

# **Programação Procedural e Estrutura de Dados**

Conceitos básicos

# Algoritmo

- O que é Algoritmo?
  - Sequência de passos para resolver um problema
    - Sequência de instruções ou operações
  - Parte de um estado inicial
    - Dados de entrada
  - Executa tarefas bem definidas num tempo finito
    - Processamento
  - Atinge um objetivo
    - Dados de saída

# Algoritmo

- Algoritmos no dia a dia:

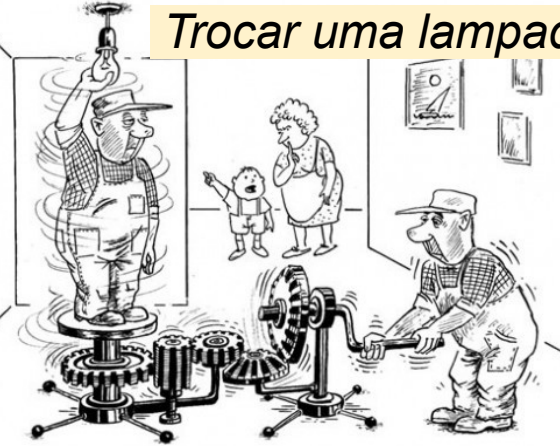
Ir para a faculdade



Somar dois números

$$\begin{array}{r} 3,5 \\ + 0,5 \\ \hline 4,0 \end{array}$$

Trocar uma lampada



Fazer um sanduíche



Sacar dinheiro

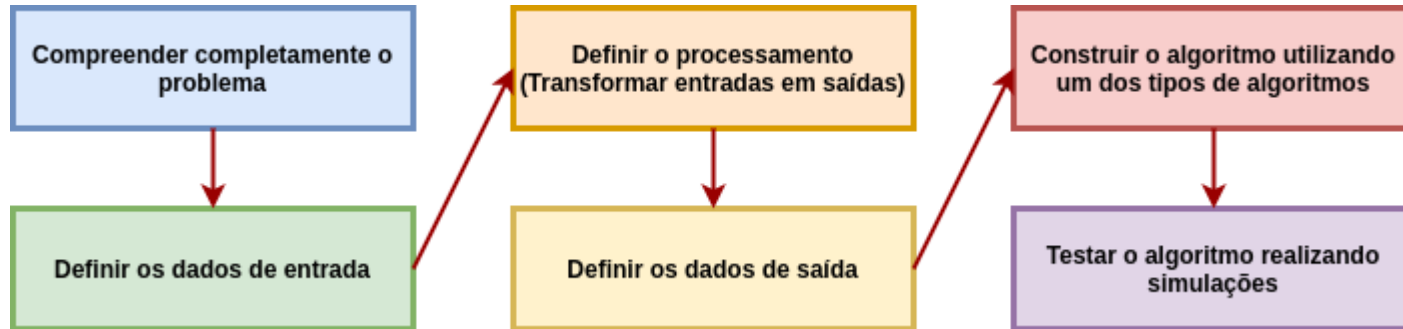


Abrir uma lata de refrigerante



# Algoritmo

- Como construir um algoritmo?



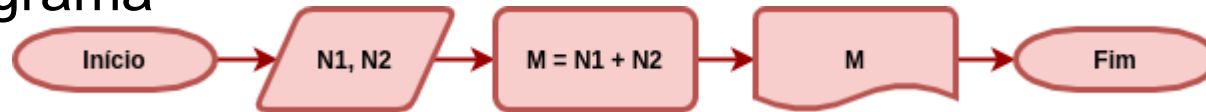
# Algoritmo

- Alguns tipos de algoritmos

- Descrição narrativa

1. Receber dois números que serão somados
2. Somar os dois números
3. Mostrar o resultado obtido na soma

- Fluxograma



- Pseudocódigo ou Portugal

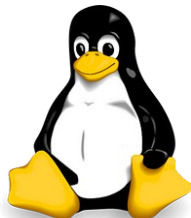
```
INICIO_ALGORITMO  
  DECLARE N1, N2, M NUMÉRICO;  
  ESCREVA "Digite dois números";  
  LEIA N1, N2;  
  M <- N1 + N2;  
  ESCREVA "Soma = ", M;  
FIM_ALGORITMO
```

