita.
- Especificação **precisa (não ambígua)** de um comportamento que visa a resolver um **problema bem definido**.

Algoritmo

- O conceito de algoritmo é frequentemente ilustrado pelo exemplo de uma receita culinária. Vamos fazer um bolo.

Ingredientes: 2 xícaras de açúcar, 3 xícaras de farinha de trigo, 4 colheres de margarina **bem cheias**, 3 ovos, 1 1/2 xícara de leite **aproximadamente**, 1 colher (sopa) de fermento em pó **bem cheia**.

Preparo: Bata as claras em neve, reserve , **bata bem as gemas** com a margarina e o açúcar, acrescente o leite e farinha **aos poucos** sem parar de bater, por último agregue as claras em neve e o fermento, coloque em **forma grande** de furo central untada e enfarinhada, asse em **forno médio**, pré-aquecido, por **aproximadamente** 40 minutos, quando espetar um palito e **sair limpo** estará assado.

Propriedades de um algoritmo

- Composto por **ações simples** e **bem definidas** (não pode haver ambiguidade, ou seja, cada instrução representa uma ação que deve ser entendida e realizada).
- **Sequência ordenada** de ações.
- **Conjunto finito** de passos.

```
peso = 80
altura = 1.60
imc = (peso / (altura * altura))
print(imc)
```

Algoritmos computacionais

- **Diferem dos algoritmos gerais** por serem executados pelo computador.
- Podem ser **classificados** de diferentes formas:
 - Implementação: recursivo, serial, paralelo, etc.
 - Paradigma: divisão e conquista, programação dinâmica, etc.
 - Campo de estudo: busca, ordenação, teoria de grafos, etc.
 - Complexidade computacional.
- Conceitos básicos utilizados na construção e interpretação de algoritmos:
 - **Estrutura de dados:** utilizada para manipulação e armazenamento das informações utilizadas no algoritmo.
 - **Estrutura de controle:** utilizada para manipulação das ações.

Diretrizes para elaboração de algoritmos

- **Identificação do problema:** determinar o que se quer resolver ou qual objetivo a ser atingido.
- **Identificação das entradas do sistema:** quais informações estarão disponíveis (serão fornecidas).
- **Identificação das saídas do sistema:** quais informações deverão ser geradas/calculadas como resultado.
- **Definição dos passos a serem realizados:** determinar a sequência de ações que levem à solução do problema (transformem as entradas nas saídas):
 - Identificar as regras e limitações do problema;
 - Identificar as limitações do computador;
 - Determinar as ações possíveis de serem realizadas pelo computador.
- **Concepção do algoritmo:** registrar a sequência de comandos, utilizando uma das formas de representação de algoritmos.
- **Teste da solução:** execução manual de cada passo do algoritmo, seguindo o fluxo estabelecido, para detectar possíveis erros.

Mais conceitos

- **Programação** – ato de instruir o computador para que ele resolva um determinado problema.
- **Linguagem de programação** – linguagem apropriada para descrever algoritmos computacionais que permitirão inserir ações e dados no computador.
- **Programa** – algoritmo implementado em uma linguagem de programação.
- **Linguagem de máquina** – linguagem de programação própria de cada modelo de computador.
- **Compilador** (para uma linguagem de programação) – programa (ou um conjunto de programas) que realiza a tradução automática de um programa, escrito em uma certa linguagem, para um programa equivalente, escrito na linguagem de máquina.
- **Interpretador** (para uma linguagem de programação) – programa (ou um conjunto de programas) que a partir da primeira instrução do programa fonte, confere se ela está escrita corretamente, converte-a em linguagem de máquina e então ordena a execução da instrução, a seguir, repete o processo para a segunda instrução, e assim sucessivamente, até a última instrução do programa.

Tipos primitivos de dados

- Os dados podem ser **numéricos**, **literais** e **lógicos**.
- Os **dados numéricos** dividem-se em **números inteiros** (-32; 0; 1; 198; 100; etc) e **números reais** (1,3; -5,03; 89,045; etc).
- Os **dados literais** (“a”; “AB”; “@”; etc...) são também chamados de **alfanuméricos**, **cadeia de caracteres** ou **strings**.
- Os **dados lógicos** incluem apenas os valores lógicos **falso** e **verdadeiro**.
- Um algoritmo manipula dados, que podem ser dados **variáveis** ou **constantes**.
- **Dados variáveis** são representados por variáveis, enquanto dados constantes são representados por constantes.
- Uma **variável** pode ser imaginada como uma “caixa” para armazenar valores de dados.

