Instituto Superiror Politécnico
de Porto Amboim

MÓDULO 1
FUNDAMENTOS DE ALGORITMOS
E LÓGICA DE PROGRAMAÇÃO

GUIA ACADÉMICO | CURSO DE ENGENHARIA



Autores:

M. Sc. ALEXEI GAMBOA MOREIRA
M. Sc. GILDO DIMITRI BENTO PAULO
ENG.º HERMENEGILDO BUANDO

NORMA TECNOLÓGICA: ISO 5807 (SIMBOLOGIA PARA FLUXOGRAMAS)

Capítulo 1: Introdução aos Algoritmos e à Lógica de Programação

1.1. Arquitectura Fundamental IPO (Input-Process-Output)

Quando um estudante inicia o curso de Engenharia Informática no ISUP, o primeiro grande objectivo consiste em aprender a estruturar o pensamento de forma computacional. Na vida quotidiana, resolvemos problemas complexos através de passos lógicos intuitivos (como seguir uma receita de culinária, trocar um pneu ou calcular o troco na cantina da universidade).

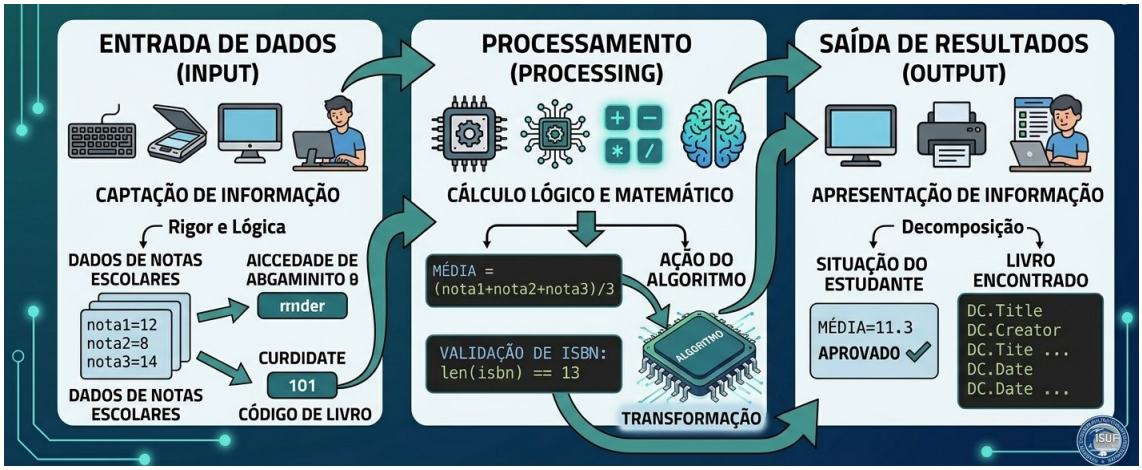
Um Algoritmo formaliza este processo: define-se como uma sequência finita, ordenada e não ambígua de instruções elementares que, quando executadas passo a passo, conduzem à resolução de um problema específico.

Todo o algoritmo assenta no modelo *IPO* (VER Figura 1)

- *Entrada (Input):* Captura dos dados brutos necessários para a resolução do problema.
- *Processamento (Processing):* Execução de cálculos matemáticos, transformações lógicas e tomadas de decisão.
- *Saída (Output):* Apresentação dos resultados finais processados ao utilizador.

Figura 1

Arquitectura básica de processamento Computacional (IPO)



1.2. Modos de Representação de Algoritmos

1.2.1. A. Linguagem Natural

A representação em Linguagem Natural consiste na descrição textual, em linguagem humana (português), do procedimento a executar.

Exemplo (Receita de Bolo Simples):

- *Entrada de Dados:* 250g de farinha, 200g de açúcar, 4 ovos, 150ml de leite, 100g de manteiga.
- *Processo Lógico:*
    1. Pré-aquecer o forno a 180 °C e untar a forma.
    2. Misturar a manteiga com o açúcar até obter um creme homogéneo.
    3. Adicionar os ovos um a um, batendo bem.
    4. Incorporar a farinha e o leite alternadamente.
    5. Colocar a massa na forma e levar ao forno por 40 minutos.
- *Saída de Resultados:* Bolo cozido pronto a servir.

Vantagem: Acessível e intuitivo.

Desvantagem: Apresenta alta ambiguidade e falta de rigor formal, sendo inviável para execução direta por um computador (Guimarães & Lacerda, 2023).

1.2.2. Fluxogramas Normalizados (ISO 5807)

O Fluxograma é a representação gráfica de um algoritmo. Para evitar ambiguidades mundiais, a ISO 5807 padroniza a simbologia geométrica (ver Tabela 1):

Tabela 1
Formas Geométricas da (ISSO 5807)

Forma Geométrica	Nome do Símbolo ISO	Função no Algoritmo	Exemplo de Aplicação Prática
Óvalo / Elipse	Terminal	Delimita o início e o termo da execução	Início ou Fim do programa
Paralelo-gramo	Entrada / Saída	Interação de dados com o utilizador	Digitar nota / Mostrar média
Rectângulo	Processamento	Execução de cálculos e atribuições	Area = Comprimento * Largura
Losango	Decisão / Condição	Desvio de fluxo baseado em lógica	Se (Media >= 10) Então...
Setas / Linhas	Fluxo de Execução	Direciona a ordem dos passos	Conectar operações sequenciais

ISO 5807 (*Information processing — Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*). Esta norma define padronizações universais para as formas geométricas (como terminais, paralelogramos de entrada/saída, rectângulos de processamento e losangos de decisão) de modo a garantir que qualquer engenheiro ou programador consiga interpretar a lógica de um algoritmo independentemente da região ou idioma.

1.2.3. O Pseudo-código (Portugol)

O Portugol actua como o elo de ligação pedagógico entre a representação gráfica e o código real em linguagens de programação. Utiliza vocabulário reservado em português (algoritmo, var, inicio, se, entao, senao, fim_algoritmo), permitindo ao estudante focar-se puramente na arquitectura lógica sem a frustração de erros de sintaxe rígidos.

O Portugol funciona como uma linguagem intermédia ou um "elo de ligação" estruturado. Enquanto a linguagem natural é demasiado ambígua e a linguagem de programação (como C++, Python ou Java) exige uma sintaxe estrita, pontuação exacta (como ponto e vírgula, chavetas e parênteses) e regras rígidas de compilação, o Portugol utiliza uma estrutura em língua portuguesa com vocabulário técnico reservado (palavras-chave como algoritmo, var, início, fim, se, entao, senao, enquanto).

Esta abordagem pedagógica permite que o estudante concentre todo o seu esforço cognitivo na lógica algorítmica e na arquitectura da solução, sem o desgaste prematuro provocado por erros de sintaxe próprios de uma linguagem de programação específica.

Um algoritmo descrito em Portugol divide-se tipicamente em três grandes secções lógicas:

1. *Cabeçalho:* Identificação formal do nome do algoritmo.
2. *Declaração de Variáveis (var):* Espaço onde se definem os tipos de dados que serão manipulados (inteiros, reais, carateres ou lógicos).
3. *Corpo do Algoritmo (inicio ... fim):* A sequência ordenada de instruções, comandos de entrada/saída, estruturas condicionais e ciclos repetitivos.

Exemplo Prático em Portugol

Para ilustrar como o Portugol serve de ponte estrutural, consideremos o problema do cálculo do aproveitamento académico: ler a nota de um estudante e determinar se está "Aprovado" nota>10 ou "Reprovado" nota< 10.

