

Fundamentos de Programação

Algoritmo

Sumário

1 Lógica

2 Definição

3 Representações

Lógica

Lógica:

- › Origem na filosofia (Platão).
- › Normalmente, relacionada à **coerência e à racionalidade**.
- › Associada a matemática, no entanto, tem relação com as demais ciências.
- › "**Correção do pensamento**" ou "**arte de bem pensar**".
- › **Silogismos:**
 - » Um argumento composto por duas premissas e uma conclusão;
 - » Pode haver uma relação válida ou não.

Exemplo:

- › Todo mamífero é um animal.
- › Todo cavalo é um mamífero.
- › **Portanto, todo cavalo é um animal.**

Exemplo Cotidiano:

- › A gaveta está fechada.
- › A caneta está dentro da gaveta.
- › **Precisamos primeiro abrir a gaveta para depois pegar a caneta.**

› Lógica de programação:

- ›› Podemos aplicar o raciocínio formal em problemas computacionais.
- ›› O raciocínio é abstrato, no entanto, **pode ser expresso em diferentes idiomas**.
- ›› Da mesma forma, também pode ser representado em linguagens de programação.
- ›› Uma linguagem de programação é um **sistema formal de comunicação entre humanos e computadores** e, que permite criar **instruções** para que um computador.
- ›› **Algoritmos** nos ajudam a representar a lógica de programação.

Definição

O que é um algoritmo?

*Um procedimento **passo a passo** para a solução de um problema.*

*Uma **sequência detalhada de ações** a serem executadas para realizar alguma tarefa.*

(Medina; Fertig, 2005)

*Pode ser definido como uma **sequência de passos** que visam a atingir um objetivo bem definido.*

(Forbellone; Eberspächer, 2005)

É uma *sequência de etapas* computacionais que transformam a entrada na saída.

(Cormen *et al.*, 2012)

Portanto, O que é um algoritmo?

Algoritmo:

Um conjunto de regras e operações **bem definidas e ordenadas**, destinadas à solução de um problema ou de uma classe de problemas, em um **número finito de passos** (De Oliveira; Manzano, 2004).

O que os algoritmos podem fazer? (Cormen *et al.*, 2012)

- **Projeto Genoma Humano:** O projeto exige algoritmos sofisticados para identificar genes, determinar sequências do DNA e armazenar dados em bancos de dados, permitindo economias de tempo e dinheiro.
- **Internet e Algoritmos:** Sites usam algoritmos para gerenciar grandes volumes de dados e resolver problemas como determinar rotas de transmissão e encontrar informações rapidamente em mecanismos de busca.
- **Comércio Eletrônico e Criptografia:** O comércio eletrônico depende de algoritmos de criptografia de chave pública e assinaturas digitais para garantir a segurança das informações pessoais, como dados de cartões de crédito.
- **Alocação de Recursos:** Empresas e organizações usam algoritmos para alocar recursos escassos de maneira otimizada, como na localização de poços de petróleo ou na alocação de recursos para campanhas eleitorais ou voos de transporte aéreo.

Exemplo - Trocar uma lâmpada

- › Pegar uma escada;
- › Posicionar a escada embaixo da lâmpada;
- › Buscar uma lâmpada nova;
- › Subir na escada;
- › Retirar a lâmpada velha;
- › Colocar a lâmpada nova.

Exemplo - Trocar um pneu de carro:

- › Desligar o carro;
- › Pegar as ferramentas necessárias;
- › Pegar o estepe;
- › Suspender o carro com o macaco;
- › Remover os 4 parafusos com a chave de roda;
- › Retirar o pneu furado;
- › Colocar o estepe no lugar do pneu furado;
- › Apertar os 4 parafusos com a chave de roda;
- › Baixar o carro com o macaco;
- › Guardar o pneu furado no porta-malas;
- › Guardar as ferramentas;
- › Ligar o carro.

Exemplo - Enviar um Pix:

- › Abrir o aplicativo do banco no seu dispositivo.
- › Selecionar a opção de transferência via Pix.
- › Incluir a chave Pix do recebedor.
- › Inserir o valor que será enviado ao recebedor.
- › Confirmar todos os dados da transferência.
- › Enviar o comprovante da transação ao recebedor para confirmação.

Como seria o algoritmo do aplicativo do banco?

Exemplo - Enviar um Pix:

- › Ler a chave Pix.
- › Ler o valor que deve ser enviado.
- › Subtrair o valor informado do saldo do pagador.
- › Adicionar o valor informado ao saldo do recebedor.
- › Exibir uma mensagem de confirmação.

Representações

Formas de representação:

- › Forma gráfica:
 - » Diagrama de Blocos (ISO 5807:1985);
 - » Fluxograma.
- › Forma textual:
 - » Pseudocódigo ou metalinguagem;
 - » Uma linguagem de projeto de programação não pode e não deve ter o mesmo rigor sintático¹ que possui uma linguagem de programação formal.
 - » Grau de rigidez sintática intermediário, entre linguagem natural e linguagem de programação;
 - » Pode ser escrito em idioma nativo;
 - » Portugol ou português estruturado.

