

INSTITUTO FEDERAL

Paraná

Campus Assis Chateaubriand

Algoritmos e Lógica de Programação

Introdução aos Computadores e à Programação

Introdução aos Computadores e à Programação

Conteúdo

- O que é e o que fazem, em linhas gerais, os computadores;
- O que são programas de computador;
- O que são linguagens de programação;
- Execução de programas em Java e em Pascal;
- O que são algoritmos e como construí-los;
- Utilização de pseudocódigo e de fluxogramas para construir algoritmos.

Introdução aos Computadores e à Programação

Objetivos de Aprendizagem

- Compreender como os computadores resolvem problemas;
- Definir o que são os programas de computador;
- Definir o que são algoritmos e como resolver problemas de forma algorítmica;
- Assimilar o conceito de ciclo EPS (entrada, processamento e saída), utilizando-o na construção de algoritmos;
- Utilizar pseudocódigo e fluxogramas para construir algoritmos sequenciais;
- Observar como as linguagens de programação são empregadas na construção de programas, utilizando exemplos em Pascal e em Java.

Os múltiplos propósitos de um computador

Um computador. Vários usos.



- **Educação:** redigir trabalhos acadêmicos, pesquisar assuntos na internet, EAD (Ensino à Distância), controlar registros escolares...
- **Trabalho:** analisar dados, fazer apresentações, processar transações comerciais, calcular planilhas eletrônicas, controlar estoques, redigir memorandos...
- **Em casa:** pagar contas, exibir filmes, realizar compras, jogar videogames, tocar músicas, comunicar com a família...

Tocadores de mp3, jogos, processadores de texto, compactadores de arquivos, editores de imagens... Diferente de um dispositivo como um rádio, um despertador ou um micro-ondas, os nossos computadores possuem várias funções diferentes. *Como é possível que um único equipamento, o computador, seja tão versátil?*



Diferente de outros dispositivos que utilizamos em nosso cotidiano, os computadores não são projetados para um uso específico, **podendo ser programados para executar tarefas diferentes**. São, portanto, **máquinas programáveis**.

Programação: as mil e uma utilidades dos computadores



Diversos dispositivos eletrônicos que utilizamos em nosso cotidiano, tais como rádio relógios, micro-ondas e calculadoras, que são construídos com um, e somente um, propósito: mostrar as horas, fazer contas aritméticas, sintonizar estações de rádio, aquecer alimentos utilizando radiação eletromagnética, etc. Em contraste, os computadores possuem infinitas aplicações, dependendo do programa que estão executando num determinado momento.

Sistemas Embarcados (*Embedded Systems*)

Certos dispositivos, apesar de serem computadores estritamente falando, possuem somente um único propósito. São os chamados **computadores embarcados**, como é o caso dos relógios de pulso digitais, navegadores GPS ou roteadores wireless, por exemplo. Teoricamente, são tão computadores quanto um tablet ou um notebook e, por isso, poderiam ser usados em várias aplicações diferentes, mas nem sempre isso é interessante (por que alguém usaria um GPS automotivo para tocar mp3?). Em síntese, enquanto os nossos computadores domésticos (tablets, notebooks, desktops) podem rodar **qualquer programa**, os computadores embarcados são concebidos para rodar **um único programa**.



Conceitos Fundamentais (1)

- Às vezes, os programas também são chamados de **software**. Em alguns contextos, os termos *programa* e *software* são ligeiramente distintos mas, em lógica de programação, geralmente são considerados sinônimos;
- As peças utilizadas na fabricação e montagem do computador, como teclado, mouse, monitor, cabos, impressora, câmeras, caixas de som, etc, constituem o que se chama de **hardware**;
- Em contraste com o **software**, o **hardware** constitui a parte tangível do computador. Nesse sentido, algumas pessoas dizem que *o hardware é a parte física do computador e o software, a parte lógica*;
- Um computador é constituído de componentes que, em conjunto, armazenam, processam e visualizam dados. Alguns desses componentes são **hardware** e outros são **software**.

Conceitos Fundamentais (2)



Hardware

(parte física do computador, suas peças)



Software

(parte lógica do computador; programas)

Programando Computadores (1)

A versatilidade dos computadores se deve à sua capacidade de serem programados

- Tudo que um computador é capaz de fazer depende da execução de um ou vários programas. *Mas o que é um programa?*
 - ✓ **Definição 1:** um programa é um conjunto de *instruções* (ou *comandos*) que indica ao computador o que ele deve fazer para concluir uma tarefa;
 - ✓ **Definição 2:** um programa é uma sequencia de passos elementares que o computador precisa executar ordenadamente para que um determinado problema seja resolvido.
- O profissional de informática que escreve (ou *codifica*) programas é chamado de *programador de computadores* ou, simplesmente, de *programador*.
- Os passos de um programa (instruções ou comandos) são escritos numa *linguagem* específica – a **linguagem de programação**. Existem centenas de linguagens de programação, como o Java, C, Pascal, assembly e muitas outras;
- Exemplos de programas: um navegador internet (Edge, Internet Explorer, Firefox, Opera...), um processador de texto (MS Office), um tocador de mp3 (Dopamine, VLC...).

Programando Computadores (2)

Dizendo "Olá, Mundo!" ("Hello, world!") em várias linguagens de programação

1. Javascript:

```
document.write('Hello, world!');
```

2. Bash:

```
echo "Hello, world!"
```

3. PHP:

```
<?php echo "Hello, world!"; ?>
```

4. CoffeeScript:

```
console.log 'Hello, world!'
```

5. R:

```
cat('Hello, world!')
```

6. C:

```
int main(void) {  
    puts("Hello, world!");  
}
```

7. C++:

```
int main(){  
    std::cout << "Hello, world!";  
    return 0;  
}
```

8. C#:

```
using System;  
class Program{  
    public static void Main(string[] args) {  
        Console.WriteLine("Hello, world!");  
    }  
}
```

9. Delphi:

```
program HelloWorld;  
begin  
    Writeln('Hello, world!');  
end.
```

10. Java:

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```

11. SQL:

```
SELECT "Hello, world!" as hello
```

12. HTML:

```
<html>  
<header><title></title></header>  
<body>  
    Hello, world!  
</body> </html>
```

13. jQuery:

```
$("body").append("Hello, world!");
```

14. Julia:

```
println("Hello world!")
```

15. Logo:

```
print [Hello, world!]
```

16. Matlab:

```
disp('Hello, world!')
```

17. Python:

```
Print "Hello, world!"
```

18. R:

```
cat('Hello, world!')
```

Programando Computadores (3)