A. Representação em Portugol (Pseudo-código)

```
algoritmo VerificarAprovacaoISUP;
var
   nota: real;
inicio
   escrever("Digite a nota do estudante:");
   ler(nota);
   se (nota >= 10) então escrever("Situação: Aprovado");
   senao escrever("Situação: Reprovado"); fim_se;
fim_algoritmo.
```

1.3 Variáveis, Operadores e Expressões

1.3.1. O Conceito de Variável e Alocação na Memória RAM

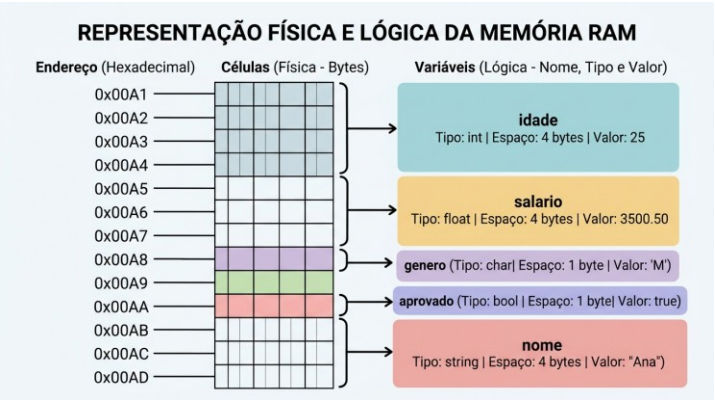
Um dos primeiros desafios conceptuais é compreender como um computador processa e armazena informação digital. Na vida quotidiana, lidamos constantemente com dados: a quantidade de sacos de cimento numa obra em Porto Amboim, a temperatura média em Sumbe, o nome de um aluno matriculado ou a confirmação do pagamento de uma propina. No entanto, para que uma máquina execute operações sobre estes dados, eles precisam de ser representados na memória de forma estruturada numa *Variável.*

1.3.2 O Conceito de Variável e a Memória do Computador

Uma variável pode ser definida como um espaço reservado e rotulado na memória RAM (Random Access Memory) do computador, cujo conteúdo pode ser modificado durante a execução de um programa. Imagine a memória como um grande armazém organizado com centenas de gavetas etiquetadas. Quando declaramos uma variável, estamos a pedir ao sistema operativo que reserve uma gaveta específica, lhe atribua um nome (identificador) e defina que tipo de objecto pode ser lá guardado.
A declaração de uma variável exige duas informações fundamentais:

1. *O Identificador (Nome):* O nome atribuído à variável para ser referenciada no código (ex: preco_produto, nota_final). Os identificadores devem ser semânticos, não podem começar por números, não podem conter espaços nem caracteres especiais ou acentuados.
2. *O Tipo de Dado:* Define a natureza da informação e a quantidade de memória necessária para o seu armazenamento.

Figura 2
Estrutura de Alocação de Variáveis na Memória RAM



Uma *Variável* é um espaço reservado e etiquetado na memória RAM (*Random Access Memory*) do computador, cujo conteúdo pode ser alterado durante a execução do programa (ver Figura 2).

1.3.3. Mapeamento de Tipos de Dados Primitivos

Os dados utilizados na resolução de problemas de engenharia dividem-se em categorias fundamentais:

- *Inteiros (int):* Representam números sem componentes fraccionárias ou decimais. Exemplo contextualizado: O número de alunos numa sala de aula no ISUP, a contagem de sacos de milho colhidos numa fazenda do Cuanza Sul ou a idade de um estudante.
- *Reais / Ponto Flutuante (float / double):* Representam quantidades contínuas que possuem casas decimais. Em Java e C++, o tipo double oferece dupla precisão, sendo o preferido para cálculos financeiros ou de precisão científica (ex: o preço de uma mercadoria em Kwanzas 35000.75 Kz, a média final de uma cadeira ou a distância entre Luanda e Porto Amboim).
- *Caractere (char):* Armazena um único caractere alfanumérico, delimitado por aspas simples ('M', 'F', 'A'). Exemplo: O género de um utente ou a letra de uma opção num menu.
- *Lógico / Booleano (bool em C++ / boolean em Java):* Armazena apenas dois estados possíveis: Verdadeiro (true) ou Falso (false). Exemplo: O estado de aprovação de um aluno (aprovado = true) ou a verificação de um pagamento.
- *Cadeia de Caracteres / Texto (string em C++ / String em Java):* Representa uma sequência de caracteres alfanuméricos (palavras, frases, códigos). Exemplo: O nome completo de um professor ou o endereço institucional.

Na Tabela 2 são detalhados os tipos de dados fundamentais, comparando a sintaxe de declaração, a tipagem estática versus dinâmica, o tamanho em memória e exemplos de uso contextualizados para a Engenharia Informática.

Tabela 2
Matriz Comparativa de Tipos de Dados Primitivos entre C++, Python e Java

Tipo Conceptual	Descrição do Dado	Linguagem C++	Linguagem Python	Linguagem Java	Exemplo de Aplicação Prática no ISUP
Número Inteiro	Valoração sem casas decimais	int idade = 25;	idade = 25	int idade = 25;	Contagem de estudantes matriculados
Número Real	Valoração decimal de precisão	double salario = 35000.75;	salario = 35000.75	double salario = 35000.75;	Cálculo de propina ou salário em Kz
Caractere Único	Símbolo alfanumérico isolado	char genero = 'M';	genero = 'M'	char genero = 'M';	Código da turma ou inicial de identificação
Valor Lógico	Estado binário (Verdadeiro/Falso)	bool activo = true;	activo = True	boolean activo = true;	Verificação de inscrição de exame
Texto / String	Sequência alfanumérica contínua	string nome = "João";	nome = "João"	String nome = "João";	Nome do estudante ou cadeira académica

1.3.4. Operadores Aritméticos e Avaliação de Expressões

Os operadores permitem executar transformações e cálculos sobre os valores armazenados. As operações aritméticas seguem a hierarquia matemática tradicional:

1. *Soma (+) e Subtração* **(-):** Operações aditivas elementares.
2. *Multiplicação (*) e Divisão* **(/):** Operações multiplicativas com precedência superior.
3. *Módulo / Resto da Divisão Inteira (%):* Retorna o resto de uma divisão entre dois inteiros (essencial para determinar paridade, múltiplos ou ciclos).

Precedência de Operadores: Ao avaliar uma expressão como media = (nota1 + nota2 + nota3) / 3, os parêntesis alteram a ordem natural de execução, garantindo que as somas das notas ocorram obrigatoriamente antes da divisão.

1.4 Secção de Exercícios Resolvidos (Taxonomia Multi-linguagem)

Exercício Resolvido 1. Cálculo da Área de um Terreno Agrícola

- *Contexto:* Um produtor agrícola em Porto Amboim necessita de determinar a área útil de um terreno rectangular para plantação de milho.
- *Entrada:* Comprimento e largura em metros (valores decimais).
- *Saída:* Área total em metros quadrados (m^2).

Algoritmo

Figura 3

Fluxograma do Cálculo da Área de um Terreno



Pseudo-código (Portugol):

```
algoritmo CalcularAreaTerreno;
var
        comprimento, largura, area: real;
inicio
        escrever("Digite o comprimento (m): ");
        ler(comprimento);
        escrever("Digite a largura (m): ");
```

```
        ler(largura);
        area <- comprimento * largura;
         escrever("Área total do terreno = ", area, " m^2");
fim_algoritmo.
```

Código C++

```
#include <iostream>
using namespace std;

int main() {
   double comprimento, largura, area;

   cout << "Digite o comprimento do terreno (m): ";
   cin >> comprimento;
   cout << "Digite a largura do terreno (m): ";
   cin >> largura;

   area = comprimento * largura;

   cout << "Área total do terreno = " << area << " m^2" << endl;
   return 0;
}
```

Código Python

```
comprimento = float(input("Digite o comprimento do terreno (m): "))
largura = float(input("Digite a largura do terreno (m): "))

area = comprimento * largura

print(f"Área total do terreno = {area:.2f} m^2")
```

Código Java

```
import java.util.Scanner;

public class AreaTerreno {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      System.out.print("Digite o comprimento do terreno (m): ");
      double comprimento = input.nextDouble();

      System.out.print("Digite a largura do terreno (m): ");
      double largura = input.nextDouble();

      double area = comprimento * largura;

      System.out.println("Área total do terreno = " + area + " m^2");
      input.close();
```

```
    }
}
```