Variáveis

- O **nome** de uma **variável** deve ser **único**, isto é, identificar, de forma única, a variável no algoritmo.
- O **tipo** de uma **variável** define os valores que podem ser armazenados na variável.
- O **conteúdo** de uma **variável** é o valor que ela armazena.
- O ato de se **criar** uma **variável** é conhecido como **declaração de variável**.

```
peso = 80
altura = 1.60
imc = (peso / (altura * altura))
print(imc)
```

Formas de representação de algoritmos

- A **descrição** de um algoritmo de forma **clara** e **fácil de ser seguida** ajuda no seu **desenvolvimento**, **depuração** (localização e correção de erros) e futura **codificação** em uma linguagem de programação.
 - **Descrição narrativa**: especificação verbal dos passos em linguagem natural.
 - **Fluxograma**: uso de ilustrações gráficas para representar as instruções.
 - **Pseudocódigo**: linguagem especial para representação de algoritmos, a qual utiliza expressões pré-definidas para representar ações e fluxos de controle.

Exemplo

- Suponha que em uma determinada loja o **preço final** que um cliente irá pagar por um **produto** seja definido com base no **preço do produto** e no seguinte **conjunto de regras**:
 - Se o pagamento for **à vista (parcela única)**, o cliente terá **10% de desconto**.
 - À prazo em **duas vezes iguais** sendo uma parcela para **30** e outra para **60** dias, sofrerá **acréscimo de 5%**.
 - À prazo em **três vezes iguais** sendo **uma parcela** para **30**, outra para **60** e uma terceira para **90** dias sofrerá **acréscimo de 10%**.
- Defina um algoritmo para calcular o preço final pago pelo cliente dado o número de parcelas e o preço do produto.

