

# Algoritmos e Programação I

Estruturação de algoritmos





01

# Conceitos Básicos

# Definição

O que é?

Sequências finitas de instruções, utilizadas a fim de resolver um problema

**Entrada**

Dados fornecidos pelo usuário (exemplo: aquilo que é digitado ou selecionado pelo mouse);

**Processamento**

Passo a passo para resolver um problema;

**Saída**

Dados já processados, problema resolvido



01

02

03

04

05

06



# Importância da clareza e organização de algoritmos


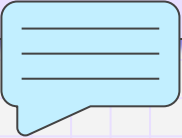


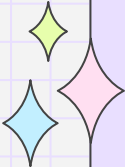
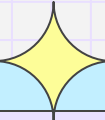
Manutenção e melhorias possam ocorrer mais facilmente.

Código limpo e bem endentado

Síntese clara e intuitiva

Reduzir ao máximo a complexidade

Trabalho em equipe torna-se muito mais estressante e difícil



02

# Pseudocódigos

# Tipos de dados e variáveis



**Inteiro**

**Pseudocódigo:**  
var nome: inteiro

**Em C:**  
int nome;



**Ponto Flutuante**

**Pseudocódigo:**  
var nome: real

**Em C:**  
float nome;



**Texto**

**Pseudocódigo:**  
var nome: caractere

**Em C:**  
char nome;



**Lógico**

**Pseudocódigo:**  
var nome: lógico

**Em C:**  
bool nome;



01

02

03

04

05

06

# Exemplos Pseudocódigo

## Declaração de variáveis

```
var nomeAluno: caractere  
var idadeAluno: inteiro  
var alturaAluno: real  
var alunoAprovado: logico
```

## Atribuição de valores

```
nomeAluno <- "Lucas"  
idadeAluno <- 20  
alturaAluno <- 1.80  
alunoAprovado <- Verdadeiro
```



01

02

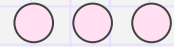
03

04

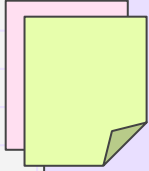
05

06





# Operadores Lógicos



Negação

!

E

&&

Ou

||



01

02

03

04

05

06





# Operador Relacional

Igual

==

Diferente

!=

Menor que

<

Maior que

>

Maior ou igual  
que

>=

Menor ou igual  
que

<=



01

02

03

04

05

06



# Operador Aritméticos

Soma

+

Subtração

-

Divisão

/

Multiplicação

\*

Módulo

%

Incremento

++

Decremento

--



01

02

03

04

05

06



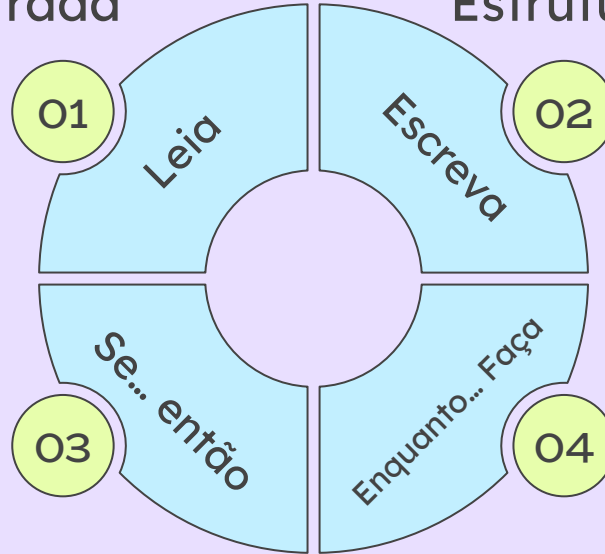
# Estruturas importantes para Pseudocódigos

Estrutura de Entrada

Estrutura de Saída

Estrutura de Decisão

Estrutura de Repetição



01

02

03

04

05

06



# Exemplos Pseudocódigo

## Estrutura de Entrada

```
Leia (turmaAluno)
Leia (idadeAluno)
Leia (alturaAluno)
Leia (alunoAprovado)
```

## Estrutura de Saída

```
Escreva ("Olá, mundo!")
Escreva (turmaAluno)
Escreva (idadeAluno)
Escreva (alturaAluno)
Escreva (alunoAprovado)
```