# Algoritmo











- Alguns tipos de algoritmos
  - O que difere?
    - Consiste em analisar o enunciado do problema e escrever, os passos a serem seguidos para sua resolução.
      - *Descrição narrativa*
        - Utilizando uma linguagem natural
      - *Fluxograma*
        - Utilizando símbolos gráficos predefinidos
      - *Pseudocódigo ou Portugol*
        - Por meio de regras predefinidas

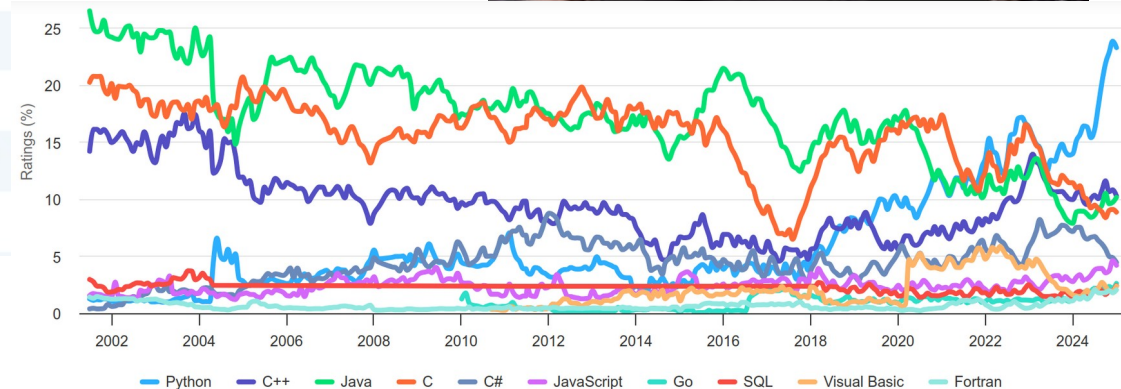
# Por que C?

- Porque deixa para o programador as operações de gerenciamento das estruturas de dados
- Porque permite manipulação de ponteiros de forma explícita
- Porque é a base do C++, C#, Java, JavaScript, Go (Golang), Rust, PHP, Objective-C, Swift, etc.
- Porque o kernels do...
  - **Linux:** Na sua maioria escrito em C com pequenas partes em Assembly
  - **Windows:** Desenvolvido principalmente em C e C++
  - **macOS e iOS:** Construído sobre C e Objective-C.



# Por que C?

Jan 2025	Jan 2024	Change	Programming Language	Ratings	Change
1	1		 Python	23.28%	+9.32%
2	3	▲	 C++	10.29%	+0.33%
3	4	▲	 Java	10.15%	+2.28%
4	2	▼	 C	8.86%	-2.59%
5	5		 C#	4.45%	-2.71%
6	6		 JavaScript	4.20%	+1.43%
7	11	▲▲	 Go		
8	9	▲	 SQL		
9	8	▼	 Visual Basic		
10	12	▲	 Fortran		



Fonte: <<https://www.tiobe.com/tiobe-index/>>  
Acesso em: Fev/2025



# Definições básicas

- Variável

- Representa uma posição na memória, que possui um tipo de dado e um nome (identificador)

- Seu conteúdo pode variar ao longo do tempo
- Só pode armazenar um valor a cada instante

- Nome ou identificador

Primeiro caractere	Deve ser uma letra ou o caractere sublinhado
Caracteres permitidos	Números, letras maiúsculas ou minúsculas e o caractere sublinhado
Caracteres não permitidos	Espaço em branco e caracteres especiais ( @, \$, +, -, %, !, etc. )
Não pode ser utilizadas	Palavras reservadas

# Definições básicas

- Constante

- Representa uma posição na memória, que possui um tipo de dado e um nome (identificador)

- Seu conteúdo não pode variar ao longo do tempo
- Só pode armazenar um valor a cada instante