Programas, Algoritmos e Lógica de Programação

- Antes de criar um programa, o programador precisa compreender bem *qual é o problema que precisa ser resolvido*, por exemplo, obter as raízes de um polinômio, determinar a média de consumo de um carro, prever o clima...
- Quando esse problema estiver bem definido, o programador pensa em *uma abordagem sistemática para resolvê-lo*; ou seja, cria um **algoritmo**:
 - Um **algoritmo** define *quais são os passos necessários para resolver um problema*, o que nos revela que *existe pouca diferença conceitual entre algoritmo e programa*. Contudo, enquanto um algoritmo se resume a uma descrição totalmente abstrata (é a expressão de uma ideia), o programa pode ser efetivamente executado num computador. **O computador executa programas, não algoritmos.**
- Um “algoritmo” pode ser pensado como a descrição (ou modelo) de um programa:
 - O **programa** é uma solução para um problema, criado por um programador. A intenção é que esse problema seja resolvido quando esse programa executar num computador;
 - O **algoritmo**, por sua vez, é a **descrição** dessa solução criada pelo programador; ou seja, **o algoritmo é um modelo do programa**, da mesma forma que uma maquete é o modelo de uma casa;
 - O programador descreve como pretende resolver o problema (ou seja, cria um algoritmo) e, em seguida, codifica um programa conforme essa descrição. **O programa implementa o algoritmo**;
 - O campo da Informática que estuda a criação de algoritmos é a **lógica de programação**.

Por que algoritmos?

Reflexão

Sabemos que os computadores executam programas (softwares) para funcionar. Por sua vez, um programa é uma sequência de instruções que precisam ser executadas ordenadamente por um computador para resolver um determinado problema. Para definir que instruções são essas, o programador precisa criar um algoritmo para entender melhor o problema e modelar de antemão o comportamento desse programa. Em resumo, antes que um programa seja codificado, é necessário entender o problema e escrever o algoritmo que o resolve.

Projetar um programa = criar um algoritmo

O processo de criar um algoritmo pode ser resumido em 2 passos:

1. Compreender o problema que o programa deve resolver;
2. Determinar quais são os passos para resolver esse problema.

Como criar um algoritmo? (1)

Depois de identificar o problema que o programa deve resolver, o programador cria uma solução. Essa solução (o algoritmo) é constituído de uma série de passos elementares.

Problema 1 – Ferver água para o chá da tarde

Algoritmo 1:

1. Colocar a quantidade de água desejada na chaleira;
2. Colocar a chaleira sobre o fogão, no centro de um dos queimadores;
3. Acionar o queimador, ajustando-o na posição "Máximo";
4. Acender o queimador imediatamente;
5. Aguardar até que bolhas grandes despontem rapidamente na água.

Problema 2 – Calcular o salário líquido de um empregado

Algoritmo 2:

1. Obter o número de horas trabalhadas pelo empregado num certo período;
2. Obter o valor da hora trabalhada, conforme a função do empregado;
3. Multiplicar o número de horas trabalhadas pelo valor da hora;
4. Descontar o INSS (8,0%) do valor acima para obter o salário líquido;
5. Mostrar o salário ao empregado.

À esquerda, temos um algoritmo criado para resolver o problema de ferver água para o chá. Analogamente, à direita, vemos um algoritmo criado resolver outro problema, calcular o salário de um empregado. Em ambos os casos, o algoritmo criado consiste num conjunto de passos bem definidos que devem ser executados em sequencia; ou seja, primeiro executa-se o passo 1, depois o passo 2 e assim por diante.

Como criar um algoritmo? (2)

- Os Algoritmos 1 e 2 descrevem os vários passos das soluções que resolvem os problemas 1 e 2;
- Somente o Algoritmo 2 é passível de implementação e pode ser transcrito num programa (o Algoritmo 1 serve apenas para ilustrar o conceito);
- Percebe-se que existe uma *sequencia* na qual os passos de um algoritmo devem ser executados:
 - Para ambos os algoritmos, os passos 1 e 2 podem ser executados em qualquer ordem mas, por outro lado, em nenhum deles faz sentido executar o passo 5 antes do 4.
- O Algoritmo 2 é uma especificação inequívoca de como calcular o salário líquido de um empregado, descontando o valor devido ao INSS:
 - Qualquer pessoa que siga os passos do Algoritmo 2, quer use um computador ou não, quer entenda de contabilidade ou não, irá conseguir calcular o salário líquido de um empregado.

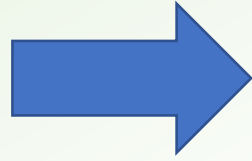
Como criar um algoritmo? (3)

- O Algoritmo 2 descreve de *forma metódica* cada passo que é necessário para calcular o salário líquido de um empregado:
 1. Primeiro, o algoritmo obtém os **dados de entrada** necessários para o cálculo do salário bruto. Para isso, a quantidade de horas trabalhadas e o valor de cada hora trabalhada precisam ser fornecidos pelo usuário mediante algumas **operações de entrada** (“Obter”);
 2. A seguir, os dados fornecidos pelo usuário são **processados** – o número de horas trabalhadas é multiplicado pelo valor de cada hora para obter o salário bruto, e esse último valor sofre um desconto relativo a cobrança do INSS para gerar o salário líquido, que é o **dado de saída**;
 3. Finalmente, os **dados de saída** (o salário líquido) são apresentados para o usuário através de **operações de saída** (“Mostrar”).
- Do exposto, claramente esse algoritmo (e, por extensão, qualquer outro) é dividido em três etapas distintas: (1) **Entrada**, (2) **Processamento** e (3) **Saída**.

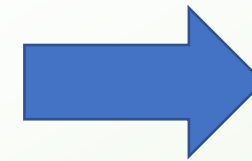
Como criar um algoritmo? (4)

Modelo EPS: Entrada, Processamento e Saída

Entrada



Processamento

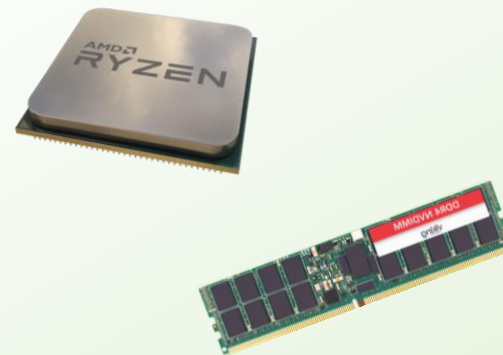


Saída

Operações de entrada, como o comando **“Leia”**
(leitura dos dispositivos de entrada, como teclado e mouse)



Transformação dos dados de entrada (operações lógicas e aritméticas, realizadas na CPU do computador)



Operações de saída, como o comando **“Escreva”**
(escrita nos dispositivos de saída, como monitor e impressora)



Como criar um algoritmo? (5)

O Algoritmo 2 está redigido num estilo casual – e isso não é bom!