01

02

03

04

05

06



# Exemplos em Pseudocódigo

## Estrutura de Decisão

```
Se (condição) então  
    <comandos>  
Se não <comando>
```

## Estrutura de Repetição

```
Enquanto (condição) faça  
    <comandos>  
  
Para (condição) faça  
    <comandos>  
  
Faça  
    <comandos>  
Enquanto (condição)
```



01

02

03

04

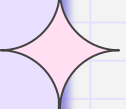
05

06





# Passo a passo



01

02

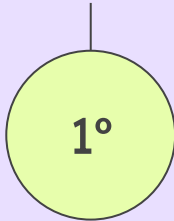
03

04

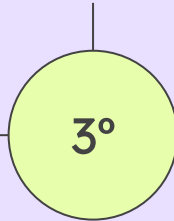
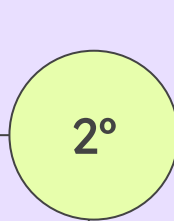
05

06

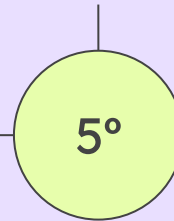
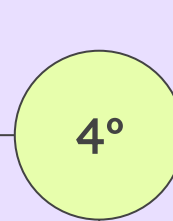
Declarar Variáveis



Estrutura de Repetição e  
Decisão, caso seja  
necessário para a lógica

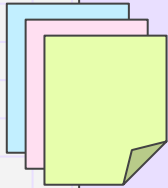


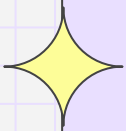
Programa pronto



Utilizar estrutura de entrada  
para coletar dados ou atribuir  
valores para as variáveis

Utilizar estrutura de  
saída para retorno de  
dados





```
//Declaração de variáveis  
var turmaAluno: caractere  
var alunoAprovado: logico
```

```
//Entrada de dados  
Leia (turmaAluno) ou nomeAluno <- "A"  
Leia (alunoAprovado) ou alunoAprovado <- verdade
```

```
//Estrutura de decisão  
Se (alunoAprovado == verdade) então  
    Escreva ("Aluno da turma " + turmaAluno + "foi aprovado!")  
se não Escreva ("Aluno da turma " + turmaAluno + "foi reprovado!")
```



01

02

03

04

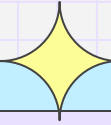
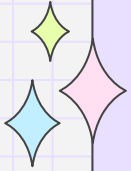
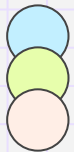
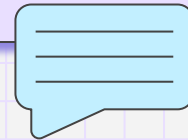
05