¹**Sintaxe:** Conjunto de regras que explica como as palavras e símbolos devem ser organizados em uma linguagem para que as instruções sejam compreendidas corretamente.

Em uma receita de bolo, temos ingredientes e o modo de preparo.

Da mesma forma, em um algoritmo, temos dados que pode ser de entrada ou saída.

E instruções, que devem seguir uma sequência, assim como o modo de preparo na receita.

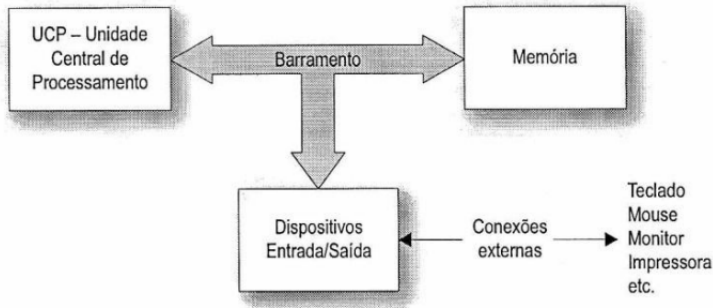


Figura 1: Arquitetura simplificada de um computador (Medina; Fertig, 2005).

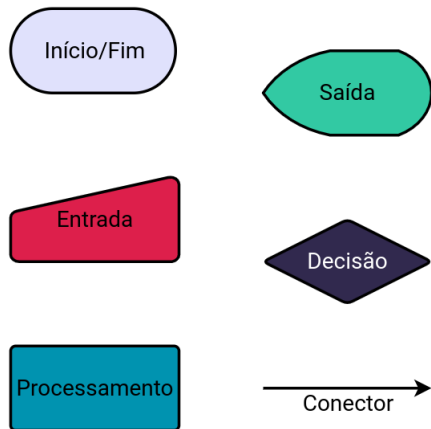


Figura 2: Símbolos utilizados no fluxograma.

Fluxograma II

Atenção!

- O processamento ou decisão devem ser representados com expressões aritméticas:

$+, -, \times, \div, >, <, \geq, \leq, =, \neq$

- Os dados de entrada ou saída devem ser definidos.
- Os símbolos devem ser conectados formando um fluxo.
- O fluxograma deve ter apenas um início e término.

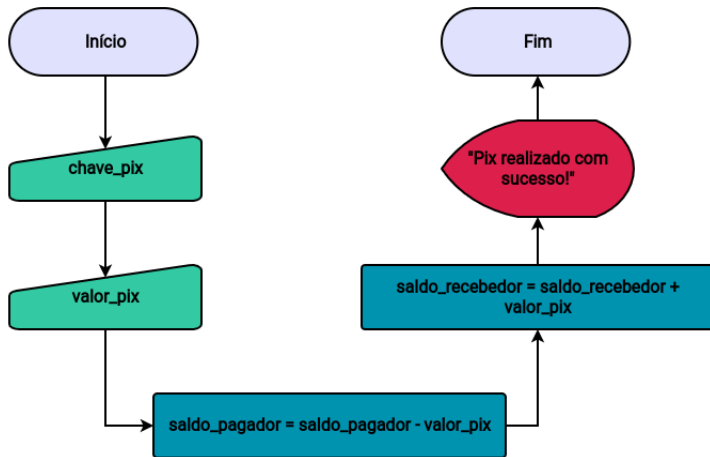


Figura 3: Exemplo Pix em fluxograma ².

²Na Figura 3 partimos da premissa que o saldo do pagador e recebedor é definido.

```
1 INICIO
2   leia(chave_pix)
3   leia(valor_pix)
4   saldo_pagador = saldo_pagador - valor_pix
5   saldo_recebedor = saldo_recebedor + valor_pix
6   escreva("Pix realizado com sucesso!")
7 FIM
```

Código 1: Exemplo Pix em pseudocódigo.

Como seria o algoritmo do aplicativo do banco?

Exemplo - Fazer um pagamento através de Pix com validação:

- Ler a chave Pix.
- Ler o valor que deve ser enviado.
- Se o saldo for maior ou igual ao valor do pix:
 - Subtrair o valor informado do saldo do pagador.
 - Adicionar o valor informado ao saldo do recebedor.
 - Exibir uma mensagem de confirmação.
- Senão:
 - Exibir uma mensagem de negação.