- O Algoritmo 2 pode ser utilizado para especificar um programa, pois descreve todos os passos para resolver um problema. Mas:
 - Claramente, ele está redigido num estilo coloquial, que mais se assemelha com a linguagem utilizada na comunicação humana do que com as linguagens utilizadas para escrever programas que efetivamente executam num computador;
 - O ideal é que, em nome da clareza e para facilitar a posterior conversão num programa, o algoritmo seja escrito de maneira mais formal, enfatizando as etapas de Entrada, Processamento e Saída.
- Os programadores costumam ser mais formais na escrita de seus algoritmos, utilizando duas ferramentas específicas para escrevê-los: **pseudocódigo** e **fluxogramas**.

Pensando como um Computador: O **Modelo EPS** e o uso de Pseudocódigo

Um exemplo geométrico simples para entender o **Modelo EPS**:
calculando o comprimento e a área de uma circunferência

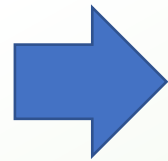
Da geometria, sabemos que tanto o comprimento l como a área A de uma circunferência são funções do seu raio r e de uma constante π , conforme a seguir:

$$l = 2\pi r, A = \pi r^2 \text{ e } \pi = 3.141592654$$

Então, de posse do raio de uma circunferência, como se cria um algoritmo para calcular seu comprimento e a sua área? E se quisermos usar **pseudocódigo** para isso?

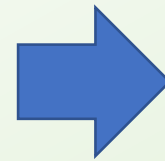
Entrada

Leia (vlRaio)



Processamento

```
ctPi ← 3.141592654;  
vlComp ← 2*ctPi*vlRaio;  
vlArea ← ctPi *vlRaio *vlRaio;
```



Saída

```
Escreva ("O Comprimento é");  
Escreva(vlComp);  
Escreva ("A Area é");  
Escreva(vlArea);
```

Pseudocódigo (1.1): Um pseudocódigo simples

Algoritmo 3

Algoritmo Calcula_Area_Comprimento;

```
1. Escreva ("Entre com o raio");  
2. Leia (vlRaio);  
3. ctPi ← 3.141592654;  
4. vlComp ← 2 * ctPi * vlRaio;  
5. vlArea ← ctPi * vlRaio * vlRaio;  
6. Escreva ("O valor do comprimento é ");  
7. Escreva (vlComp);  
8. Escreva ("O valor da area é ");  
9. Escreva (vlArea);
```

Fim algoritmo.

Repare que os passos desse algoritmo são numerados. Embora a numeração **não** seja obrigatória, ela ajuda a reforçar a ideia de que *esses passos devem ser executados em sequencia.* Afinal, que sentido há em tentar calcular o comprimento da circunferência sem conhecer o raio dela??

Isso é chamado de estrutura sequencial que, junto com as estruturas de decisão e de repetição, forma as estruturas de controle. Qualquer algoritmo pode ser escrito somente com o uso dessas três estruturas, independentemente da complexidade.

Pseudocódigo (1.2): Um pseudocódigo simples

Repare que o Algoritmo 3, como qualquer outro, obedece ao modelo **EPS**:

Entrada: o algoritmo lê o valor do raio de uma circunferência, que é fornecido pelo usuário. Esse valor é armazenado numa **variável** chamada “*vlRaio*”;

Processamento: usando o raio da circunferência e uma constante (“*ctPi*”), são calculados o comprimento e a área dela (variáveis “*vlComp*” e “*vlArea*”);

Saída: escreve-se os resultados obtidos na tela.

A **entrada de dados** é feita por meio de uma *leitura* num *dispositivo de entrada*. Nesse intento, o comando **Leia** pode capturar uma informação digitada pelo usuário no teclado e armazená-la numa **variável**. Por exemplo, para armazenar o raio da circunferência na **variável** *vlRaio*, o Algoritmo 3 utiliza o comando **Leia (vlRaio)**.

A **saída de dados** consiste em *escrever* num dos dispositivos de saída, como o monitor ou impressora, o que é geralmente feito por um comando como o **Escreva**. Por exemplo, para informar o valor da área da circunferência, armazenado na variável *vlArea*, o Algoritmo 3 utiliza o comando **Escreva (vlArea)**.

Uma **variável** é um local da memória RAM que o algoritmo utiliza para armazenar um determinado valor, que pode ser *numérico*, *alfanumérico* ou *lógico*. Cada variável possui um **identificador** (o “nome” da variável, como *vlRaio*) e um **valor** (por exemplo, 2.5). *Sempre que for utilizar o valor de uma variável, o algoritmo a invoca por meio do seu identificador.*

Para que uma variável assuma um valor, pode-se empregar o operador de **atribuição**, muitas vezes denotado por “←”. Por exemplo, no algoritmo, o comprimento da circunferência é armazenado na variável *vlComp* através do comando **vlComp ← 2 * ctPi * vlRaio**. Outra possibilidade é utilizar o comando **Leia** (quadro superior esquerdo).

Pseudocódigo (2): Um pseudocódigo para o Algoritmo 2

Algoritmo 4

Algoritmo Calcula_salario;

1. **Escreva** ("Entre com o número de horas trabalhadas");
2. **Leia** (numHorasTrab);
3. **Escreva** ("Entre com o valor da hora trabalhada");
4. **Leia** (valorHoraTrab);
5. $\text{valorDevido} \leftarrow \text{numHorasTrab} * \text{valorHoraTrab};$
6. $\text{vlINSS} \leftarrow 0,08 * \text{valorDevido};$
7. $\text{valorDevido} \leftarrow \text{valorDevido} - \text{vlINSS};$
8. **Escreva** ("O valor a ser pago é ", valorDevido);

Fim algoritmo.

Pseudocódigo (3): Outro pseudocódigo para o Algoritmo 2

Algoritmo 5

Algoritmo Calcula_salario;

Variaveis de entrada

numHorasTrab, valorHoraTrab: **número real**;

Variaveis de saída

valorDevido, valorINSS : **número real**;

Inicio do algoritmo

Escreva ("Entre com o número de horas trabalhadas");

Leia (numHorasTrab);

Escreva ("Entre com o valor da hora trabalhada");

Leia (valorHoraTrab);

valorDevido \leftarrow numHorasTrab * valorHoraTrab;

valorINSS \leftarrow 0,08 * valorDevido;

valorDevido \leftarrow valorDevido - valorINSS;

Escreva ("O valor a ser pago é ", valorDevido);

Fim algoritmo.

Pseudocódigo (4): E quanto às linguagens de programação?

Por que utilizar pseudocódigo, ao invés de programar diretamente?

- As linguagens de programação têm regras rígidas (definidas pela sua *sintaxe*) que devem ser seguidas estritamente na criação de um programa;
- Se essas regras forem violadas, o programa criado não pode ser executado até que o erro de sintaxe seja descoberto e corrigido;
- O **pseudocódigo** *é uma linguagem de programação imaginária, utilizada para criar algoritmos e que não possui uma sintaxe rigidamente definida:*
 - Com o pseudocódigo, o programador pode se concentrar mais na **lógica de programação** necessária para resolver o problema que se dispôs a resolver, ao invés de se preocupar com a sintaxe de uma linguagem de programação real.
- Embora o pseudocódigo seja mais formal do que a linguagem do cotidiano, não é tão formal quanto um programa em Java:
 - Compare o Algoritmo 2 com os Algoritmos 4 e 5, e esses últimos com o programa em Java para o cálculo do IMC (mostrado adiante nessa apresentação).