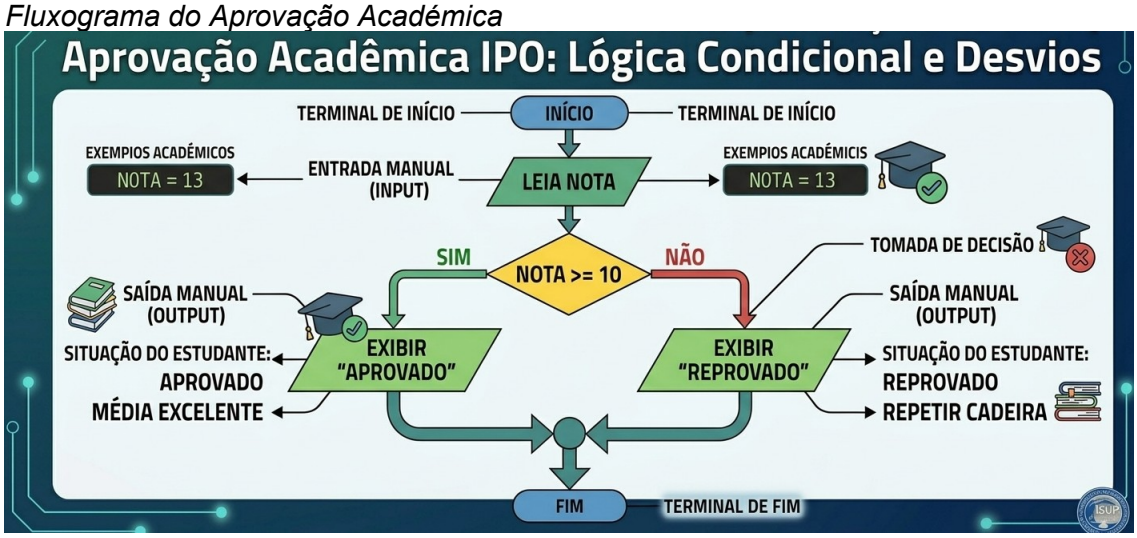
Exercício Resolvido 2 Verificação de Aproveitamento Académico

- *Contexto:* um professor do ISUP precisa de determinar se um estudante obteve aproveitamento numa disciplina, sabendo que a nota de aprovação é igual ou superior a 10 valores.
- *Entrada:* leer a nota final.
- *Saída:* verificar se a nota é maior ou igual a 10. Se sim, exibir "Aprovado"; caso contrário, exibir "Reprovado".

Algoritmo

Figura 4

Fluxograma do Aprovação Académica



Código C++

```
#include <iostream>
using namespace std;

int main() {
   double nf;
   cout << "Digite a nota final: ";
   cin >> nf;

   if (nf >= 10.0) {
      cout << "Situação: APROVADO" << endl;
   } else {
      cout << "Situação: REPROVADO" << endl;
   }
   return 0;
}
```

Código Python

```
nf = float(input("Digite a nota final: "))
```

```
if nf>= 10.0:
   print("Situação: APROVADO")
else:
   print("Situação: REPROVADO")
```

Código Java

```
import java.util.Scanner;

public class MediaNotas {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite a nota final: ");
      double nf = sc.nextDouble();
      if (nf>= 10.0) {
         System.out.println("Situação: APROVADO");
      } else {
         System.out.println("Situação: REPROVADO");
      }
      sc.close();
   }
}
```

Exercício Resolvido 3. Cálculo de Média Aritmética Semestral

- *Contexto:* Um docente do ISUP precisa de calcular a média final de três provas parciais de um estudante.
- *Entrada:* Três notas numéricas reais.
- *Saída:* Média calculada e indicação se foi aprovado (texto ”Média >= 10”).

Algoritmo

Igual ao exercício anterior só se acrescenta um rectângulo para representar o calcula da nota media

```
media=(n1+n2+n3)/3
```

Código C++

```
#include <iostream>
using namespace std;

int main() {
   double n1, n2, n3, media;
   cout << "Digite a primeira nota: ";
   cin >> n1;
   cout << "Digite a segunda nota: ";
   cin >> n2;
   cout << "Digite a terceira nota: ";
   cin >> n3;

   media = (n1 + n2 + n3) / 3.0;
```

```
   cout << "Média semestral: " << media << endl;
   if (media >= 10.0) {
      cout << "Situação: APROVADO" << endl;
   } else {
      cout << "Situação: REPROVADO" << endl;
   }
   return 0;
}
```

Código Python

```
n1 = float(input("Digite a primeira nota: "))
n2 = float(input("Digite a segunda nota: "))
n3 = float(input("Digite a terceira nota: "))

media = (n1 + n2 + n3) / 3.0

print(f"Média semestral: {media:.2f}")
if media >= 10.0:
   print("Situação: APROVADO")
else:
   print("Situação: REPROVADO")
```

Código Java

```
import java.util.Scanner;

public class MediaNotas {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite a primeira nota: ");
      double n1 = sc.nextDouble();
      System.out.print("Digite a segunda nota: ");
      double n2 = sc.nextDouble();
      System.out.print("Digite a terceira nota: ");
      double n3 = sc.nextDouble();

      double media = (n1 + n2 + n3) / 3.0;

      System.out.printf("Média semestral: %.2f\n", media);
      if (media >= 10.0) {
         System.out.println("Situação: APROVADO");
      } else {
         System.out.println("Situação: REPROVADO");
      }
      sc.close();
   }
}
```

Exercício Resolvido 4. Preço Final de Compra com Desconto

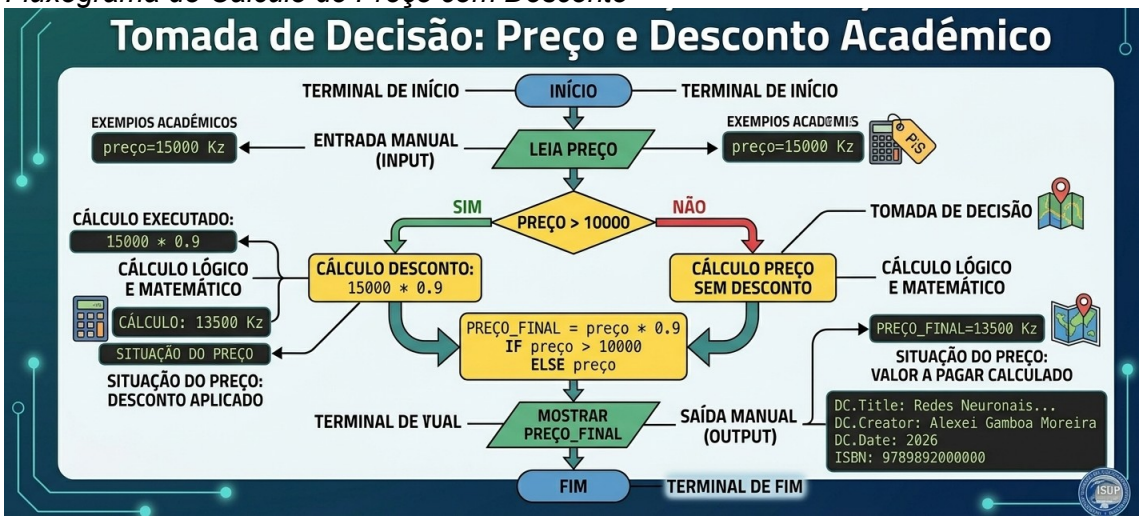
- *Contexto:* Uma loja comercial em Sumbe oferece 10% de desconto imediato em compras cujos valores sejam superiores a 10.000 Kz.
- *Entrada:* Preço original do produto.

- *Saída:* Preço final a pagar em Kwanzas.

Algoritmo

Figura 5

Fluxograma do Cálculo de Preço com Desconto



Código C++

```
#include <iostream>
using namespace std;

int main() {
   double preco, precoFinal;
   cout << "Digite o preço original do produto (Kz): ";
   cin >> preco;

   if (preco > 10000) {
      precoFinal = preco * 0.90;
   } else {
      precoFinal = preco;
   }

   cout << "Preço final a pagar: " << precoFinal << " Kz" << endl;
   return 0;
}
```

Código Python

```
preco = float(input("Digite o preço original do produto (Kz): "))

if preco > 10000:
   preco_final = preco * 0.90
else:
   preco_final = preco

print(f"Preço final a pagar: {preco_final:.2f} Kz")
```

Código Java

```
import java.util.Scanner;
```

```
public class PrecoDesconto {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite o preço original do produto (Kz): ");
      double preco = sc.nextDouble();

      double precoFinal = (preco > 10000) ? preco * 0.90 : preco;

      System.out.println("Preço final a pagar: " + precoFinal + " Kz");
      sc.close();
   }
}
```