06



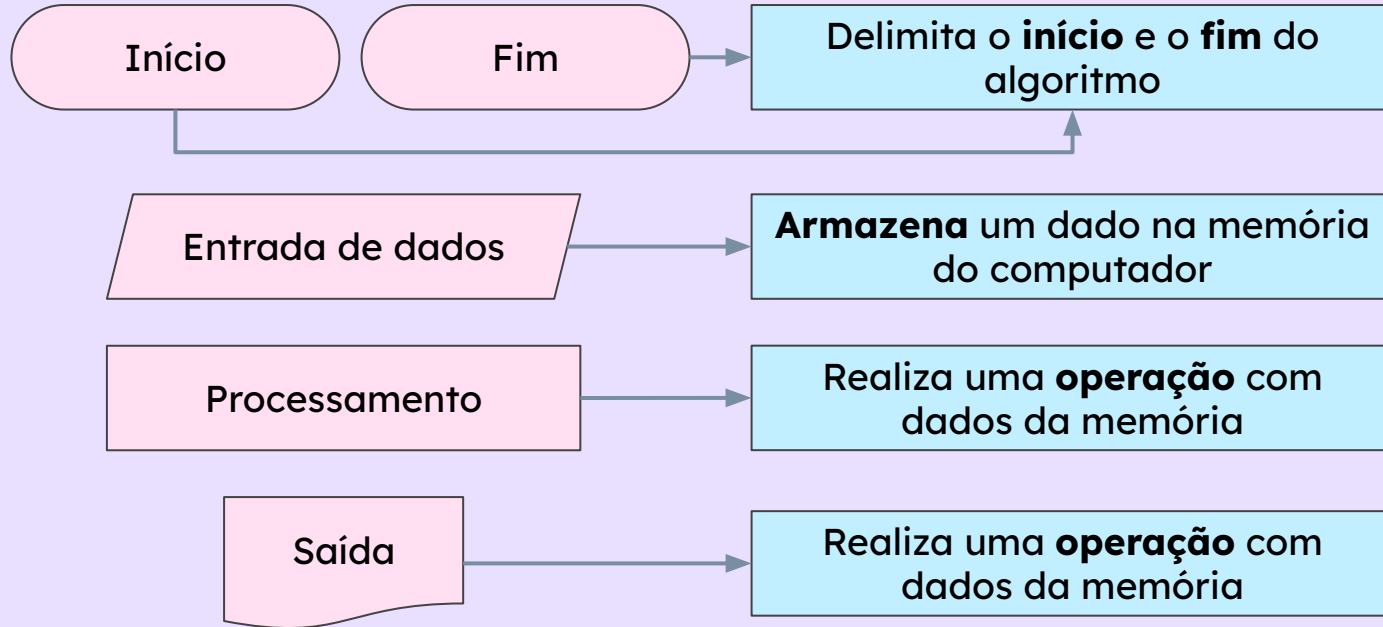
03

# Fluxogramas





# Símbolos Gráficos



01

02

03

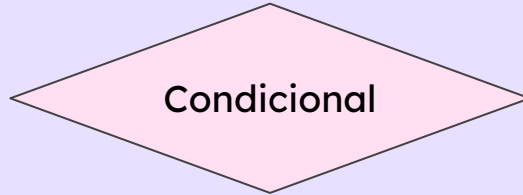
04

05

06



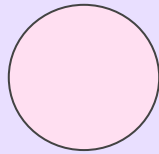
# Símbolos Gráficos



Condicional



Cria uma condição



Estrutura de repetição



01

02

03

04

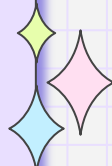
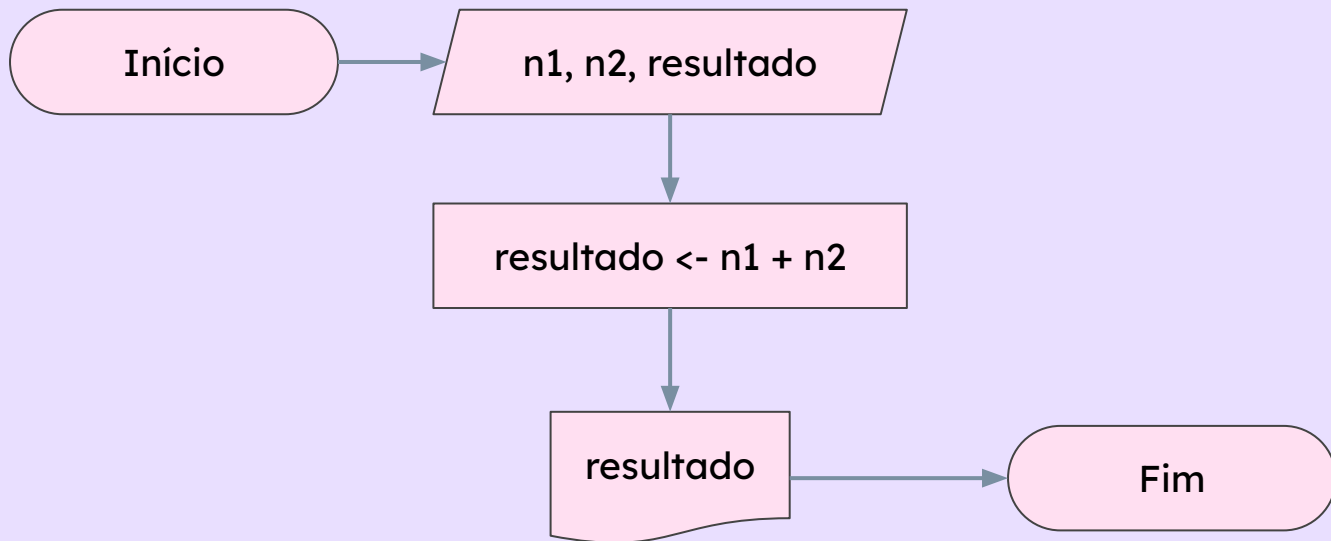
05