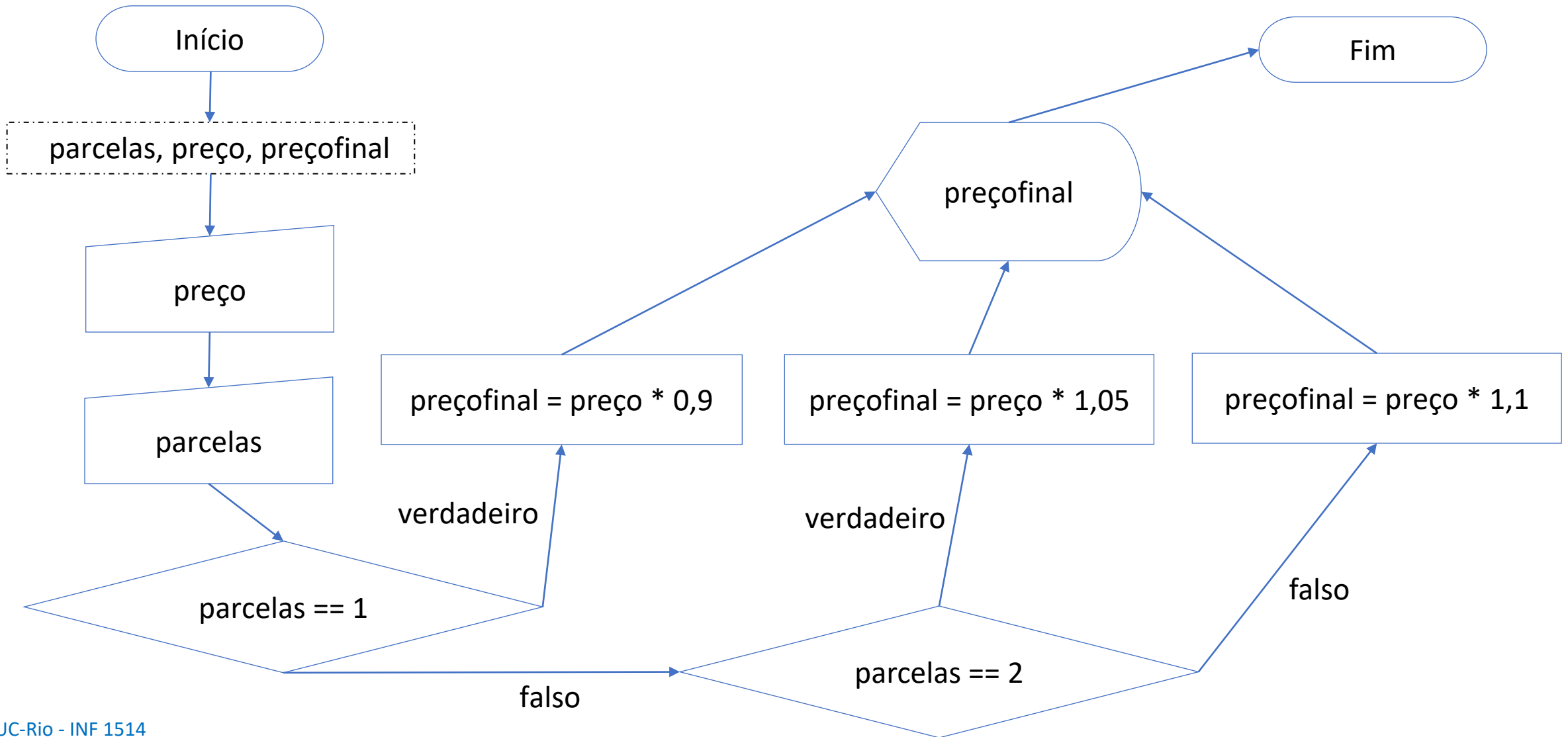
Usando descrição narrativa

1. Obter o **preço do produto**.
2. Obter o **número de parcelas**.
3. Se o número de parcelas for **1**, calcular o preço final a ser pago usando a fórmula ***preço final = valor do produto * 0,9***.
4. Caso contrário, se o número de parcelas for **2**, calcular o preço final a ser pago usando a fórmula ***preço final = preço do produto * 1,05***.
5. Caso contrário, calcular o preço final a ser pago usando a fórmula ***preço final = preço do produto * 1,1***.

Descrição narrativa

- **Especificação verbal** dos passos em **linguagem natural**.
- Desvantagens:
 - a **linguagem natural** é **imprecisa** (possibilita ambiguidades);
 - proporciona **maior trabalho na codificação**.
- Sugere-se sua utilização apenas para **comentar** algoritmos e/ou programas, **esclarecendo** ou **realçando** pontos específicos.








Usando fluxograma



Fluxograma

- Usa **ilustrações gráficas** para representar as **instruções**.
- Apresenta a lógica de um algoritmo, enfatizando **passos individuais** (objetos gráficos) e o **fluxo de execução** (setas).
- Desvantagens:
 - fluxogramas **detalhados** podem **obscurecer** a estrutura do programa;
 - permite **transferências arbitrárias** de controle.

Fluxograma

Símbolo	Nome	Descrição
	Terminador	Indica o início e o fim do fluxo do algoritmo.
	Seta de fluxo	Indica o sentido do fluxo de execução do algoritmo. É através dela que os símbolos do fluxograma são conectados.
	Declaração	Delimita a seção de declaração de variáveis.
	Entrada de dados	Corresponde à instrução de entrada de dados através do teclado.
	Atribuição	Símbolo utilizado para indicar cálculos e atribuição de valores.
	Saída de dados	Corresponde à instrução de saída de dados. Os dados serão exibidos na tela do computador.
	Desvio condicional	Divide o fluxo do programa em dois caminhos, dependendo do teste lógico que fica dentro do losango.

Usando pseudocódigo

```
variáveis
    parcelas, preço, preçofinal
início
    leia preço;
    leia parcelas;
    se (parcelas == 1) então
        preçofinal = preço * 0,9;
    senão
        se (parcelas == 2) então
            preçofinal = preço * 1,05;
        senão
            preçofinal = preço * 1,1;
        fim-se;
    fim-se;
    escreva preçofinal;
fim.
```

Pseudocódigo

- **Linguagem** especial para desenvolvimento de algoritmos, que utiliza **expressões pré-definidas** para representar **ações** e **fluxos de controle**.
- Funciona como uma **linguagem simplificada** de programação, logo, **facilita a codificação futura**.
- É uma **descrição textual, estruturada** e regida por **regras**, que descrevem os **passos executados** no algoritmo.
- Possui características **similares às linguagens de programação**:
 - utiliza **palavras-chaves** (ex: escreva, se-então, etc.) e **identação** (alinhamento dos blocos de comandos);
 - possui um comando por linha;
 - utiliza “;” como finalizador de comando.