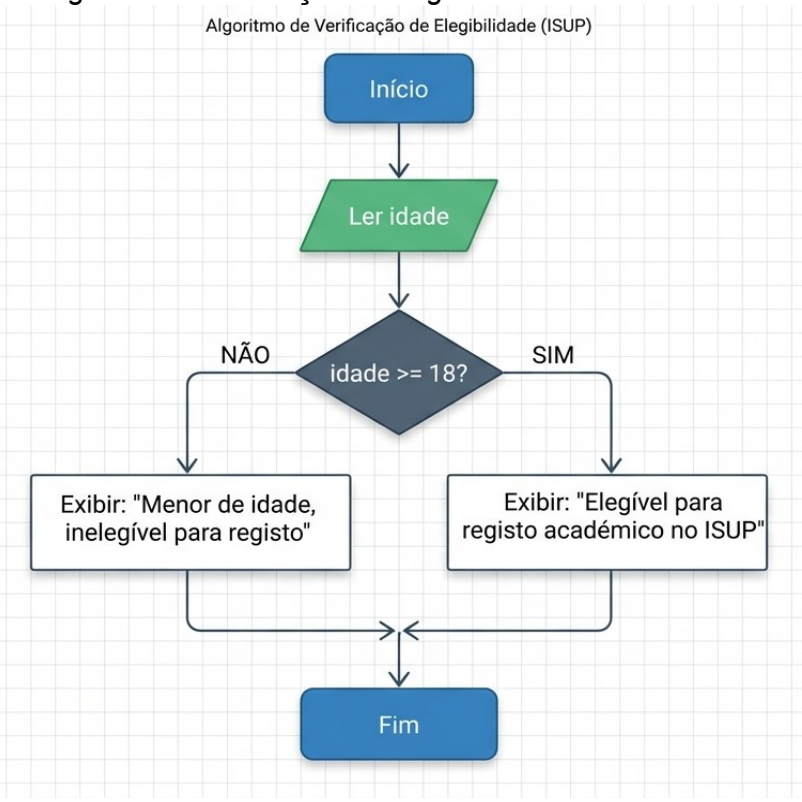
Exercício Resolvido 5. Validação de Maioridade para Inscrição

- *Contexto:* Verificar se um candidato possui idade legal (>=18 anos) para efectuar a matrícula autónoma no ensino superior.
- *Entrada:* Ler a idade.
- *Saída:* avaliar se a idade é maior ou igual a 18. Se sim, exibir mensagem “Elegível para o Registro Académico no ISUP” "; caso contrário, exibir "Menor de Idade, Inelegível para o registro".

  Algoritmo

Figura 6

Fluxograma do Verificação de Eligibilidade



Código C++

```
#include <iostream>
using namespace std;

int main() {
   int idade;
   cout << "Digite a sua idade: ";
   cin >> idade;

   if (idade >= 18) {
      cout << " Elegível para o Registro Académico no ISUP " << endl;
   } else {
      cout << " Menor de Idade, Inelegível para o registro " << endl;
   }
   return 0;
}
```

Código Python

```
if float(input("Digite a sua idade: "))>= 18:
   print("Elegível para o Registro Académico no ISUP")
else:
   print("Menor de Idade, Inelegível para o registro ")
```

Código Java

```
import java.util.Scanner;
public class MaiorIdade {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite a sua idade ");
      double idade = sc.nextInt();
      if (idade>= 18.0) {
         System.out.println("Elegível para o Registro Académico no ISUP");
      } else {
         System.out.println("Menor de Idade, Inelegível para o ");
      }
      sc.close();
   }
}
```

Exercício Resolvido 6 Contagem Simples de 1 até n

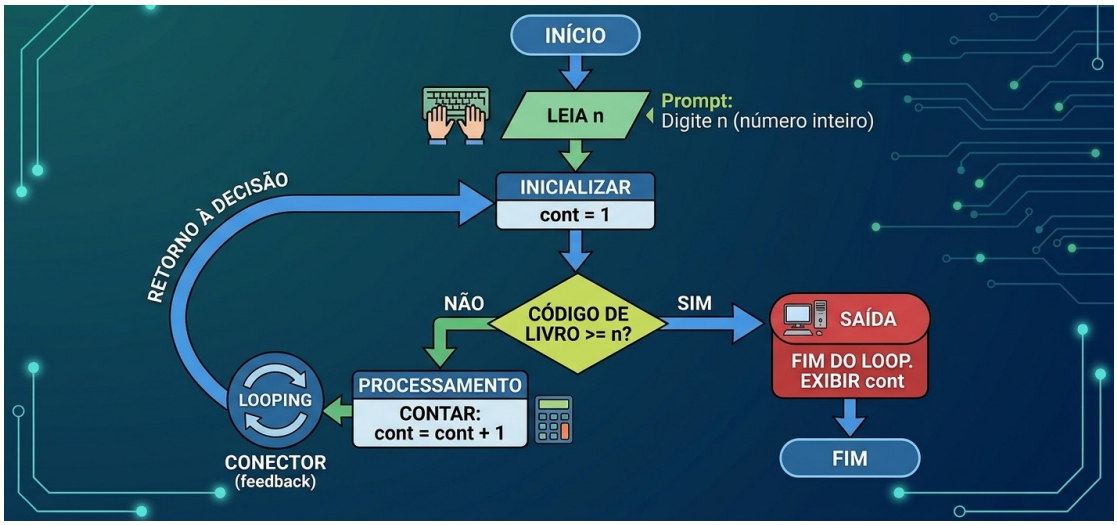
Contexto: Apresentar uma sequência numérica controlada de 1 a n no ecrã que representa a contagem de livros na biblioteca híbrida do ISUP.

Entrada: valor de n n ∈ aso números inteiros desde 1 até 10^6

Saída: imprimir o valor, incrementar e repetir até concluir o ciclo.

Algoritmo

Figura 7
Fluxograma de Contagem Repetitiva



Código C++

```
#include <iostream>
using namespace std;

int main() {
   int n;
   cout << "Digite um número inteiro n: ";
   cin >> n;

   int cont = 1;

   // Ciclo de contagem repetitiva
   while (cont <= n) {
      cout << "Contagem atual: " << cont << endl;
      cont++;
   }

   cout << "Fim do loop." << endl;
   return 0;
}
```

Código Python

```
# Leitura do valor limite n
n = int(input("Digite um número inteiro n: "))
cont = 1

# Ciclo de contagem repetitiva (equivalente ao fluxograma)
while cont <= n:
   print(f"Contagem atual: {cont}")
   cont += 1
```

```
print("Fim do loop.")
```

Código Java

```
import java.util.Scanner;

public class ContagemAteN {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite um número inteiro n: ");
      int n = sc.nextInt();

      int cont = 1;

      // Ciclo de contagem repetitiva
      while (cont <= n) {
         System.out.println("Contagem atual: " + cont);
         cont++;
      }

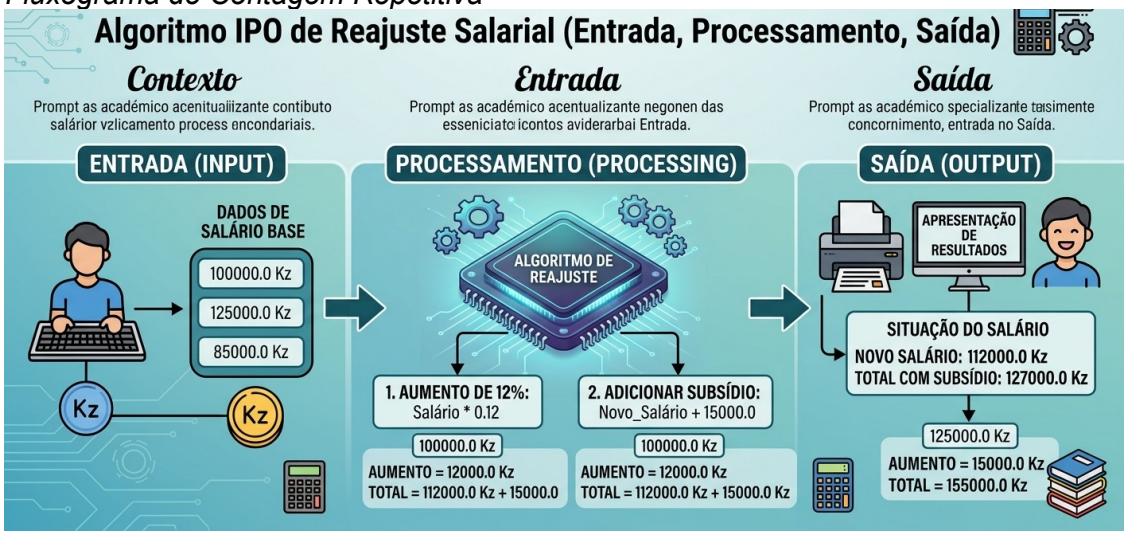
      System.out.println("Fim do loop.");
      sc.close();
   }
}
```

Exercício Resolvido 7 Conversão de Salário com Subsídio de Transporte

- *Contexto:* A administração de uma empresa em Porto Amboim necessita de reajustar o salário base dos seus técnicos, aplicando um aumento de 12% e adicionando um subsídio fixo de transporte no valor de 15.000 Kz.
- *Entrada:* Salário base actual em Kwanzas.
- *Saída:* Salário ilíquido ajustado em Kwanzas.