- Nome ou identificador

Primeiro caractere	Deve ser uma letra ou o caractere sublinhado
Caracteres permitidos	Números, letras maiúsculas ou minúsculas e o caractere sublinhado
Caracteres não permitidos	Espaço em branco e caracteres especiais ( @, \$, +, -, %, !, etc. )
Não pode ser utilizadas	Palavras reservadas
Pré-processador (#)	<code>#define PI 3.1415 // Definida antes do inicio do programa</code>
Palavra reservada	<code>const double PI = 3.1415; // Definida dentro do programa</code>

# Definições básicas

- Tipos básicos

Tipo	Faixa de valores	Tamanho
char	-128 a 127	8 bits
unsigned char	0 a 255	8 bits
int	-32.768 a 32.767	16 bits
unsigned int	0 a 65.535	16 bits
shot int	-32.768 a 32.767	16 bits
unsigned shot int	0 a 65.535	16 bits
long	-2.147.483.648 a 2.147.483.647	32 bits
unsigned long	0 a 4.294.967.295	32 bits
float	$3.4 \times 10^{-38}$ a $3.4 \times 10^{38}$	32 bits
double	$1.7 \times 10^{-308}$ a $1.7 \times 10^{308}$	64 bits
long double	$3.4 \times 10^{-4932}$ a $1.1 \times 10^{4932}$	80 bits

# Definições básicas

- Operadores matemáticos

Op.	Exemplo	Comentário
+	$x + y$	Soma o conteúdo de X com o conteúdo de Y
-	$x - y$	Subtrai o conteúdo de Y do conteúdo de X
*	$x * y$	Multiplica o conteúdo de X pelo conteúdo de Y
/	$x / y$	Obtém o quociente da divisão de X por Y Se os operadores são inteiros, o resultado da operação será o quociente inteiro da divisão <b><code>int z = 5/2 &gt;&gt;&gt; a variável Z receberá 2</code></b> Se os operadores são reais, o resultado da operação será a divisão <b><code>float z = 5/2; &gt;&gt;&gt; a variável Z receberá 2.5</code></b>
%	$x \% y$	Obtém o resto da divisão de X por Y <b><code>int z = 5%2 &gt;&gt;&gt; a variável Z receberá 1</code></b>

# Definições básicas

- Operadores matemáticos de atribuição

Op.	Exemplo	Comentário
=	x = y	Conteúdo da variável Y é atribuído à variável X
+=	x += y	Equivale a $X = X + Y$
-=	x -= y	Equivale a $X = X - Y$
*=	x *= y	Equivale a $X = X * Y$
/=	x /= y	Equivale a $X = X / Y$
%=	x %= y	Equivale a $X = X \% Y$
++	x++	Equivale a $X = X + 1$
	y = x++	Equivale a $Y = X$ e depois $X = X + 1$
	y = ++x	Equivale a $X = X + 1$ e depois $Y = X$
--	x--	Equivale a $X = X - 1$
	y = x--	Equivale a $Y = X$ e depois $X = X - 1$
	y = --x	Equivale a $X = X - 1$ e depois $Y = X$

# Definições básicas

- Operadores comparativos

Op.	Exemplo	Comentário
==	x == y	Verdade se o conteúdo de X é igual ao conteúdo de Y
!=	x != y	Verdade se o conteúdo de X é diferente do conteúdo de Y
<	x < y	Verdade se o conteúdo de X é menor que o conteúdo de Y
>	x > y	Verdade se o conteúdo de X é maior que o conteúdo de Y
<=	x <= y	Verdade se o conteúdo de X é menor ou igual ao conteúdo de Y
>=	x >= y	Verdade se o conteúdo de X é maior ou igual ao conteúdo de Y

# Definições básicas

- Operadores lógicos

Op.	Exemplo	Comentário
-----	---------	------------

&&	x && y	Verdadeiro somente se X e Y forem verdadeiros
----	--------	---

	x    y	Verdadeiro se X ou Y for verdadeiro
--	--------	-------------------------------------