Usando pseudocódigo (outra forma de escrever)

```
variáveis
    parcelas, preço, preçofinal
início
    leia preço;
    leia parcelas;
    se (parcelas == 1) então
        preçofinal = preço * 0,9;
    senão
        se (parcelas == 2) então
            preçofinal = preço * 1,05;
        senão
            preçofinal = preço * 1,1;
        fim-se;
    fim-se;
    escreva preçofinal;
fim.
```

```
variáveis
    parcelas, preço, preçofinal
início
    leia preço;
    leia parcelas;
    se (parcelas == 1) então
        preçofinal = preço * 0,9;
    senão se (parcelas == 2) então
        preçofinal = preço * 1,05;
    senão
        preçofinal = preço * 1,1;
    fim-se;
    escreva preçofinal;
fim.
```

Exercício 1

- Construa o pseudocódigo de um algoritmo para obter o resultado da divisão de dois números quaisquer fornecidos pelo usuário.

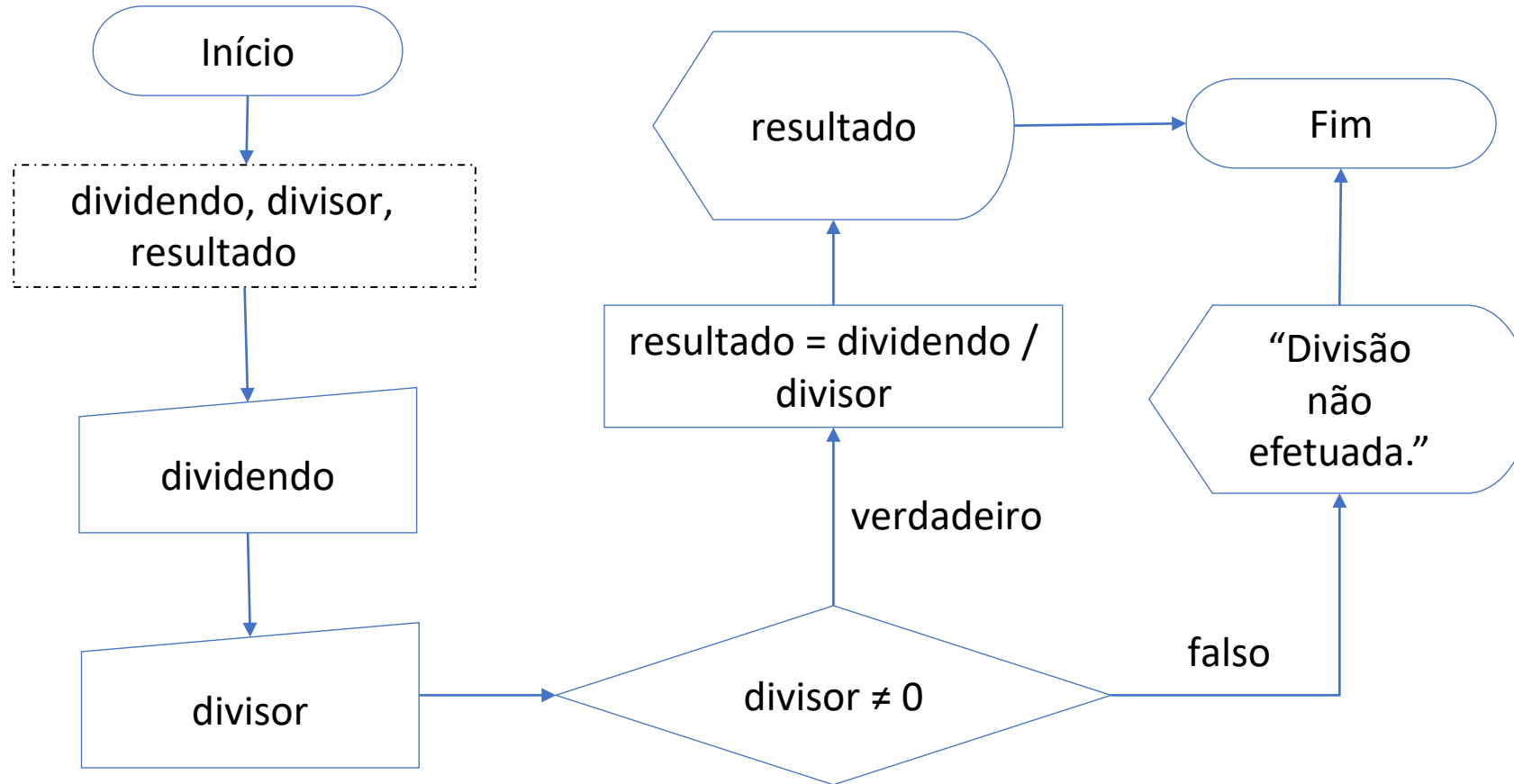
Exercício 1 - Resolução

```
variáveis
    dividendo, divisor, resultado
início
    leia dividendo;
    leia divisor;
    se (divisor ≠ 0) então
        resultado = dividendo / divisor;
        escreva resultado;
    senão
        escreva "Divisão não efetuada.";
    fim-se;
fim.
```