Figura 8

Fluxograma de Contagem Repetitiva



Código C++

```
#include <iostream>
```

```
using namespace std;

int main() {
   double salarioBase, salarioComAumento, salarioFinal;
   const double SUBSIDIO_TRANSPORTE = 15000.0;

   cout << "Digite o salário base do funcionário (Kz): ";
   cin >> salarioBase;

   salarioComAumento = salarioBase * 1.12;
   salarioFinal = salarioComAumento + SUBSIDIO_TRANSPORTE;

   cout << "Salário ajustado final: " << salarioFinal << " Kz" << endl;
   return 0;
}
```

Código Python

```
SUBSIDIO_TRANSPORTE = 15000.0

salario_base = float(input("Digite o salário base do funcionário (Kz): "))

salario_com_aumento = salario_base * 1.12
salario_final = salario_com_aumento + SUBSIDIO_TRANSPORTE

print(f"Salário ajustado final: {salario_final:.2f} Kz")
```

Código Java

```
import java.util.Scanner;

public class ReajusteSalario {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      final double SUBSIDIO_TRANSPORTE = 15000.0;

      System.out.print("Digite o salário base do funcionário (Kz): ");
      double salarioBase = sc.nextDouble();

      double salarioComAumento = salarioBase * 1.12;
      double salarioFinal = salarioComAumento + SUBSIDIO_TRANSPORTE;

      System.out.printf("Salário ajustado final: %.2f Kz\n", salarioFinal);
      sc.close();
   }
}
```

1.5 Secção de Exercícios Propostos (Estudo Autónomo)

Exercício Proposto 1. Consumo Médio de Combustível

- *Contexto:* A frota de autocarros do ISUP realiza trajectos diários entre Sumbe e Porto Amboim. Desenvolva um programa que leia a distância total percorrida (em km) e o total de combustível gasto (em litros) e exiba o consumo médio do veículo em km/l.

- *Linguagens requeridas:* C++, Python e Java.

Exercício Proposto 2 Cálculo do Índice de Massa Corporal - IMC

- *Contexto:* O centro de saúde escolar necessita de uma ferramenta para calcular o IMC de cada estudante do 1.º ano.
- *Fórmula:* IMC = [peso(kg)/altura (m)]^2.
- *Linguagens requeridas:* C++, Python e Java.

Exercício Proposto.3 Conversão de Tempo de Execução

- *Contexto:* Um algoritmo de simulação computacional leva um determinado tempo em segundos para processar dados de sensoriamento em Cuanza Sul. Crie um programa que leia o tempo total em segundos e o converta para o formato de Horas, Minutos e Segundos.
- *Linguagens requeridas:* C++, Python e Java.

Exercício Proposto 4 Cálculo de Perímetro e Área do Círculo

- *Contexto:* Um projeto de topografia em Porto Amboim requer a medição precisa de reservatórios circulares de água. Escreva um programa que receba o raio do círculo e exiba a área Área = pi*r^2 e o perímetro Perímetro = 2 *pi*r. Considere pi = 3.14159.
- *Linguagens requeridas*: C++, Python e Java.

Exercício Proposto 5 Cálculo do Preço de Custo de Importação de Equipamento

- *Contexto:* O laboratório de informática importou computadores novos. O programa deve receber o valor original do equipamento, adicionar 14% de imposto (IVA) e adicionar uma taxa fixa de alfândega de 25.000 Kz, exibindo o custo total final por unidade.
- *Linguagens requeridas:* C++, Python e Java.

Exercício Proposto 6 Conversão de Câmbio de Kwanzas para Dólares

- *Contexto:* Um estudante do ISUP pretende realizar uma viagem académica e precisa de converter um montante em Kwanzas (Kz) para Dólares norte-americanos (USD), dada uma taxa de câmbio informada.
- *Entrada:* Valor em Kwanzas e taxa de câmbio actual (1 USD em Kz).
- *Saída:* Valor equivalente convertido em Dólares (USD).

1.6. Perguntas de Reflexão e Consolidação Pedagógica

1. Que tipo de dado escolheria para armazenar o número de matrícula de um estudante do ISUP (ex: "ISUP20260102")? Por que motivo o tipo int não seria apropriado nesta situação?

2. Em termos de gestão de memória RAM, qual é a diferença entre os tipos de ponto flutuante float e double nas linguagens C++ e Java?
3. O que sucede ao resultado de uma divisão de dois números inteiros (7 / 2) em C++ ou Java quando comparado com a mesma operação executada em Python 3? Como se pode forçar uma divisão real em C++ e Java?

1.7. Resumo Teórico

As variáveis são espaços reservados na memória RAM do computador utilizados para armazenar dados temporários durante a execução de um programa. Cada variável precisa de um tipo definido para saber que tipo de informação pode albergar.

1.7.1 Tipos Primitivos Básicos:

- int: Números inteiros (ex: idade, quantidade de produtos).
- float / double: Números com parte decimal ou fracionária (ex: salário, médias).
- char: Um único caractere alfanumérico (ex: género 'M' ou 'F').
- bool: Valores lógicos booleanos (true ou false).
- string: Sequência de caracteres para manipulação de texto.

Refrencias bibliográficas

Guimarães, R. M., & Lacerda, J. A. (2023). *Fundamentos de lógica de programação e arquitetura de algoritmos*. Editora Universitária.

TEMA 2: ESTRUTURAS DE CONTROLO (CONDICIONAIS E CICLOS)

2.1. Fundamentação Teórica e Contextualização

Até ao momento, os algoritmos que estudámos executavam-se de forma estritamente sequencial: uma instrução após a outra, de cima para baixo, sem desvios nem repetições. No entanto, no mundo real e na resolução de problemas de engenharia, os programas precisam de tomar decisões e automatizar tarefas repetitivas. Por exemplo, o sistema académico do ISUP não processa a situação de um estudante da mesma forma se ele tiver média 12 (aprovado) ou média 8 (reprovado); ele avalia uma condição e decide o caminho a seguir. Da mesma forma, se precisarmos de calcular a média de notas de 100 estudantes, não seria viável escrever 100 linhas de código idênticas (recorremos, para isso, aos ciclos de repetição).

As estruturas de controlo dividem-se em duas categorias principais:

1. *Estruturas Condicionais (Selecção):* Permitem que o programa escolha qual bloco de código executar com base na veracidade de uma condição lógica.
2. *Estruturas de Repetição (Ciclos / Iteração):* Permitem executar um bloco de código várias vezes enquanto uma condição for verdadeira ou por um número definido de vezes.

2.2. Estruturas Condicionais (if, else, else if / switch)

A estrutura condicional mais elementar é o comando if (se). Ela avalia uma expressão booleana (verdadeiro ou falso). Se o resultado for verdadeiro, o bloco de código interno é executado; caso contrário, é ignorado. Quando precisamos de oferecer uma alternativa para o caso de a condição ser falsa, utilizamos a extensão else (senão) nesta condição são definidos dois caminhos mutuamente exclusivos: um executado quando a condição é verdadeira e outro quando é falsa. Para múltiplos cenários encadeados (como a atribuição de menções qualitativas a notas), recorremos a else if.

- *Exemplo contextualizado:* Verificar se um candidato a um concurso académico em Cuanza Sul tem idade igual ou superior a 18 anos para poder efectuar a inscrição de forma autónoma.

2.2.1. Metodologia de Rastreio Lógico (Teste de Mesa / Debug Manual)

Para evitar a sobrecarga gráfica de conectores complexos nos fluxogramas e desenvolver um pensamento computacional estruturado, utilizamos o *Teste de Mesa.* Esta técnica consiste em simular manualmente a execução do algoritmo passo a passo, registando numa tabela o valor actualizado de cada variável na memória RAM e a avaliação das expressões lógicas.