!	!x	Verdadeiro somente se X for falso
---	----	-----------------------------------

- Tabela verdade

&&	1ª Condição	2ª Condição	
Falso	<i>Falso</i>	<i>Falso</i>	<i>Falso</i>
Falso	Falso	Verdadeiro	Verdadeiro
Falso	Verdadeiro	Falso	Verdadeiro
<i>Verdadeiro</i>	<i>Verdadeiro</i>	<i>Verdadeiro</i>	Verdadeiro

# Definições básicas

- Códigos especiais

Código	Significado
--------	-------------

\n	Nova linha
----	------------

\t	Tabulação
----	-----------

\\	\ – Barra invertida
----	---------------------

\'	Aspa simples (apóstrofo)
----	--------------------------

\"	Aspa dupla
----	------------

%c	Caractere simples
----	-------------------

%d	Inteiro decimal com sinal
----	---------------------------

%f	Ponto flutuante em decimal
----	----------------------------

%s	Cadeia de caracteres ( <i>string</i> )
----	--

%u	Inteiro decimal sem sinal
----	---------------------------

%c	Esvazia o <i>buffer</i> após atribuir um valor a uma variável
----	---

**Estrutura de Dados**

**Cicero Tadeu Pereira**



# Estrutura condicional

- Permitir determinar qual é a ação a ser tomada com base no resultado de uma expressão condicional lógica
  - Pode ser usada para selecionar uma ação entre as alternativas disponibilizadas
  - Pode ser usada para verificar opções de escolha
- Outros nomes
  - Estrutura de decisão
  - Estrutura de seleção
  - Desvio condicional
  - Comando de decisão



# Estrutura condicional

- Estrutura condicional simples

- Sintaxe

```
if ( condicao ) {  
    bloco_de_comandos  
}
```

- Semântica

- Se **condicao** for verdadeira **bloco\_de\_comandos** será executado

# Estrutura condicional

- Estrutura condicional simples

- Exemplo

```
void main() {  
    int idade;  
    printf("Qual sua idade?");  
    scanf ("%d", &idade);  
    if ( idade >= 16 ) {  
        printf("Você pode votar");  
    }  
}
```

# Estrutura condicional

- Estrutura condicional composta

- Sintaxe

```
if ( condicao ) {  
    bloco_de_comandos_1  
} else {  
    bloco_de_comandos_2  
}
```

- Semântica

- Se **condicao** for verdadeira **bloco\_de\_comandos\_1** será executado  
Senão **bloco\_de\_comandos\_2** será executado

# Estrutura condicional

- Estrutura condicional composta
  - Exemplo

```
void main() {  
    int idade;  
    printf("Qual sua idade?");  
    scanf ("%d", &idade);  
    if ( idade >= 16 ) {  
        printf("Você pode votar");  
    } else {  
        printf("Você não pode votar");  
    }  
}
```

# Estrutura condicional

- Estrutura condicional composta (aninhamento)

- Sintaxe<sub>1</sub>

```
if ( condicao_1 ) {  
    bloco_de_comandos_1  
} else {  
    if ( condicao_2 ) {  
        bloco_de_comandos_2  
    } else {  
        bloco_de_comandos_3  
    }  
}
```

- Sintaxe<sub>2</sub>

```
if ( condicao_1 ) {  
    bloco_de_comandos_1  
} else if ( condicao_2 ) {  
    bloco_de_comandos_2  
} else {  
    bloco_de_comandos_3  
}
```

# Estrutura condicional

- Estrutura condicional composta (*SWITCH CASE*)

- Sintaxe

```
switch ( variavel ) {  
    case valor_1: bloco_de_comandos_1  
    case valor_2: bloco_de_comandos_2  
    default: bloco_de_comandos_default  
}
```

- Semântica

- Caso **variavel** tenha o **valor\_1** o **bloco\_de\_comandos\_1** será executado
    - Caso **variavel** tenha o **valor\_2** o **bloco\_de\_comandos\_2** será executado
    - Caso não encontre um valor correspondente na lista para **variavel** o **bloco\_de\_comandos\_default** será executado