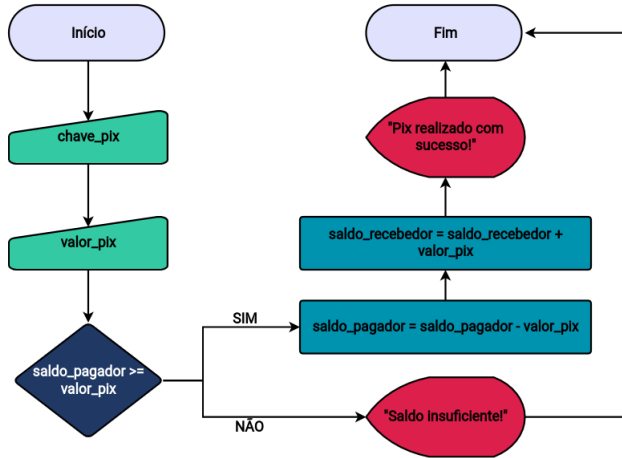


Figura 4: Exemplo de Pix em fluxograma com decisão.

```
1 INICIO
2   leia(chave_pix)
3   leia(valor_pix)
4   SE (saldo_pagador >= valor_pix) ENTAO
5       saldo_pagador = saldo_pagador - valor_pix
6       saldo_recebedor = saldo_recebedor + valor_pix
7       escreva("Pix realizado com sucesso!")
8   SENAO
9       escreva("Saldo insuficiente!")
10  FIMSE
11 FIM
```

Código 2: Exemplo Pix com decisão em pseudocódigo.

```
1 chavePix = input()
2 valorPix = float(input())
3 if(saldoPagador >= valorPix):
4     saldoPagador = saldoPagador - valorPix
5     saldoRecebedor = saldoRecebedor + valorPix
6     print("Pix realizado com sucesso!")
7 else:
8     print("Saldo insuficiente!")
```

Código 3: Exemplo Pix em Python.

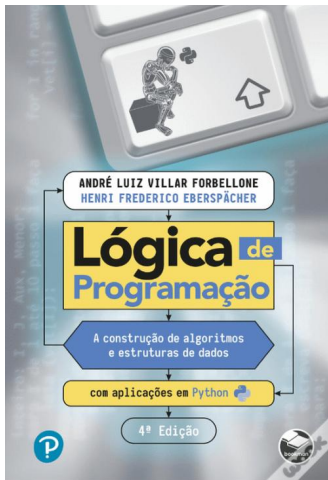
```
1 int main() {
2     scanf("%s", chave_pix);
3     scanf("%f", &valor_pix);
4     if (saldo_pagador >= valor_pix) {
5         saldo_pagador = saldo_pagador - valor_pix
6         saldo_recebedor = saldo_recebedor + valor_pix
7         printf("Pix realizado com sucesso!\n");
8     } else {
9         printf("Saldo insuficiente!\n");
10    }
11    return 0;
12 }
```

Código 4: Exemplo Pix em C.

- › **Clareza e Precisão:** Os algoritmos devem ser claros e precisos, de modo que qualquer pessoa possa compreendê-los sem ambiguidades.
- › **Finitude:** Um algoritmo deve ter um número finito de passos para ser concluído. Isso garante que o algoritmo não entre em um *loop* infinito.
- › **Entrada e Saída:** Todo algoritmo tem dados de entrada e produz dados de saída. É importante entender claramente quais são esses dados para projetar o algoritmo corretamente.
- › **Sequência:** Os passos do algoritmo devem ser executados em uma ordem específica, garantindo que cada ação seja realizada no momento adequado.

- › **Seleção:** Decisões devem ser tomadas dentro do algoritmo com base em condições específicas. Isso é feito através de estruturas de controle, como condicionais (`if-else`) ou `switch-case`.
- › **Repetição:** Alguns passos do algoritmo podem precisar ser repetidos várias vezes até que uma condição específica seja atendida. Isso é feito usando estruturas de repetição, como loops (`for`, `while`).
- › **Abstração:** Os algoritmos devem ser construídos de forma abstrata, focando nos passos essenciais para resolver o problema, sem se preocupar com detalhes irrelevantes.
- › **Modularidade:** Os algoritmos devem ser divididos em partes menores, ou módulos, para facilitar a compreensão e manutenção do código. Isso também promove a reutilização de código.

(Forbellone; Eberspächer, 2005) - Capítulo 1 e 2.





CORMEN, T.H. *et al.* **Algoritmos: Teoria e Prática**. 3. ed. Rio de Janeiro: Elsevier, 2012.



DE OLIVEIRA, J.F.; MANZANO, J.A.N.G. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 16. ed. São Paulo: Editora Érica, 2004.

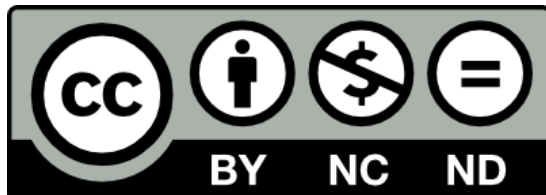


FORBELLONE, A.L.V.; EBERSPÄCHER, H.F. **Lógica de Programação**. 3. ed. São Paulo: Makron Books, 2005.



MEDINA, M.; FERTIG, C. **Algoritmos e Programação – Teoria e Prática**. São Paulo: Novatec, 2005.

Estes slides estão protegidos por uma licença Creative Commons



Este modelo foi adaptado de Maxime Chupin.

Fundamentos de Programação

Algoritmo