Pseudocódigo (5): E quanto às linguagens de programação?

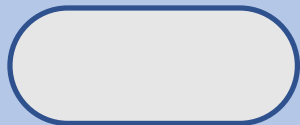
- Repare que os pseudocódigos apresentados realizam diversas operações: leitura (**Leia**), escrita (**Escreva**), multiplicação (“*”), atribuição (“←”) e assim por diante;
- *Cada uma dessas operações corresponde a um comando de uma linguagem de programação, qualquer que seja. É por isso se diz que o pseudocódigo contém a “lógica” do programa;*
- Diferente das linguagens de programação, *existe grande liberdade na forma de se escrever um pseudocódigo:*
 1. Os pseudocódigos dos Algoritmos 4 e 5 são ligeiramente diferentes, mas claramente fazem a mesma coisa: todos os passos do Algoritmo 4 são numerados, os do Algoritmo 5 não são; o Algoritmo 5 declara as variáveis, o Algoritmo 4 não declara; somente o Algoritmo 5 marca explicitamente o início do algoritmo;
 2. O algoritmo é apenas um modelo do programa que reproduz o raciocínio do programador, ou seja, expressa como ele pretende resolver um problema;
 3. Não faria diferença se o pseudocódigo utilizasse comandos diferentes dos mostrados como, por exemplo, “**Obtenha**” ao invés de “**Leia**”, “**Imprima**” ao invés de “**Escreva**” ou “=” ao invés de “←”. Como a intenção do pseudocódigo é comunicar o raciocínio do programador, detalhes sintáticos como esses são irrelevantes.

Pseudocódigo (6): Mais algumas considerações

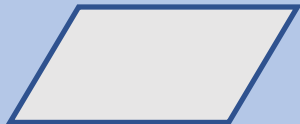
- Não importa muito quais são as convenções adotadas para escrever um pseudocódigo:
O objetivo do pseudocódigo é tão somente mostrar o raciocínio do programador claramente, com algum formalismo e sem ambiguidades!
- Naturalmente, como o pseudocódigo não executa no computador, em algum momento ele precisa ser convertido para uma linguagem de programação tal como o Java.
- **Nem sempre o pseudocódigo é utilizado na prática:**
Programadores experientes acham que é mais prático codificar seus programas diretamente na linguagem de programação que utilizam, como o Java.
É que, para eles, o Java se tornou mais fácil e intuitivo do que o pseudocódigo!
- Nem sempre a pessoa que escreve o algoritmo é quem vai implementar o programa e, talvez, essa pessoa não conheça a linguagem que será utilizada. Nesse caso, uma das alternativas desse indivíduo é escrever um pseudocódigo;
- De qualquer forma, o pseudocódigo continua sendo útil como ferramenta para o ensino de lógica de programação.

Fluxogramas (1): uma alternativa gráfica ao pseudocódigo

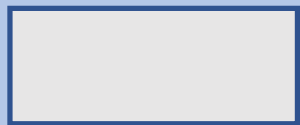
- Fluxogramas são ferramentas para a especificação de algoritmos que, ao contrário do pseudocódigo, são gráficas;
- Os fluxogramas são uma alternativa ao pseudocódigo, mas já foram muito mais populares no passado, até a década de 80;
- No pseudocódigo, cada passo do algoritmo é descrito por um comando tal como **Leia**, **Escreva**, etc. Em contrapartida, no fluxograma, cada passo é representado por um símbolo gráfico:



Símbolo terminal. Indica o início e o fim do fluxo de processamento.



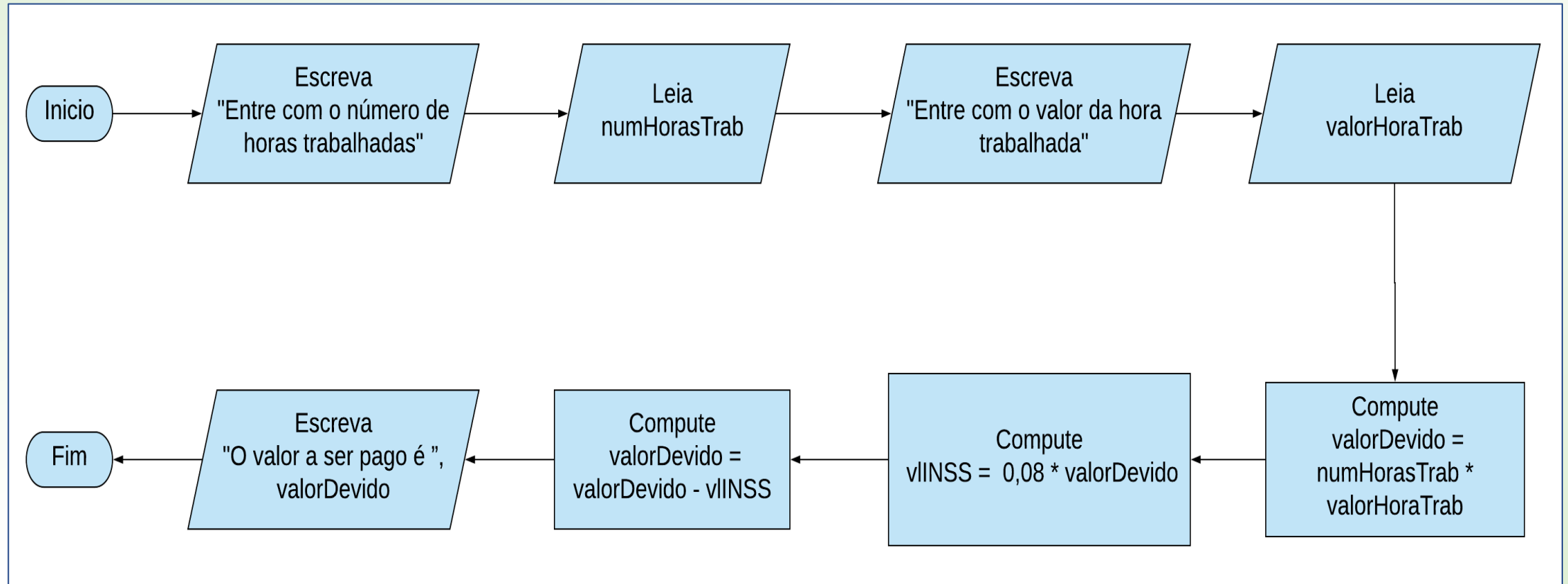
Símbolo de entrada ou saída de dados. Escreve ou lê uma informação



Símbolo de processamento. Realiza um cálculo ou qualquer outra operação em que o valor de um dado é alterado.

