NOTAS DE AULA
INTRODUÇÃO À PROGRAMAÇÃO COM C

15 de abril de 2025

Prof. Maurício Massaru Arimoto
UENP

Sumário

1 O que é programação?

Segundo a Sociedade Brasileira de Computação (SBC) [Raabe 2017], é difícil imaginar uma sociedade na qual os indivíduos não necessitem de habilidades e conhecimentos básicos de Computação, tão importantes para a vida na sociedade contemporânea quanto os conhecimentos básicos de Matemática, Física, ou outras ciências.

A programação de computadores se insere nesse contexto, sendo uma das habilidades básicas de todo cientista ou engenheiro da computação. Ela exerce um papel fundamental na formação dos estudantes, abordando princípios de lógica e programação que visam desenvolver a capacidade de análise e de resolução de problemas. Basicamente, a programação consiste em criar soluções para problemas diversos e fazer com que o computador execute-as de forma automatizada.

Programas de computadores são criados (ainda) por seres humanos, que utilizam a capacidade de raciocínio para gerar soluções para os problemas propostos. No entanto, o raciocínio é algo intangível.

Neste sentido, a lógica de programação é uma ferramenta para transformar esse raciocínio abstrato em algo que possa ser transformado em programa de computador. Na programação a lógica pode ser entendida como o modo corerente com que organizamos as ideias para resolver um problema.

Um computador só entende o que deve fazer se receber instruções claras, bem definidas e em ordem correta. Computadores (ainda) não são programáveis em linguagem natural, pois esta é sujeita a ambiguidades e imprecisões. A lógica de programação tem o objetivo de estabelecer uma sequência lógica de passos que devem ser executados por um programa de computador. Essa sequência costuma ser chamada de algoritmo.

2 Algoritmos

2.1 Conceitos de Algoritmos

Para criarmos um programa de computador para realizar uma determinada tarefa, devemos ser capazes de construir algoritmos. Um algoritmo é um conjunto de instruções, bem definido (sem ambiguidades), para a solução de um problema dentro de um determinado contexto em tempo finito.

Um exemplo clássico de algoritmo é o processo de troca de uma lâmpada queimada, cujos passos seriam:

1. Pegar a escada;
2. Posicionar a escada embaixo da lâmpada;
3. Buscar a lâmpada nova;
4. Retirar a lâmpada velha;
5. Colocar a lâmpada nova.

Agora, imagine o seguinte problema [Paes 2016]: de um lado do rio temos um homem, uma raposa, uma galinha e um milho. O objetivo é transportar todos eles para outro lado. Para isso, existe uma canoa, mas o homem só pode transportar um por vez. Além disso, a raposa não pode ficar sozinha com a galinha, pois a raposa comerá a galinha. Ainda, se a galinha ficar sozinha com o milho, o milho será devorado.

Qual é a sequência de instruções necessárias para que esse problema seja resolvido? A Tabela 1 apresenta uma possível solução para esse problema, demonstrado passo a passo todas as ações/instruções necessárias para transportar todos com segurança para o outro lado do rio. Essa sequência de ações constitui um algoritmo.

Cabe ressaltar que na Tabela 1 todos os itens são representados pelas suas letras iniciais, ou seja, o homem (H), a raposa (R), a galinha (G) e o milho(M).

Os exemplos supracitados são solucionados seguindo uma sequência de ações bem definidas, que devem ser executadas em uma determinada ordem. Outras aplicações de nosso dia a dia podem ser detalhadas de forma semelhante: o acesso a terminais eletrônicos de bancos, a troca do pneu de um carro, etc.

Além dos algoritmos usados para resolver problemas do dia a dia, podemos desenvolver algoritmos que podem ser transformados em programas e executados em computadores.

Tabela 1: Solução para o problema proposto

Passo	Esquerda	Direita	Descrição
0	H R G M		Estado inicial
1	R M	H G	Leve a galinha para outro lado
2	H R M	G	Volte sozinho
3	M	H R G	Leve a raposa para outro lado
4	H G M	R	Volte com a galinha
5	G	H R M	Leve o milho para o outro lado
6	H G	R M	Volte sozinho
7		H R G M	Leve a galinha para outro lado

Este texto concentra-se em problemas resolvidos através de algoritmos que podem ser integralmente executados por computadores.

2.2 Algoritmos Computacionais

Suponha que você decidiu efetuar a compra de um smarphone em algum site de comércio eletrônico:

- Ao entrar no site e fazer sua busca, um algoritmo vai analisar suas atividades e recomendar produtos para você comprar.
- Ao efetuar um pagamento no site, um algoritmo vai verificar seu cartão de crédito.
- Ao finalizar a compra e efetuar o pedido, um algoritmo vai identificar a maneira mais eficiente de entregar o produto com comodidade na sua casa.

Agora, suponha que você tenha desistido de comprar pela Internet. Mesmo assim, pode existir um algoritmo para maximizar as vendas colocando os produtos na prateleira de maneira diferenciada.

Enfim, na Computação tudo vira algoritmos!

Os algoritmos computacionais diferem de outros algoritmos convencionais pois são propostos para serem executados por computador. Eles auxiliam o desenvolvedor na concepção da solução de um problema, independentemente da linguagem de programação a ser utilizada na sua construção.

Um algortimo, quando programado em um computador, é constituído basicamente por três partes:

1. **Entrada de dados**;

2. **Processamento de dados**;

3. **Saída de dados**.