Exercício 2

- Construa o fluxograma de um algoritmo para obter o resultado da divisão de dois números quaisquer fornecidos pelo usuário.

Exercício 2 - Resolução



Estruturas de controle de fluxo

- Instrução condicional.

```
variáveis
    dividendo, divisor, resultado
início
    leia dividendo;
    leia divisor;
    se (divisor ≠ 0) então
        resultado = dividendo / divisor;
        escreva resultado;
    senão
        escreva "Divisão não efetuada.";
    fim-se;
fim.
```

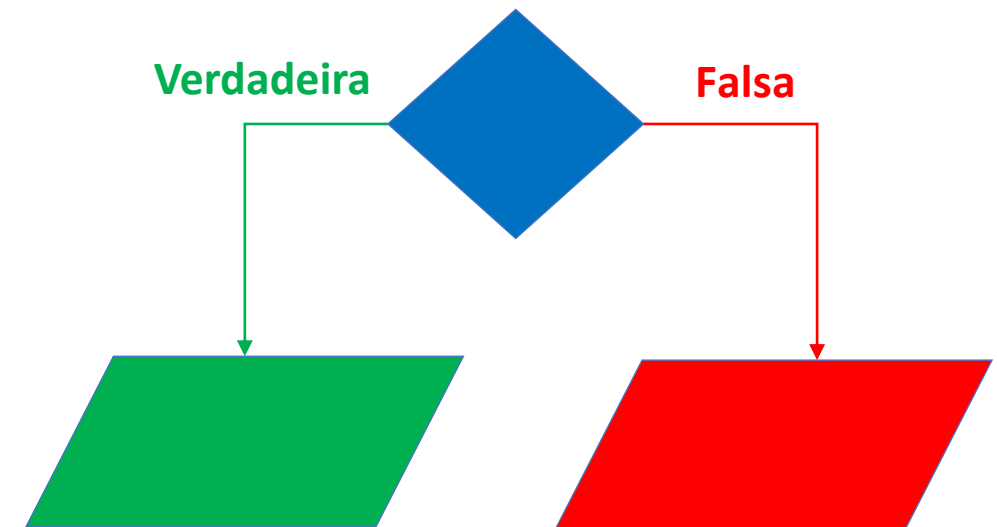
Estruturas de controle de fluxo

- Instrução condicional.

```
variáveis
    dividendo, divisor, resultado
início
    leia dividendo;
    leia divisor;
    se (divisor ≠ 0) então
        resultado ← dividendo / divisor;
        escreva resultado;
    senão
        escreva "Divisão não efetuada.";
    fim-se;
fim.
```

Nos pseudocódigos e fluxogramas deve-se usar \neq para "diferente de" e $=$ para "igual a".

```
....
se (<expressão-lógica>) então
    <sequência-de-comandos-verdadeira>
senão
    <sequência-de-comandos-falsa>
fim-se;
...
```



Expressões booleanas

- Podem ser utilizadas para **criar condições** que ao serem avaliadas retornam um valor **verdadeiro** ou **falso**.

<pre>.... se (<i>parcela</i> < 3) então <...> senão <...> fim-se; ...</pre>	<pre>.... se (<i>parcela</i> <= 5) então <...> senão <...> fim-se; ...</pre>	<pre>.... se (<i>parcela</i> >= 8) então <...> senão <...> fim-se; ...</pre>
--	---	---

- Podem ser combinadas através de conectores (**e**, **ou**) gerando **condições mais complexas**:

<pre>.... se (<i>parcela</i> == 3) e (<i>preço</i> < 65) então <...> senão <...> fim-se; ...</pre>	<pre>.... se (<i>parcela</i> > 4) ou (<i>preço</i> >= 230) então <...> senão <...> fim-se; ...</pre>
---	--