Lógica e Linguagem de Programação

Fluxograma, tabela verdade e teste de mesa

Prof. Flávio Murilo de Carvalho Leal
Instituto Centro de Ensino Tecnológico
Faculdade de Tecnologia - FATEC Cariri

- **Fluxograma:** representação gráfica da lógica de um algoritmo.

- **Fluxograma:** representação gráfica da lógica de um algoritmo.
- **Tabela verdade:** todos os resultados possíveis de uma expressão lógica.

- **Fluxograma:** representação gráfica da lógica de um algoritmo.
- **Tabela verdade:** todos os resultados possíveis de uma expressão lógica.
- **Teste de mesa:** simulação manual da execução de um algoritmo.

- **Fluxograma:** representação gráfica da lógica de um algoritmo.
- **Tabela verdade:** todos os resultados possíveis de uma expressão lógica.
- **Teste de mesa:** simulação manual da execução de um algoritmo.

Três ferramentas para planejar, verificar e depurar algoritmos antes de programar.

O que é um fluxograma?

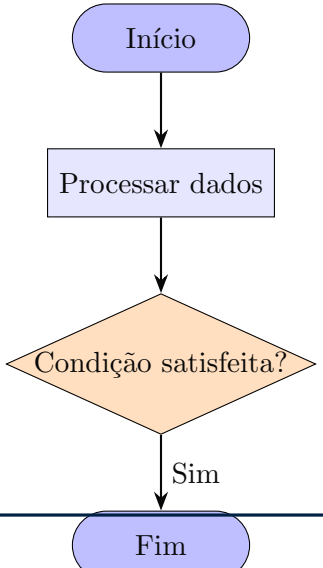
- **Definição:** representação gráfica de um algoritmo, em que cada etapa do processo é desenhada por meio de símbolos padronizados, ligados por setas que indicam a ordem de execução.

O que é um fluxograma?

- **Definição:** representação gráfica de um algoritmo, em que cada etapa do processo é desenhada por meio de símbolos padronizados, ligados por setas que indicam a ordem de execução.
- Facilita o entendimento da lógica antes de escrever código.
- Evidencia decisões, repetições e o fluxo entre etapas.
- É independente de linguagem de programação.

Símbolo	**Significado**
Oval (terminal)	Início ou fim do algoritmo
Retângulo	Processo – uma ação ou cálculo
Losango	Decisão – teste lógico com duas ou mais saídas
Paralelogramo	Entrada ou saída de dados
Seta	Indica o sentido do fluxo

Estrutura básica de um fluxograma



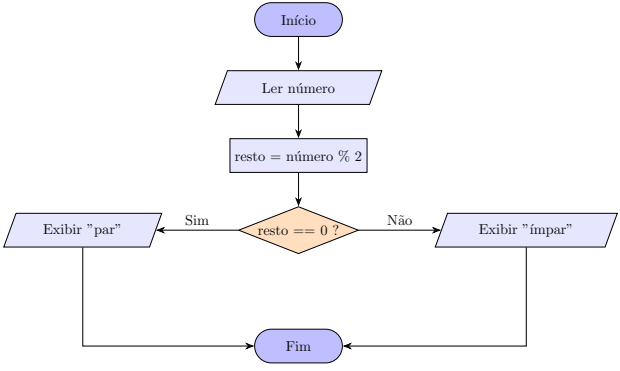
Exemplo: número par ou ímpar?

Algoritmo: lê um número inteiro e informa se ele é par ou ímpar, usando o resto da divisão por 2 (operador %).

Exemplo: número par ou ímpar?

Algoritmo: lê um número inteiro e informa se ele é par ou ímpar, usando o resto da divisão por 2 (operador %).

```
1 int numero, resto;
2 numero = ler_entrada();
3 resto = numero % 2;
4 se (resto == 0)
5     escrever("par");
6 senao
7     escrever("impar");
```



- **Um único início e um único fim:** evite múltiplos pontos de entrada.

- **Um único início e um único fim:** evite múltiplos pontos de entrada.
- **Setas sempre com direção:** toda ligação deve indicar o sentido do fluxo.

- **Um único início e um único fim:** evite múltiplos pontos de entrada.
- **Setas sempre com direção:** toda ligação deve indicar o sentido do fluxo.
- **Um símbolo por ação:** cada retângulo ou losango representa uma única ação ou decisão simples.

- **Um único início e um único fim:** evite múltiplos pontos de entrada.
- **Setas sempre com direção:** toda ligação deve indicar o sentido do fluxo.
- **Um símbolo por ação:** cada retângulo ou losango representa uma única ação ou decisão simples.
- **Decisões com saídas nomeadas:** rotule os ramos do losango (ex.: Sim/Não).

O que é uma tabela verdade?