Fluxogramas (2):

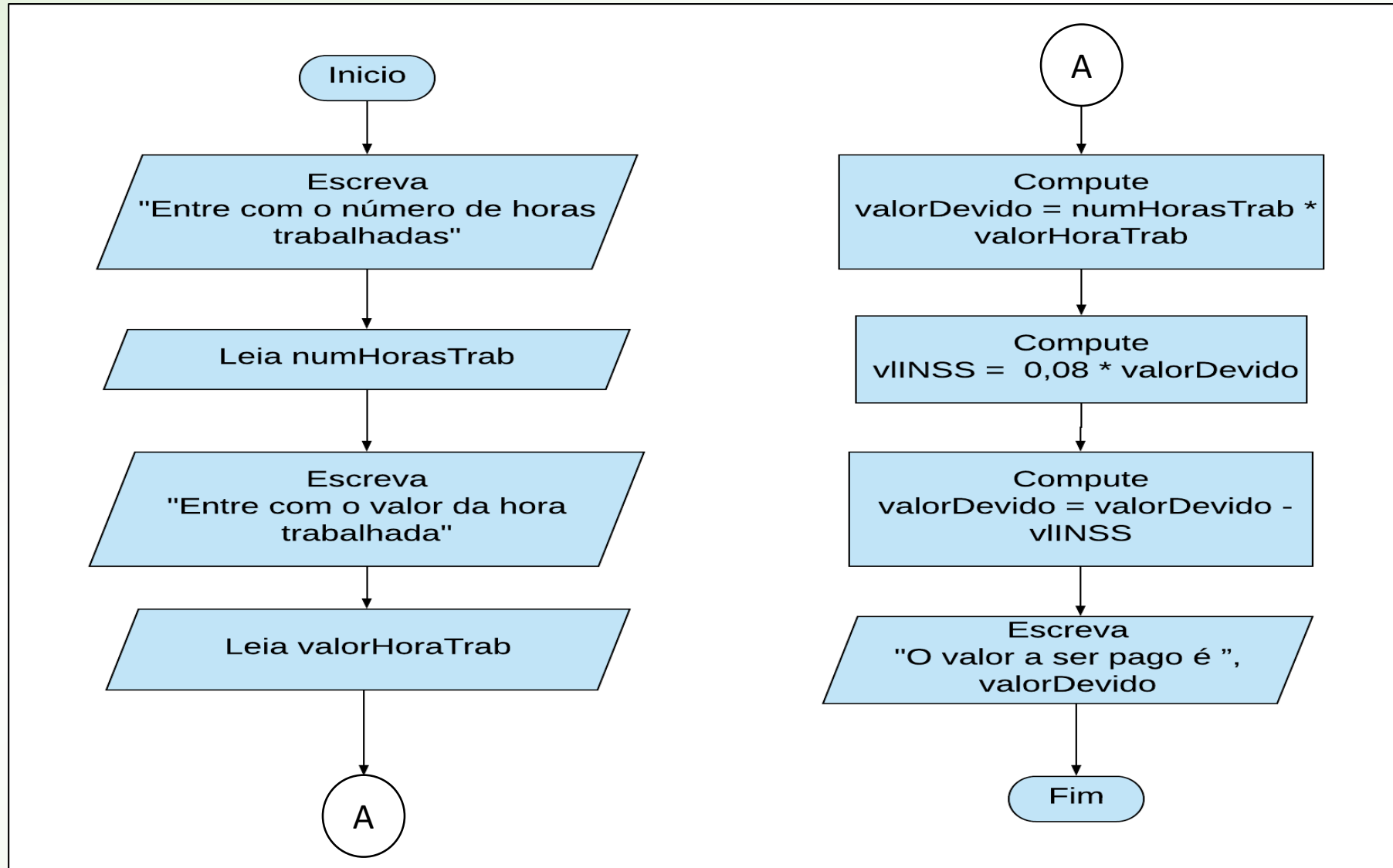
Um fluxograma para o Algoritmo 3



Esse diagrama foi criado on line, com o uso da ferramenta Lucidchart (<https://www.lucidchart.com>)

Fluxogramas (3):

Outro fluxograma para o Algoritmo 3



Um algoritmo, muitos estilos (1)

Uma descrição algorítmica informal para calcular o IMC

Algoritmo 6 – Cálculo do IMC (Índice de Massa Corporal)

“O IMC de uma pessoa é muito fácil de ser calculado. Tudo o que você precisa é dividir o peso dessa pessoa pela altura dela, elevada ao quadrado. Ou seja, de forma mais simples, você multiplica a altura por ela mesma e depois divide seu peso pelo resultado dessa última conta. Por exemplo, uma pessoa com 1,70 m e 70 kg fará o seguinte cálculo:

Altura multiplicada por ela mesma: $1,70 \times 1,70 = 2,89$

Peso dividido pelo quadrado da altura: $70 / 2,89 = 24,22$

Logo, essa pessoa tem IMC de 24.

Depois, basta localizar esse resultado na tabela ao lado para classificar a situação de obesidade da pessoa. No caso do exemplo dado, o IMC corresponde ao de uma pessoa com peso normal conforme os critérios da Organização Mundial de Saúde.”

IMC	Resultado
Abaixo de 18,5	Abaixo do peso
Entre 18,5 e 24,9	Peso normal
Entre 25 e 29,9	Sobrepeso
Entre 30 e 34,9	Obesidade grau 1
Entre 35 e 39,9	Obesidade grau 2
Mais do que 40	Obesidade grau 3

Fonte: adaptado de <https://www.minhavidade.com.br/alimentacao/tudo-sobre/32159-imc>

Um algoritmo, muitos estilos (2)

O Algoritmo 6, descrito mais formalmente, em pseudocódigo

Algoritmo 7 – Cálculo do IMC (Índice de Massa Corporal)

Algoritmo Calcula_IMC;

Variaveis de entrada

pesoPessoa, alturaPessoa: **número real**;

Variaveis de saída

valorIMC : **número real**;

grauObesidade: **caracter**;

Inicio do algoritmo

Escreva ("Entre com a altura da pessoa em metros");

Leia (alturaPessoa);

Escreva ("Entre com o peso da pessoa em kilogramas");

Leia (pesoPessoa);

$\text{valorIMC} \leftarrow \text{pesoPessoa} / (\text{alturaPessoa}^2);$

Escreva ("O IMC da pessoa é", valorIMC);

Um algoritmo, muitos estilos (3)

O Algoritmo 6, descrito mais formalmente, em pseudocódigo (cont)

Se (valorIMC<18,5) **então**

 grauObesidade ← "A pessoa está abaixo do peso";

Senão se (valorIMC<25) **então**

 grauObesidade ← "A pessoa está com peso normal";

Senão se (valorIMC<30) **então**

 grauObesidade ← "A pessoa está com sobrepeso";

Senão se (valorIMC<35) **então**

 grauObesidade ← "A pessoa está com obesidade grau 1";

Senão se (valorIMC<40) **então**

 grauObesidade ← "A pessoa está com obesidade grau 2";

Senão grauObesidade ← "A pessoa está com obesidade grau 3".