# Estrutura condicional

- Estrutura condicional composta (*SWITCH CASE*)
  - Exemplo

```
void main() {  
    int num; printf("Qual o número?"); scanf("%d", &num);  
    switch ( num ) {  
        case 1:  
            printf("Número 1"); break;  
        case 2:  
            printf("Número 2"); break;  
        default:  
            printf("Número diferente de 1 e de 2"); break;  
    }  
}
```



# Estrutura de repetição

- Permitir que determinada ação seja repetida com base no resultado de um expressão condicional lógica
  - Pode ser usada para repetir uma ação um número de vezes pré-definido
  - Pode ser usada para repetir uma ação até chegar no estado desejado
  - Pode ser usada para repetir uma ação até que determinada alternativa seja informada
- Outros nomes
  - Laços de repetição
  - Malhas de repetição
  - *Looping*



# Estrutura de repetição

- Repetição com variável de controle (*FOR*)
  - Sintaxe

```
for ( inicializacao; condicao; processamento ) {  
    bloco_de_comandos  
}
```
  - Semântica
    - Para entrar na estrutura **for** é executada **inicializacao** e na sequência é verificada **condicao**
      - Se for **condicao** verdadeira entra na estrutura **for**
    - Após concluir a iteração, voltar para o inicio da estrutura **for** e executada **processamento** e na sequência é verificada a **condicao**
      - Se for **condicao** verdadeira faz outra iteração, caso contrário sai do **for**

# Estrutura de repetição

- Repetição com variável de controle (*FOR*)
  - Exemplo

```
void main() {  
    int i, numero;  
    printf("De 1 até que número? ");  
    scanf("%d", &numero);  
    for ( i = 1; i <= numero; i++ ) {  
        printf("%d\n", i);  
    }  
}
```

# Estrutura de repetição

- Repetição com teste lógico no início (*WHILE*)
  - Sintaxe

```
while ( condicao ) {  
    bloco_de_comandos  
}
```
  - Semântica
    - Se **condicao** for verdadeira entra na estrutura **while** e executa **bloco\_de\_comandos**
      - Permanece na estrutura **while** executando **bloco\_de\_comandos** enquanto a **condicao** for verdadeira

# Estrutura de repetição

- Repetição com teste lógico no início (*WHILE*)
  - Exemplo

```
void main() {  
    int num1, num2;  
    printf("Informe dois números: ");  
    scanf("%d%d", &num1, &num2);  
    while ( num1 < num2 ) {  
        num1 = num1 + 2;  
        num2 = num2 + 1;  
        printf("%d %d\n", num1, num2);  
    }  
}
```

# Estrutura de repetição

- Repetição com teste lógico no fim (*DO-WHILE*)
  - Sintaxe

```
do {  
    bloco_de_comandos  
} while ( condicao )
```
  - Semântica
    - Executa **bloco\_de\_comandos** uma vez e testa **condicao**
      - Permanece na estrutura **do-while** executando **bloco\_de\_comandos** enquanto a **condicao** for verdadeira

# Estrutura de repetição

- Repetição com teste lógico no fim (*DO-WHILE*)
  - Exemplo

```
void main() {  
    int num1, num2;  
    do {  
        printf("Informe dois números: ");  
        scanf("%d%d", &num1, &num2);  
        printf("%d\n", num1 * num2);  
    } while (num1 * num2 != 0);  
}
```

# Exercícios de fixação

- Criar um programa que...
  - i. dado dois números inteiros, mostre o resultado da soma.
  - ii. calcule o valor do pagamento de uma parcela em atraso cobrando juros de 0.5% ao dia.
  - iii. verifique o maior número entre três informados, assumindo que os valores são diferentes entre si.
  - iv. mostre a série *Fibonacci* até a posição informada pelo usuário
  - v. mostre se um número digitado pelo usuário é primo



# Desafio

1. Desenvolva um programa na linguagem de programação C que verifique o maior número entre três informados, examinando se os valores são diferentes entre si.
2. Fazer um programa na linguagem de programação C que mostre o fatorial de um número informado pelo usuário.