2.3. Exemplo Prático 1: Avaliação de Elegibilidade para Bolsa de Estudo no ISUP

Contexto: A direcção académica do ISUP pretende validar a concessão de uma bolsa de estudo por mérito. Um estudante é considerado elegível se possuir uma *média semestral >= 14.0 valores* e uma *assiduidade >= 80%.*

2.3.1 Algoritmo em Portugol (Pseudo-código)

```
algoritmo VerificarElegibilidadeBolsa;
var
   media: real; assiduidade: inteiro; elegivel: logico;
inicio
   escrever("Digite a média semestral do estudante:"); ler(media);
   escrever("Digite a percentagem de assiduidade (0-100):"); ler(assiduidade);
   se (media >= 14.0) e (assiduidade >= 80) entao
      elegivel <- verdadeiro;
      escrever("Resultado: Elegível para Bolsa de Mérito.");
   senao
      elegivel <- falso;
      escrever("Resultado: Não elegível para Bolsa de Mérito.");
   fim_se;
fim_algoritmo.
```

2.3.2. Tabela de Teste de Mesa (Debug Manual)

Na Tabela 3 demonstra-se a simulação passo a passo para um caso de teste com media = 15.5 e assiduidade = 75:

Tabela 3
Debug do Algoritmo

Passo	Linha de Código / Instrução	Variável media	Variável assiduidade	Expressão Lógica (media >= 14.0) e (assiduidade >= 80)	Variável elegível	Saída no Ecrã
1	Entrada de dados	15.5	75	*Não avaliada*	indefinido	—
2	Avaliação da condição	15.5	75	(15.5 >= 14.0) e (75 >= 80) -> FALSO	indefinido	—
3	Desvio para o bloco senao	15.5	75	FALSO	falso	—
4	Execução de Saída	15.5	75	FALSO	falso	"Resultado: Não elegível."

Análise Pedagógica do Debug: Observa que, embora a media (15.5) cumpra o requisito (>=14.0), o operador lógico E exige que ambas as condições sejam verdadeiras. Como a assiduidade (75) é inferior a 80, o resultado lógico global é FALSO, desviando o fluxo para o bloco senao.

2.3.3. Implementação Multi-Linguagem

C++

```
#include <iostream>
using namespace std;

int main() {
   double media;
   int assiduidade;
   bool elegivel;

   cout << "Digite a média semestral do estudante: ";
   cin >> media;
   cout << "Digite a percentagem de assiduidade (0-100): ";
   cin >> assiduidade;

   if (media >= 14.0 && assiduidade >= 80) {
      elegivel = true;
      cout << "Resultado: Elegível para Bolsa de Mérito." << endl;
   } else {
      elegivel = false;
      cout << "Resultado: Não elegível para Bolsa de Mérito." << endl;
   }

   return 0;
}
```

Python

```
media = float(input("Digite a média semestral do estudante: "))
assiduidade = int(input("Digite a percentagem de assiduidade (0-100): "))
if media >= 14.0 and assiduidade >= 80:
   elegivel = True
   print("Resultado: Elegível para Bolsa de Mérito.")
else:
   elegivel = False
   print("Resultado: Não elegível para Bolsa de Mérito.")
```

Java

```
import java.util.Scanner;

public class ElegibilidadeBolsa {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      System.out.print("Digite a média semestral do estudante: ");
      double media = input.nextDouble();

      System.out.print("Digite a percentagem de assiduidade (0-100): ");
```

```
        int assiduidade = input.nextInt();

        boolean elegivel;

        if (media >= 14.0 && assiduidade >= 80) {
            elegivel = true;
            System.out.println("Resultado: Elegível para Bolsa de Mérito.");
        } else {
            elegivel = false;
            System.out.println("Resultado: Não elegível para Bolsa de Mérito.");
        }

        input.close();
    }
}
```

2.4. Condicionantes Encadeadas e Aninhadas (Múltiplas Hipóteses)

Na prática da engenharia e da gestão académica, raras vezes os problemas resolvem-se com uma única condição binária (se / senão). Frequentemente, precisamos de avaliar uma cascata de condições exclusivas — por exemplo, para determinar menções qualitativas com base em intervalos de notas (Excelente, Bom, Suficiente, Insuficiente).

- *Condicionantes Encadeadas (se ... senao se ... senao):* Permitem testar sequencialmente múltiplos critérios até encontrar o primeiro ramo cuja expressão lógica seja verdadeira.

2.5. Exemplo Prático 2: Atribuição de Menção Qualitativa no ISUP

Contexto: O regulamento pedagógico do ISUP classifica a nota final de uma disciplina numa escala qualitativa:

- Nota >= 16: "Excelente"
- 14 >= Nota < 16: "Bom"
- 10 >= Nota} < 14: "Suficiente"
- Nota < 10: "Insuficiente"

2.5.1. Algoritmo em Portugol (Pseudo-código)

```
algoritmo AtribuirMencao;
var
   notaFinal: real;
inicio
   escrever("Digite a nota final do estudante (0 a 20):");    ler(notaFinal);
   se (notaFinal >= 16.0) entao escrever("Menção: Excelente");
   senao se (notaFinal >= 14.0) entao  escrever("Menção: Bom");
      senao se (notaFinal >= 10.0) entao escrever("Menção: Suficiente");
         senao escrever("Menção: Insuficiente"); fim_se;
      fim_se;
   fim_se; fim_algoritmo.
```

2.5.2. Tabela de Teste de Mesa (Debug Manual)

Na Tabela 4 está asimulação para um caso de teste com notaFinal = 12.5:

Tabela 4

Debug do Algoritmo

Passo	Linha / Condição Avaliada	notaFinal	Avaliação Lógica	Saída no Ecrã
1	Entrada de dados	12.5	—	—
2	1.ª Condição: notaFinal >= 16.0	12.5	12.5 >= 16.0 → FALSO	—
3	2.ª Condição: notaFinal >= 14.0	12.5	12.5 >= 14.0 →FALSO	—
4	3.ª Condição: notaFinal >= 10.0	12.5	12.5 >= 10.0 →VERDADEIRO	"Menção: Suficiente"

2.5.3 Implementação Multi-Linguagem

C++

```
#include <iostream>
using namespace std;
int main() {
   double notaFinal;
   cout << "Digite a nota final do estudante (0 a 20): ";
   cin >> notaFinal;
   if (notaFinal >= 16.0) { cout << "Menção: Excelente" << endl; }
   else if (notaFinal >= 14.0) { cout << "Menção: Bom" << endl; }
   else if (notaFinal >= 10.0) { cout << "Menção: Suficiente" << endl; }
   else { cout << "Menção: Insuficiente" << endl; }
   return 0; }
```

Python

```
nota_final = float(input("Digite a nota final do estudante (0 a 20): "))
if nota_final >= 16.0: print("Menção: Excelente")
elif nota_final >= 14.0: print("Menção: Bom")
elif nota_final >= 10.0: print("Menção: Suficiente")
else: print("Menção: Insuficiente")
```

Java

```
import java.util.Scanner;
public class AtribuirMencao {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Digite a nota final do estudante (0 a 20): ");
      double notaFinal = input.nextDouble();
      if (notaFinal >= 16.0) {  System.out.println("Menção: Excelente");}
       else if (notaFinal >= 14.0) {          System.out.println("Menção: Bom");}
       else if (notaFinal >= 10.0) {          System.out.println("Menção: Suficiente");}
       else {          System.out.println("Menção: Insuficiente");}
      input.close();}
}
```

2.6. Estrutura de Selecção Múltipla (switch / case)

Quando uma tarefa requer esforço repetitivo, utilizamos os laços de repetição:

- *Ciclo for (Para):* Utilizado primordialmente quando sabemos antecipadamente o número exacto de vezes que o bloco deve repetir-se (repetição definida). Possui uma variável de controlo, uma condição de paragem e um passo de incremento.
- *Ciclo while (Enquanto):* Utilizado quando o número de repetições é desconhecido à partida e depende de uma condição que é validada antes de cada execução (teste no início).
- *Ciclo do-while (Repetir... Enquanto):* Semelhante ao while, mas garante que o bloco de código executa pelo menos uma vez, pois a condição de teste é avaliada no final.