Escreva("O diagnostico da pessoa é:", grauObesidade)

Fim do algoritmo.

Um Exemplo de Programa (1.1): Transcrevendo o Algoritmo 7 para Java

```
import java.util.Scanner;
public class Calcula_IMC {
public static void main(String args[]) {
    float pesoPessoa, alturaPessoa, valorIMC;
    String grauObesidade;
    Scanner sc = new Scanner(System.in);
    System.out.println("Entre com a altura da pessoa em metros");
    alturaPessoa = sc.nextFloat();
    System.out.println("Entre com o peso da pessoa em kilogramas");
    pesoPessoa = sc.nextFloat();
    valorIMC = (pesoPessoa) / (alturaPessoa * alturaPessoa);
    System.out.println("O IMC da pessoa é " + valorIMC);
}
```

Nesse momento, não
abordaremos nenhuma
linguagem de
programação, apenas
algoritmos. Este
exemplo é meramente
ilustrativo!

Um Exemplo de Programa (1.2):

Transcrevendo o Algoritmo 7 para Java (cont.)

```
if (valorIMC < 18.5) {
    grauObesidade = "A pessoa está abaixo do peso";}
else if (valorIMC < 25) {
    grauObesidade = "A pessoa está com peso normal";}
else if (valorIMC < 30) {
    grauObesidade = "A pessoa está com sobrepeso";}
else if (valorIMC < 35) {
    grauObesidade = "A pessoa está com obesidade grau 1";}
else if (valorIMC < 40) {
    grauObesidade = "A pessoa está com obesidade grau 2";}
else {
    grauObesidade = "A pessoa está com obesidade grau 3";}
System.out.println("O diagnóstico da pessoa é: " + grauObesidade);
} //Fim do método main()
} //Fim da classe Calcula_IMC
```

Um Exemplo de Programa (2.1): Transcrevendo o Algoritmo 7 para Pascal

```
Program Calcula_IMC;  
Var  
    pesoPessoa, alturaPessoa, valorIMC: Real;  
    grauObesidade: String;  
Begin  
    Writeln('Entre com a altura da pessoa em metros');  
    Readln(alturaPessoa);  
    Writeln('Entre com o peso da pessoa em kilogramas');  
    Readln(pesoPessoa);  
    valorIMC := (pesoPessoa) / (alturaPessoa * alturaPessoa);  
    Writeln('O IMC da pessoa é ', valorIMC);
```

Nesse momento, não
abordaremos nenhuma
linguagem de
programação, apenas
algoritmos. Este
exemplo é meramente
ilustrativo!

Um Exemplo de Programa (2.2):

Transcrevendo o Algoritmo 7 para Pascal (cont.)

```
If (valorIMC < 18.5) Then  
    grauObesidade := 'A pessoa está abaixo do peso'  
Else If (valorIMC < 25) Then  
    grauObesidade := 'A pessoa está com peso normal'  
Else If (valorIMC < 30) Then  
    grauObesidade := 'A pessoa está com sobrepeso'  
Else If (valorIMC < 35) Then  
    grauObesidade := 'A pessoa está com obesidade grau 1'  
Else If (valorIMC < 40) Then  
    grauObesidade := 'A pessoa está com obesidade grau 2'  
Else  
    grauObesidade := 'A pessoa está com obesidade grau 3';  
Writeln('O diagnóstico da pessoa é: ' + grauObesidade);  
End.
```


Exemplo Ilustrativo (1)

Rodando um programa Java no navegador internet

The screenshot shows the Browxy web IDE interface. At the top, there's a header with 'Browxy BETA' and navigation links: Home, Feedback, Submissions, Published Code. Below this is a toolbar with buttons: File - Login to enable, Refresh, Open, New, Save, Download, Rename, Delete, GetUrl, Publish, Start, Stop, and a 'Finished OK' button. A text input field labeled 'Args' contains '-encoding UTF-8'. Two red arrows point to this field and the 'Start' button, with red circles containing the numbers '1' and '2' respectively. The main area displays a Java code editor with the following code:

```
1 import java.util.Scanner;
2 public class Calcula_IMC {
3     public static void main(String args[]) {
4         float pesoPessoa, alturaPessoa, valorIMC;
5         String grauObesidade;
6         Scanner sc = new Scanner(System.in);
7         System.out.println("Entre com a altura da pessoa em metros");
8         alturaPessoa = sc.nextFloat();
9         System.out.println("Entre com o peso da pessoa em kilogramas");
10        pesoPessoa = sc.nextFloat();
11        valorIMC = (pesoPessoa) / (alturaPessoa*alturaPessoa);
12        System.out.println("O IMC da pessoa é "+valorIMC);
13        if (valorIMC < 18.5) {grauObesidade = "A pessoa está abaixo do peso";}
14        else if (valorIMC < 25) {grauObesidade = "A pessoa está com peso normal";}
15        else if (valorIMC < 30) {grauObesidade = "A pessoa está com sobrepeso";}
16        else if (valorIMC < 35) {grauObesidade = "A pessoa está com obesidade grau 1";}
17        else if (valorIMC < 40) {grauObesidade = "A pessoa está com obesidade grau 2";}
18        else {grauObesidade = "A pessoa está com obesidade grau 3";}
19        System.out.println("O diagnóstico da pessoa é: "+grauObesidade);
20    }
21 }
```