06





## Exemplo



01

02

03

04

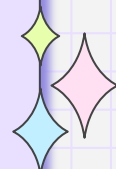
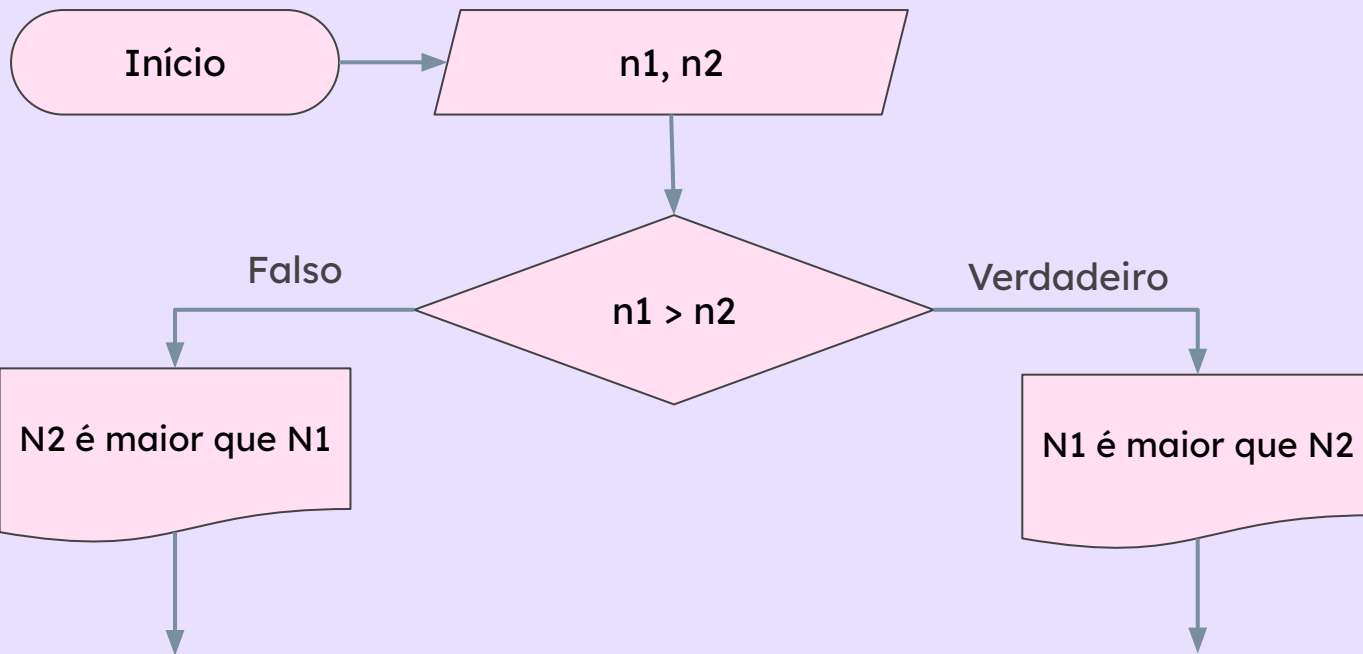
05