2.6.1 Exemplo Prático: Gestão de Departamentos no ISUP

- Opção 1: Departamento de Ciências Tecnológicas (DCT)
- Opção 2: Departamento de Ciências Económicas, Sociais e Humanas (DCESH)

- Opção 3: Departamento de Ciências da Saúde (DS)

Implementação em C++

```
#include <iostream>
using namespace std;
int main() {
   int opcao;
   cout << "Selecione o departamento (1- DCT, 2- DCESH), 3- DS): ";
   cin >> opcao;

   switch (opcao) {
      case 1:
         cout << "Direccionado ao Departamento de Ciências Tecnológicas." << endl;
         break;
      case 2:
         cout << "Direccionado ao Departamento de Ciências Económicas, Sociais e
Humanas." << endl;
         break;
      case 3:
         cout << " Departamento de Ciências da Saúde." << endl;
         break;
      default:
         cout << "Opção inválida!" << endl;
         break;
   }
   return 0;
}
```

2.6.2 Trabalho Autónomo

Implemente o mesmo exercício em Python e Java

2.7 Estruturas Repetitivas e Rastreio de Iterações (*Testes de Mesa*)

Muitas vezes, um algoritmo precisa de executar o mesmo bloco de instruções várias vezes (por exemplo, processar as notas de todos os estudantes de uma turma do ISUP, contar livros numa biblioteca ou calcular somas acumuladas).

As estruturas repetitivas evitam a duplicação massiva de código e dividem-se essencialmente em duas categorias principais:

1. *Ciclo com Condição no Início (enquanto / while):* Avalia a condição *antes* de cada execução do bloco. Se a condição for falsa logo no início, o bloco pode nem sequer ser executado.
2. *Ciclo com Contador Controlado (para / for):* Utilizado quando sabemos previamente o número exacto de vezes que o bloco deve repetir (geralmente controlado por uma variável de contagem).

2.8. Exemplo Prático 3: Contagem e Soma Acumulada de Livros na Biblioteca

Contexto: Um sistema de registo no ISUP precisa de simular a contagem de n livros que entram no sistema e calcular o identificador acumulado da sequência. Vamos analisar o comportamento do ciclo enquanto (while).

2.8.1. Algoritmo em Portugol (Pseudo-código)

```
algoritmo SomarContagemLivros;
var
   n, contador, soma: inteiro;
inicio
   escrever("Digite o número total de livros (n):");
   ler(n);

   contador <- 1;
   soma <- 0;

   enquanto (contador <= n) faca
      soma <- soma + contador;
      contador <- contador + 1;
   fim_enquanto;

   escrever("Soma total dos identificadores: ", soma);
fim_algoritmo.
```

2.8.2. Tabela de Teste de Mesa (Debug Manual)

Na Tabela 5 está simulada a execução passo a passo para um caso de teste em que o utilizador insere n = 3:

Tabela 5

Debug do Algoritmo

Iteração / Passo	Instrução Executada	n	contador	soma	Condição contador <= n	Saída / Acção no Ecrã
1	Inicialização	3	1	0	1 <= 3 →VERDADEIRO	—
2	1.ª Iteração (Corpo)	3	1	soma = 0 + 1 → 1	—	contador passa a 2
3	Início do Ciclo	3	2	1	2 <= 3 →VERDADEIRO	—
4	2.ª Iteração (Corpo)	3	2	soma = 1 + 2 →3	—	contador passa a 3
5	Início do Ciclo	3	3	3	3 <= 3 →VERDADEIRO	—
6	3.ª Iteração (Corpo)	3	3	soma = 3 + 3 →6	—	contador passa a 4
7	Início do Ciclo	3	4	6	4 <= 3 →FALSO	O ciclo termina
8	Instrução Final	3	4	6	—	"Soma total dos identificadores: 6"

*Análise Pedagógica do Debug***:** Agora você consegue rastrear perfeitamente como a variável soma acumula os valores de forma dinâmica e como o contador actua como a guarda de segurança que impede um ciclo infinito.

2.8.3. Implementação Multi-Linguagem

C++

```
#include <iostream>
using namespace std;
int main() {
   int n, contador = 1, soma = 0;
   cout << "Digite o número total de livros (n): ";
   cin >> n;
   while (contador <= n) {
      soma += contador;   // Equivalente a soma = soma + contador
      contador++;        // Incrementa o contador
   }
   cout << "Soma total dos identificadores: " << soma << endl;
   return 0;
}
```

Python

```
n = int(input("Digite o número total de livros (n): "))
contador = 1
soma = 0
while contador <= n:
   soma += contador
   contador += 1
print(f"Soma total dos identificadores: {soma}")
```

Java

```
import java.util.Scanner;
public class SomarContagemLivros {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Digite o número total de livros (n): ");
      int n = input.nextInt();
      int contador = 1;
      int soma = 0;
      while (contador <= n) {
         soma += contador;
         contador++;
      }
      System.out.println("Soma total dos identificadores: " + soma);
      input.close();
   }
}
```

2.9 Secção de Exercícios Resolvidos (Taxonomia Multi-linguagem)

Problema 1 (Resolvido): Verificação de Número Par ou Ímpar

- **Contexto:** Desenvolver um programa que receba um número inteiro introduzido por um utilizador e determine se este é par ou ímpar, utilizando o operador de módulo (%).

- **Entrada:** Um número inteiro.
- **Saída:** A mensagem indicando "Par" ou "Ímpar".

Código C++

```
#include <iostream>
using namespace std;

int main() {
   int numero;
   cout << "Digite um número inteiro: ";
   cin >> numero;

   if (numero % 2 == 0) {
      cout << "O número é Par." << endl;
   } else {
      cout << "O número é Ímpar." << endl;
   }
   return 0;
}
```

Código Python

```
numero = int(input("Digite um número inteiro: "))

if numero % 2 == 0:
   print("O número é Par.")
else:
   print("O número é Ímpar.")
```

Código Java

```
import java.util.Scanner;

public class ParImpar {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite um número inteiro: ");
      int numero = sc.nextInt();

      if (numero % 2 == 0) {
         System.out.println("O número é Par.");
      } else {
         System.out.println("O número é Ímpar.");
      }
      sc.close();
   }
}
```

Problema 2 (Resolvido): Contagem Regressiva para Temporizador

- **Contexto:** Um laboratório de robótica no ISUP precisa de um temporizador automático que exiba uma contagem regressiva de 5 até 1 antes de disparar um processo.
- **Entrada:** Nenhuma (sequência fixa de contagem).
- **Saída:** Os números de 5 a 1 impressos sequencialmente.

Código C++

```
#include <iostream>
using namespace std;

int main() {
   cout << "Iniciando contagem regressiva..." << endl;
   for (int i = 5; i >= 1; i--) {
      cout << i << endl;
   }
   cout << "Sistema disparado!" << endl;
   return 0;
}
```

Código Python

```
Python
print("Iniciando contagem regressiva...")
for i in range(5, 0, -1):
   print(i)
print("Sistema disparado!")
```

Código Java

```
Java
public class ContagemRegressiva {
   public static void main(String[] args) {
      System.out.println("Iniciando contagem regressiva...");
      for (int i = 5; i >= 1; i--) {
         System.out.println(i);
      }
      System.out.println("Sistema disparado!");
   }
}
```

Problema 3 (Resolvido): Classificação de Desempenho Académico por Escalões

- **Contexto:** Atribuir uma menção qualitativa com base na nota numérica final (0 a 20) obtida por um estudante do ISUP.
- **Entrada:** Nota numérica real.
- **Saída:** Menção ("Mau", "Suficiente", "Bom", "Muito Bom", "Excelente").

Código C++

```
#include <iostream>
using namespace std;

int main() {
   double nota;
   cout << "Digite a nota do estudante (0 a 20): ";
   cin >> nota;

   if (nota < 10) {
      cout << "Menção: Reprovado / Insuficiente" << endl;
   } else if (nota >= 10 && nota < 13) {
      cout << "Menção: Suficiente" << endl;
   } else if (nota >= 13 && nota < 16) {
      cout << "Menção: Bom" << endl;
   } else if (nota >= 16 && nota < 19) {
      cout << "Menção: Muito Bom" << endl;
```

```
    } else {
        cout << "Menção: Excelente" << endl;
    }
    return 0;
}
```

Código Python

```
nota = float(input("Digite a nota do estudante (0 a 20): "))

if nota < 10:
    print("Menção: Reprovado / Insuficiente")
elif 10 <= nota < 13:
    print("Menção: Suficiente")
elif 13 <= nota < 16:
    print("Menção: Bom")
elif 16 <= nota < 19:
    print("Menção: Muito Bom")
else:
    print("Menção: Excelente")
```

Código Java

```
import java.util.Scanner;

public class ClassificacaoNota {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Digite a nota do estudante (0 a 20): ");
        double nota = sc.nextDouble();

        if (nota < 10) {
            System.out.println("Menção: Reprovado / Insuficiente");
        } else if (nota >= 10 && nota < 13) {
            System.out.println("Menção: Suficiente");
        } else if (nota >= 13 && nota < 16) {
            System.out.println("Menção: Bom");
        } else if (nota >= 16 && nota < 19) {
            System.out.println("Menção: Muito Bom");
        } else {
            System.out.println("Menção: Excelente");
        }
        sc.close();
    }
}
```

Problema 4 (Resolvido): Validação de Palavra-Passe de Acesso

- **Contexto:** Um terminal de auto-atendimento exige que o utilizador digite uma palavra-passe numérica correcta (1234) para aceder ao menu principal, repetindo a solicitação enquanto o valor inserido estiver errado.
- **Entrada:** Tentativas sucessivas de código PIN.
- **Saída:** Mensagem de acesso concedido ou erro.