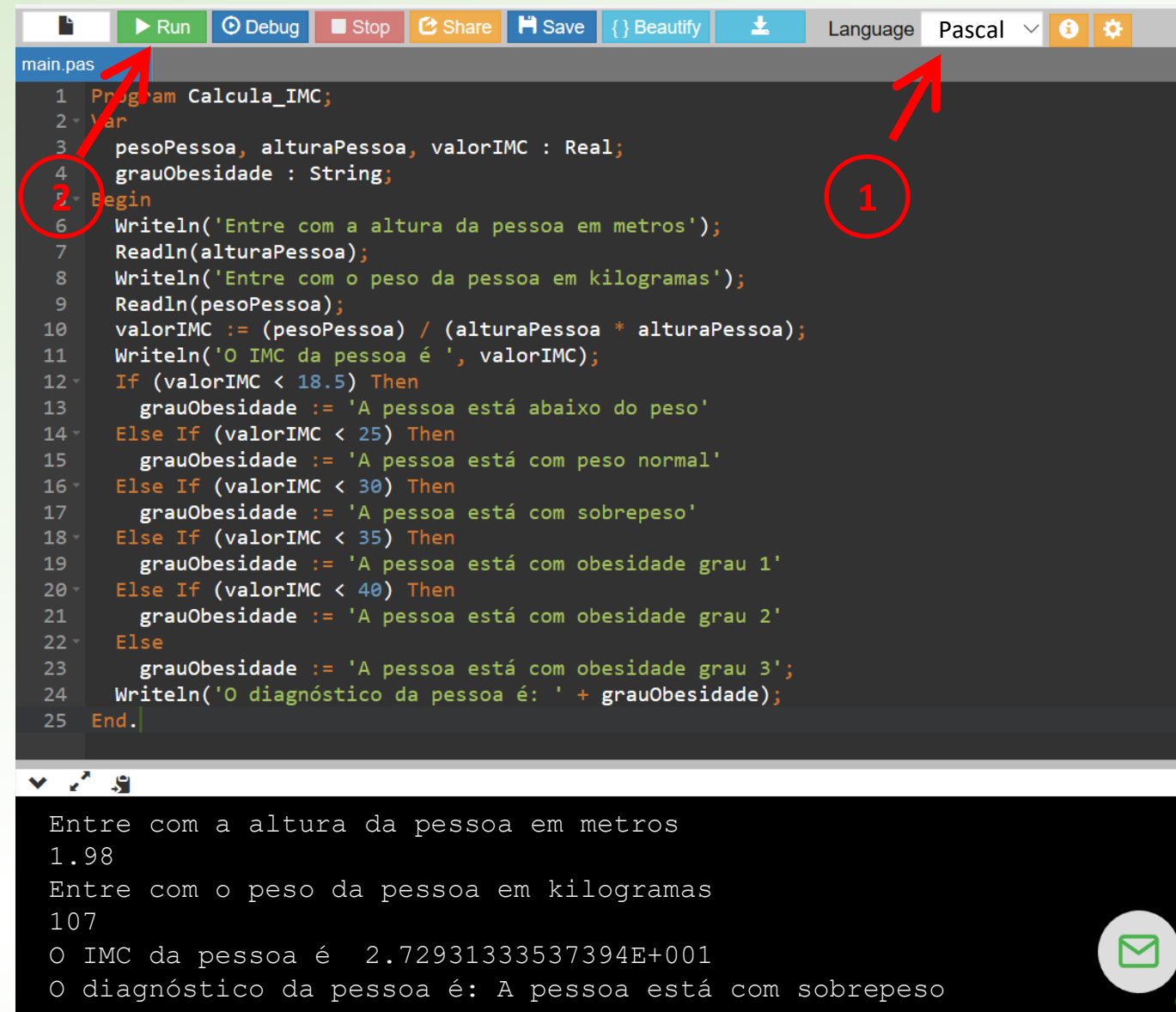
Below the code editor, there's a status bar showing 'Position: Ln 19, Ch 48' and 'Total: Ln 20, Ch 1028'. At the bottom, there are three tabs: Console View, Applet View, and Log View. The Console View is active, showing the output of the program:

```
Entre com a altura da pessoa em metros
1.8
Entre com o peso da pessoa em kilogramas
76
O IMC da pessoa é 23.45679
O diagnóstico da pessoa é: A pessoa está com peso normal
```

Alguns web sites permitem rodar programas Java (e outras linguagens), como o [Browxy](http://www.browxy.com) (<http://www.browxy.com>). Para rodar o programa Java que utilizamos como exemplo, copie seu código como mostrado na figura acima (se preferir, baixe o código do programa [aqui](#)), não se esquecendo de utilizar o argumento “-encoding UTF-8” como indicado pela seta “1” e clicar no botão *Start* (seta “2”) para iniciar a execução.

Exemplo Ilustrativo (2)

Rodando um programa Pascal no navegador internet



```
main.pas
1 Program Calcula_IMC;
2 Var
3   pesoPessoa, alturaPessoa, valorIMC : Real;
4   grauObesidade : String;
5
6 Begin
7   Writeln('Entre com a altura da pessoa em metros');
8   Readln(alturaPessoa);
9   Writeln('Entre com o peso da pessoa em kilogramas');
10  Readln(pesoPessoa);
11  valorIMC := (pesoPessoa) / (alturaPessoa * alturaPessoa);
12  Writeln('O IMC da pessoa é ', valorIMC);
13  If (valorIMC < 18.5) Then
14    grauObesidade := 'A pessoa está abaixo do peso'
15  Else If (valorIMC < 25) Then
16    grauObesidade := 'A pessoa está com peso normal'
17  Else If (valorIMC < 30) Then
18    grauObesidade := 'A pessoa está com sobrepeso'
19  Else If (valorIMC < 35) Then
20    grauObesidade := 'A pessoa está com obesidade grau 1'
21  Else If (valorIMC < 40) Then
22    grauObesidade := 'A pessoa está com obesidade grau 2'
23  Else
24    grauObesidade := 'A pessoa está com obesidade grau 3';
25  Writeln('O diagnóstico da pessoa é: ' + grauObesidade);
26 End.
```

Entre com a altura da pessoa em metros
1.98
Entre com o peso da pessoa em kilogramas
107
O IMC da pessoa é 2.72931333537394E+001
O diagnóstico da pessoa é: A pessoa está com sobrepeso

Uma alternativa ao Browsersy é o website do onlinegdb (https://www.onlinegdb.com/online_pascal_compiler), que permite compilar múltiplas linguagens on line, como o Pascal. Para rodar o programa Pascal de nosso exemplo, digite-o no editor do site (ou baixe-o [aqui](#)), certificando-se que a linguagem escolhida seja Pascal como indicado pela seta “1” e clicando no botão *Run* (seta “2”) para executá-lo.

O **Pascal** é uma linguagem de programação criada para o ensino de boas práticas de programação. Muitas pessoas aprendem a programar em Pascal e existe uma boa razão para isso – *trata-se de uma linguagem com sintaxe simples cujos comandos lembram palavras do cotidiano (em inglês)*. Por exemplo, o comando **Escreva** é transcrito como **write**, e o comando **Leia** como **read**. Em contraste, o java utiliza **System.out.println** como comando de escrita e, para leitura, utiliza **sc.nextFloat()** para números reais, **sc.nextInt()** para inteiros e **sc.nextLine()** para string (desde que uma instância da classe Scanner tenha sido instanciada previamente com o nome de “sc”). Claramente, o Pascal é mais intuitivo mas, de todo modo, note que *a lógica de programação é a mesma em qualquer linguagem imperativa*.

Comparando os Algoritmos 6 e 7 com o programa em Java (1)

- O Algoritmo 7 é mais formal do que o Algoritmo 6, mas não tanto quanto o programa em Java;
- O Algoritmo 6 está descrito numa forma coloquial:
 - Se distancia muito da lógica empregada no programa;
 - É **muito informal para ser transformado num programa**:
 - Uma descrição algorítmica coloquial está sujeita a interpretação e dá margem a ambiguidades.
- Cada passo do Algoritmo 7 é descrito com maior precisão:
 - O pseudocódigo não se atém a certas minúcias de sintaxe do Java que são irrelevantes para se **compreender** a solução;
 - A maior formalidade do pseudocódigo o torna mais rigoroso e mais apropriado para escrever um algoritmo.