- **Definição:** tabela que lista todas as combinações possíveis de valores lógicos (verdadeiro/falso) das variáveis de entrada e o resultado correspondente de uma expressão lógica.

O que é uma tabela verdade?

- **Definição:** tabela que lista todas as combinações possíveis de valores lógicos (verdadeiro/falso) das variáveis de entrada e o resultado correspondente de uma expressão lógica.
- Em C, verdadeiro = 1 (ou diferente de zero) e falso = 0.
- Com n variáveis de entrada, existem 2^n combinações possíveis.
- Usada para prever o resultado de condições em *ifs* e laços.

O que é uma tabela verdade?

- **Definição:** tabela que lista todas as combinações possíveis de valores lógicos (verdadeiro/falso) das variáveis de entrada e o resultado correspondente de uma expressão lógica.
- Em C, verdadeiro = 1 (ou diferente de zero) e falso = 0.
- Com n variáveis de entrada, existem 2^n combinações possíveis.
- Usada para prever o resultado de condições em *ifs* e laços.

A	**!A**
0	1
1	0

Operadores lógicos AND e OR

A && B (E lógico)

A	B	A&&B
0	0	0
0	1	0
1	0	0
1	1	1

`A || B` (OU lógico)

A	B	A‖B
0	0	0
0	1	1
1	0	1
1	1	1

Operadores lógicos AND e OR

A && B (E lógico)

A	B	A&&B
0	0	0
0	1	0
1	0	0
1	1	1

`A || B` (OU lógico)

A	B	A‖B
0	0	0
0	1	1
1	0	1
1	1	1

`&&` só é verdadeiro quando as duas condições são verdadeiras. `||` é verdadeiro quando ao menos uma delas é verdadeira.

Expressão: `(A && B) || !C`

Com 3 variáveis, temos $2^3 = 8$ combinações possíveis.

A	**B**	**C**	**A&&B**	**!C**	`(A&&B)||!C`
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

- **Definição:** simulação manual da execução de um algoritmo – seguir as instruções, uma a uma, como se fosse o computador, anotando como cada variável muda a cada passo.

O que é teste de mesa?

- **Definição:** simulação manual da execução de um algoritmo – seguir as instruções, uma a uma, como se fosse o computador, anotando como cada variável muda a cada passo.
- Encontra erros de lógica antes de compilar o código.
- Ajuda a entender o comportamento de laços e condições.
- Não depende de computador – só de papel e atenção.

1. **Monte a tabela:** uma coluna para cada variável, mais uma para a linha/instrução atual.

1. **Monte a tabela:** uma coluna para cada variável, mais uma para a linha/instrução atual.
2. **Inicialize as variáveis:** anote os valores iniciais antes do primeiro comando ser executado.

1. **Monte a tabela:** uma coluna para cada variável, mais uma para a linha/instrução atual.
2. **Inicialize as variáveis:** anote os valores iniciais antes do primeiro comando ser executado.
3. **Execute mentalmente:** siga o algoritmo instrução por instrução, atualizando os valores a cada mudança.

1. **Monte a tabela:** uma coluna para cada variável, mais uma para a linha/instrução atual.
2. **Inicialize as variáveis:** anote os valores iniciais antes do primeiro comando ser executado.
3. **Execute mentalmente:** siga o algoritmo instrução por instrução, atualizando os valores a cada mudança.
4. **Registre cada iteração:** em laços, crie uma nova linha a cada repetição do bloco.

1. **Monte a tabela:** uma coluna para cada variável, mais uma para a linha/instrução atual.
2. **Inicialize as variáveis:** anote os valores iniciais antes do primeiro comando ser executado.
3. **Execute mentalmente:** siga o algoritmo instrução por instrução, atualizando os valores a cada mudança.
4. **Registre cada iteração:** em laços, crie uma nova linha a cada repetição do bloco.
5. **Compare com o esperado:** confira se o resultado bate com o que o algoritmo deveria produzir.

Exemplo prático

```
1 soma = 0;
2 i = 1;
3 enquanto (i <= 3) {
4       soma = soma + i;
5       i = i + 1;
6 }
7 escrever(soma);
```

Laço que soma os números de 1 a 3.

Iteração	**i**	$i \leq 3$**?**	**soma**
Início	1	—	0
1ª	1	V	1
2ª	2	V	3
3ª	3	V	6
4ª (fim)	4	F	6

Exemplo prático

```
1 soma = 0;
2 i = 1;
3 enquanto (i <= 3) {
4       soma = soma + i;
5       i = i + 1;
6 }
7 escrever(soma);
```

Laço que soma os números de 1 a 3.

Iteração	**i**	$i \leq 3$**?**	**soma**
Início	1	—	0
1ª	1	V	1
2ª	2	V	3
3ª	3	V	6
4ª (fim)	4	F	6

Resultado final: soma = 6