Código C++

```
#include <iostream>
using namespace std;

int main() {
    int senha = 0;
```

```
    const int SENHA_CORRETA = 1234;

    while (senha != SENHA_CORRETA) {
        cout << "Digite a palavra-passe de acesso: ";
        cin >> senha;
        if (senha != SENHA_CORRETA) {
            cout << "Palavra-passe incorreta. Tente novamente." << endl;
        }
    }
    cout << "Acesso concedido com sucesso!" << endl;
    return 0;
}
```

Código Python

```
senha = 0
SENHA_CORRETA = 1234

while senha != SENHA_CORRETA:
    senha = int(input("Digite a palavra-passe de acesso: "))
    if senha != SENHA_CORRETA:
        print("Palavra-passe incorreta. Tente novamente.")

print("Acesso concedido com sucesso!")
```

Código Java

```
import java.util.Scanner;

public class ValidarSenha {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int senha = 0;
        final int SENHA_CORRETA = 1234;

        while (senha != SENHA_CORRETA) {
            System.out.print("Digite a palavra-passe de acesso: ");
            senha = sc.nextInt();
            if (senha != SENHA_CORRETA) {
                System.out.println("Palavra-passe incorreta. Tente novamente.");
            }
        }
        System.out.println("Acesso concedido com sucesso!");
        sc.close();
    }
}
```

Problema 5 (Resolvido): Gerador de Tabuada de Multiplicação

- **Contexto:** Um estudante do 1.º ano precisa de visualizar a tabuada completa de multiplicação de um número inteiro fornecido (de 1 a 10).
- **Entrada:** Um número inteiro base.
- **Saída:** A tabuada formatada de 1 a 10.

Código C++

```
#include <iostream>
```

```
using namespace std;

int main() {
   int n;
   cout << "Digite o número para ver a tabuada: ";
   cin >> n;

   for (int i = 1; i <= 10; i++) {
      cout << n << " x " << i << " = " << (n * i) << endl;
   }
   return 0;
}
```

Código Python

```
n = int(input("Digite o número para ver a tabuada: "))

for i in range(1, 11):
   print(f"{n} x {i} = {n * i}")
```

Código Java

```
import java.util.Scanner;

public class Tabuada {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Digite o número para ver a tabuada: ");
      int n = sc.nextInt();

      for (int i = 1; i <= 10; i++) {
         System.out.println(n + " x " + i + " = " + (n * i));
      }
      sc.close();
   }
}
```

2.10. Secção de 5 Exercícios Propostos (Trabalho Autónomo)

Problema 1 (Somar Números até Zero):

- **Contexto:** Escreva um programa que solicite continuamente números inteiros ao utilizador e some todos eles. O programa deve parar e exibir a soma total acumulada assim que o utilizador digitar o número 0.
- **Linguagens requeridas:** C++, Python e Java.

Problema 2 (Maior entre Três Números):

- **Contexto:** Desenvolva um algoritmo que receba três números inteiros distintos e utilize estruturas condicionais compostas para determinar e exibir qual deles é o maior valor.
- **Linguagens requeridas:** C++, Python e Java.

Problema 3 (Calculadora de Tarifas de Electricidade):

- **Contexto:** Uma empresa local calcula a factura de luz com base no consumo em kWh. Se o consumo for até 100 kWh, o preço por kWh é 50

Kz; se ultrapassar 100 kWh, o excedente é cobrado a 85 Kz por kWh. Crie um programa que calcule o valor total a pagar.

- **Linguagens requeridas:** C++, Python e Java.

Problema 4 (Soma dos Números Pares num Intervalo):

- **Contexto:** Crie um programa que utilize um ciclo de repetição para somar todos os números pares compreendidos entre 1 e 100 (inclusive), exibindo o resultado final no ecrã.
- **Linguagens requeridas:** C++, Python e Java.

Problema 5 (Simulador de Caixa Automático - ATM Simplificado):

- **Contexto:** Um sistema bancário elementar apresenta um menu com opções: 1. Consultar Saldo, 2. Depositar Dinheiro, 3. Levantar Dinheiro, 4. Sair. Utilize uma estrutura de selecção múltipla (switch ou if-else encadeado) dentro de um ciclo para manter o menu activo até que o utilizador escolha sair.
- **Linguagens requeridas:** C++, Python e Java.

Problema 6: Validação de Notas de Turma com Média Dinâmica

- *Contexto:* Um docente de Engenharia Informática no ISUP precisa de processar as notas de uma turma. O programa deve solicitar repetidamente a nota de um estudante (0 a 20) até que o utilizador insira um valor sentinela negativo (ex: -1) para encerrar a introdução de dados. No final, o sistema deve apresentar o número total de estudantes avaliados, a média aritmética da turma e a percentagem de estudantes aprovados (nota>=10).
- *Linguagens Requeridas*: C++, Python e Java.
- *Orientação de Teste de Mesa:* Construa uma tabela acompanhando as variáveis contador, somaNotas, notaAtual e aprovados a cada iteração de entrada de dados.

Problema 7: Simulação de Pagamento de Propinas com Multa por Atraso

- *Contexto:* O departamento financeiro do ISUP aplica uma política de acréscimo automático nas propinas pagas fora do prazo. Se o pagamento for efectuado após a data limite, aplica-se uma penalização de 5% por cada mês de atraso (acumulável de forma linear sobre o valor base). O algoritmo deve ler o valor base da propina e o número de meses em atraso, calcular o valor final a pagar utilizando uma estrutura condicional para verificar se há atraso, e exibir o extracto final.
- *Linguagens Requeridas:* C++, Python e Java.

Problema 8: Controlo de Acesso ao Laboratório de Redes

- *Contexto:* O acesso ao laboratório informático do ISUP é restrito a estudantes que possuam o cartão de acesso activo e pertençam a um dos três anos curriculares do curso. O programa deve validar interactivamente o estado do cartão (activo = verdadeiro/falso) e o ano curricular (1, 2 ou 3). Caso ambas as condições sejam verdadeiras, autoriza a entrada; caso contrário, exibe o motivo específico da recusa através de condicionais aninhadas.
- *Linguagens Requeridas:* C++, Python e Java.

2.10. Perguntas de Reflexão e Consolidação Pedagógica (Módulo II)

1. *Gestão de Ciclos:* Qual é a principal diferença de comportamento na memória RAM entre um ciclo enquanto (while) cuja condição é falsa logo na primeira avaliação e um ciclo fazer-enquanto (do-while)?
2. *Depuração Lógica:* Explique por que motivo a realização de um *teste de mesa* (tabela de rastreio) antes de transcrever o código para C++ ou Java reduz drasticamente a ocorrência de erros lógicos (como ciclos infinitos).
3. *Eficiência de Selecção:* Em que circunstâncias práticas a estrutura de selecção múltipla (switch / case) se torna preferível face ao uso sucessivo de condicionais encadeadas (if... else if... else)?

2.11. Perguntas de Reflexão e Consolidação Pedagógica

1. Qual é a principal diferença prática na escolha entre um ciclo controlado por contador (for) e um ciclo controlado por condição lógica (while)?
2. O que sucede num programa se a condição de paragem de um ciclo while nunca se tornar falsa (ausência de actualização da variável de controlo)? Como se denomina este fenómeno técnico?
3. Em que circunstâncias estruturais é preferível utilizar a instrução de selecção múltipla (switch) em detrimento de múltiplos blocos else if encadeados?