Comparando os Algoritmos 6 e 7 com o programa em Java (2)

- O Algoritmo 7 é mais metódico que o 6 e se assemelha mais ao programa. Contudo, diferentemente do Algoritmo 7, o programa em Java:
 1. Deve obedecer regras rígidas quanto à sintaxe e o uso dos comandos;
 2. Distingue maiúsculas de minúsculas;
 3. O início e o fim de cada bloco de comandos são delimitados por chaves (“{” e “}”);
 4. Depende de componentes externos (como a classe `java.util.Scanner`);
 5. É orientado a objetos, ao passo que o algoritmo é estruturado (mas essa informação é totalmente irrelevante para nós nesse momento).

Guia de Referência Rápida

Operadores Matemáticos Mais Comuns

Símbolo	Operador	Descrição	Exemplo
+	Adição	Adiciona dois números	$10 + 20 + 30 + 40 = 100$
-	Subtração	Subtrai um número de outro	$10 - 20 - 30 - 40 = -80$
*	Multiplicação	Multiplica um número por outro	$10 * 20 * 30 * 40 = 240000$
/	Divisão	Divide um número por outro e retorna o quociente	$40 / 20 / 10 = 0,2$
MOD	Módulo	Divide um número por outro e retorna o resto da divisão inteira	$40 \text{ mod } 30 = 10$
^	Exponenciação	Eleva um número a uma potência expressa por outro número	$10 ^ 2 = 100$

Atividades de Fixação

Taxa por Uso do Celular Acima da Franquia (1)

Suponha que seu plano de celular cobre uma taxa fixa para que você use utilize seu aparelho por 700 minutos por mês. Caso você ultrapasse esse limite, cobra-se uma taxa adicional de 35 centavos por minuto excedente. Sua conta de telefone mostra quantos minutos excedentes você usou a cada mês, mas não mostra quanto que você está pagando por esse excesso. Para ter essa informação, você faz contas da maneira antiga (com lápis e papel), mas gostaria de criar um algoritmo que simplifique a tarefa. A ideia é que você insira o número de minutos em excesso e o algoritmo execute esse cálculo.

Taxa por Uso do Celular Acima da Franquia (2)

Como no caso anterior, suponha que seu plano de celular tenha uma franquia de 700 minutos, e que cobre uma taxa adicional de 35 centavos por minuto excedente. Sua conta de telefone mostra o valor da franquia e o valor total a ser pago (que pode ser igual ou maior que a franquia), mas não quantos minutos excedentes você utilizou num mês. Crie um algoritmo que determine esses minutos excedentes.

Taxa por Uso do Celular Acima da Franquia (3)

Agora suponha que seu plano de celular tenha uma franquia de 700 minutos, e que sua conta de telefone mostre o valor da franquia, o valor total a ser pago (que pode ser igual ou maior que a franquia) e a quantidade de minutos excedentes utilizados num mês. Contudo, você não sabe quanto a operadora lhe cobra por cada minuto excedente. Crie um algoritmo que determine quando você pagou por cada minuto excedente.

Taxa por Uso do Celular Acima da Franquia (4)

Cansado de escrever algoritmos para saber tudo o que você precisa sobre o consumo do seu plano de celular, você contrata uma outra operadora. Contudo, a conta da nova operadora mostra apenas o valor da franquia, o total da fatura, o quanto se paga por cada minuto dentro da franquia e quanto se paga pelos minutos que excedem a franquia, mas não revela por quanto tempo você utilizou seu aparelho. Portanto, você não tem alternativa senão escrever um algoritmo novamente, que lhe permita determinar por quanto tempo você utilizou seu aparelho num determinado mês.

Alguns Exercícios Propostos (1)

Crie algoritmos para resolver os seguintes problemas:

1. Calcule a média de um aluno, definida como a média aritmética de 4 avaliações;
2. Calcule a média de um aluno, considerando que as notas parciais são de dois exercícios (chamados de $E1$ e $E2$), uma avaliação parcial (VP) e uma prova final (PF) e que o exercício $E1$ vale 10% da média final, o $E2$ vale 20%, a VP vale 30% e a PF vale 40%;
3. Leia dois números, $N1$ e $N2$, os compare e diga qual deles é o maior;
4. Sabendo que a soma dos ângulos internos de um triângulo é 180° e a dos ângulos de um quadrilátero convexo é 360° , construa os seguintes algoritmos:

Algoritmo 1 – Dados dois ângulos de um triângulo, calcule o terceiro ângulo;

Algoritmo 2 – Dados três ângulos de um quadrilátero, calcule o quarto ângulo;

Algoritmo 3 – Pergunte se os ângulos que o usuário fornecerá serão de um triângulo ou de um quadrilátero. No primeiro caso, utilize a lógica que você criou para o Algoritmo 1 e, no segundo, utilize a lógica do Algoritmo 2.

Alguns Exercícios Propostos (2)

5. Considere as seguintes informações a respeito do cálculo da média de um aluno:
- a) A média é calculada com base em dois exercícios (chamados de $E1$ e $E2$), uma avaliação parcial (VP) e uma prova final (PF);
 - b) O exercício $E1$ vale 10% da média final, o $E2$ vale 20%, a VP vale 30% e a PF vale 40%;
 - c) O aluno estará aprovado se possuir média superior a 7, em recuperação se sua média for superior a 4 porém inferior a 7 ou reprovado se a média menor ou igual a 4;
 - d) Caso o aluno fique em recuperação, ele tem direito a fazer um exame e será considerado aprovado se a nota do exame mais a média for igual ou maior a 10.
- Cria um algoritmo para calcular a média de um aluno, determinar seu status (aprovado, recuperação ou reprovado) e, caso esteja em recuperação, diga a nota que ele tem que tirar no exame.
6. Construa um algoritmo que transforme um certo número de horas, expresso no formato decimal, para minutos (por exemplo, 1.5 hora é igual a 90 minutos) e para segundos (1 hora = 60 minutos, e 1 minuto = 60 segundos);
7. Usando o Teorema de Pitágoras, calcule o valor da hipotenusa a partir dos catetos ou, conforme a escolha do usuário, calcule um dos catetos utilizando o valor da hipotenusa e do cateto restante.

Conceitos Abordados

- Algoritmo;
- Ciclo EPS (*Entrada, Processamento e Saída*);
- Computador;
- Computador (ou sistema) embarcado;
- Fluxograma;
- Hardware;
- Java;

- Linguagem de programação;
- Lógica de programação;
- Operadores relacionais;
- Operador de atribuição;
- Pascal;
- Programa;
- Pseudocódigo;
- Software.