06





## Exemplo condicional



01

02

03

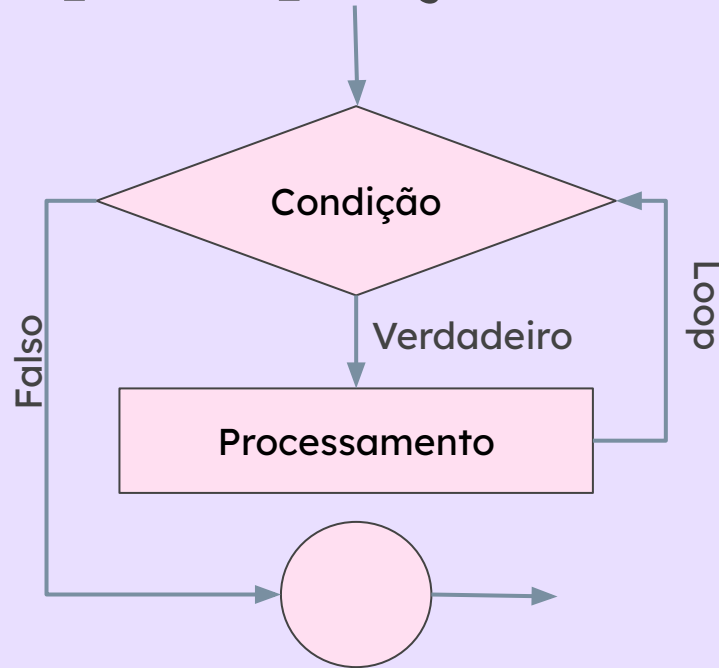
04

05

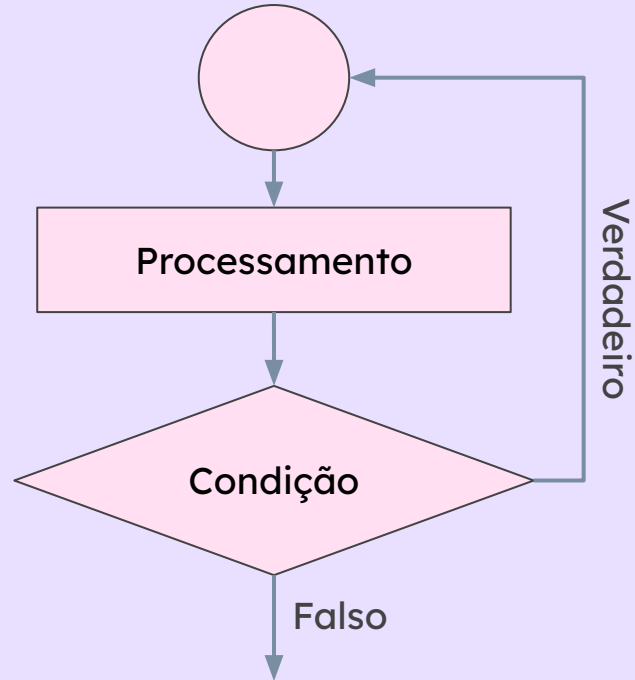
06



# Exemplo Repetição (While)



# Exemplo Repetição (Do, While)



01

02

03

04

05

06

# Exemplo Repetição (For)

Inicia a expressão

Condição

Verdadeiro

Falso

Processamento

Atualiza expressão

Loop



01

02

03

04

05

06



04

Códigos em C



# Exemplos em C

## Declaração de variáveis

```
char nomeAluno[30];  
int idadeAluno;  
float alturaAluno;  
bool alunoAprovado;
```

## Atribuição de valores

```
turmaAluno = 'A';  
idadeAluno = 20;  
alturaAluno = 1.80;  
alunoAprovado = Verdadeiro;
```



01

02

03

04

05

06



# Exemplos em C

## Estrutura de Entrada

```
scanf("%c", &turmaAluno);  
scanf("%d", &idadeAluno);  
scanf("%f", &alturaAluno);
```

## Estrutura de Saída

```
printf("Olá, mundo!");  
printf("%c", turmaAluno);  
printf("%s", idadeAluno);  
printf("%f", alturaAluno);  
printf(alunoAprovado);
```



01

02

03

04

05

06



# Exemplos em C

## Estrutura de Decisão

```
If (mediaAluno < 7){  
    printf("Aluno reprovado");  
}else{  
    printf("Aluno aprovado");  
}
```

## Estrutura de Repetição

```
while (i < 200){  
    i++;  
}  
  
for(i = 0; i < 10; i++){  
    print("%d", i);  
}  
  
do{  
    scanf("%d", &i);  
}while (i != 0);
```



01

02

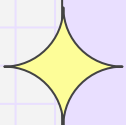
03

04

05

06





```
//Declaração de variáveis
char turmaAluno;
bool alunoAprovado;

//Entrada de dados
scanf("%c", &turmaAluno);
alunoAprovado = true;

//Estrutura de decisão
if (alunoAprovado == true)
{
    printf("Aluno da turma %c foi aprovado!", turmaAluno);
} else printf("Aluno da turma %c foi reprovado!", turmaAluno);
```



01

02

03

04

05

06