Analisando o problema aqui colocado, para obter a média de quatro provas de um aluno, temos como:

1. **Entrada de dados**: as quatros notas de provas de um aluno obtidas via teclado, por exemplo;

2. **Processamento de dados**: soma das quatro notas e o respectivo cálculo da média das notas;

3. **Saída de dados**: resultado da média calculada e mostrada na tela, por exemplo.

Para construção de algoritmos, algumas premissas básicas devem ser adotadas:

- Definir ações claras e precisas (não-ambíguas);
- Organizar as ações de forma ordenada (lógica); e
- Estabelecer as ações dentro de uma sequência finita de passos (previsível).

Além disso, existem algumas diretrizes que auxiliam na construção de algoritmos computacionais:

- Identificar e analisar detalhadamente o problema a ser resolvido;
- Identificar as entradas do sistema, ou seja, quais informações serão fornecidas como entrada;
- Identificar as saídas do sistema, ou seja, quais informações deverão ser produzidas e/ou calculadas;
- Definir os passos a serem realizados, ou seja, a sequência de ações para a solução do problema – transformar entradas em saídas;
- Implementar o algoritmo, ou seja, registrar a sequência de comandos, usando alguma notação. Por exemplo, pseudolinguagem/pseudocódigo); e
- Testar a solução proposta, ou seja, executar manualmente cada passo do algoritmo para detectar erros. Por exemplo, realizando um teste de mesa).

Analisando o problema para obter a média de 4 notas de provas de um aluno, precisamos realizar basicamente quatro operações na ordem a seguir:

- Passo 1: receber as notas das quatro provas.
- Passo 2: somar as notas das provas.
- Passo 3 calcular a média das notas das provas.
- Passo 4: mostrar o resultado obtido das notas das provas.

Na Figura 1 apresenta-se um exemplo do uso de fluxograma na representação do algoritmo para o cálculo da média de quatro notas de provas de um aluno. Cada ação/comando é representado por um bloco, sendo os blocos interligados por linhas dirigidas (setas) que representam o fluxo de execução. Cada forma de bloco representa uma ação.

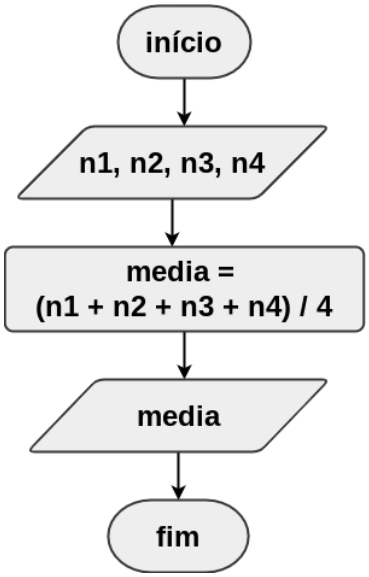

Figura 1: Fluxograma: média das notas das provas de um aluno

No algoritmo 1 é apresentado um exemplo de um pseudocódigo para solucionar o problema do cálculo da média das notas das provas de um aluno.

Algoritmo 1: MediaNotas

```
1 var n1, n2, n3, n4: real
2 var media: real
3 início
4 |  leia (n1, n2, n3, n4)
5 |  media ← (n1 + n2 + n3 + n4)/ 4
6 |  escreva (media)
7 fim
```

2.2.1 Eficácia e Eficiência de Algoritmos

Durante a construção de um algoritmo executado por computador, dois aspectos de qualidade devem ser considerados [Edelweiss and Livi 2014]:

1. **Eficácia** (corretude, exatidão): um algoritmo deve exercer corretamente a tarefa para a qual foi proposto. Além disso, o algoritmo deve fornecer resultados corretos para quaisquer que sejam os dados fornecidos como entrada. A eficácia de um algoritmo deve ser bastante testada antes que ele seja implementado em um computador. A abordagem mais simples de testar um algoritmo é por meio de "teste de mesa", em que se simula a execução do algoritmo em papel, usando diferentes dados de entrada.

2. **Eficiência**: geralmente a solução de um problema não é única, podendo ser construídos diferentes algoritmos para resolvê-los. Entretanto, em alguns casos não se sabe qual é a melhor solução. Neste caso, pode-se calcular qual a forma mais eficiente, com base em dois critérios: tempo de execução e espaço de memória ocupado.

Um exemplo da diferença entre eficácia e eficiência pode ser observado na receita de ovo mexido mostrada a seguir [Edelweiss and Livi 2014]:

1. Ligar o fogão em fogo baixo;

2. Separar 1 ovo, 1 colher de sobremesa de manteiga e sal a gosto;

3. Quebrar o ovo em uma tigela e colocar sal na tigela;

4. Misturar levemente o ovo e o sal, com um garfo;
5. Aquecer a manteiga na frigideira até que comece a derreter;
6. Jogar o ovo na frigideira, mexendo com uma colher até ficar firme;
7. Retirar da frigideira e servir.

Se analisarmos o algoritmo mencionado, podemos notar que embora o ovo mexido seja obtido, garantindo a eficácia da receita, existe uma clara ineficiência em relação ao gasto de gás, uma vez que ligar o fogão não é pré-requisito para a quebra do ovo e mistura do ovo e do sal. Neste caso, se modificarmos apenas a sequência das ações, podemos ter um algoritmo eficaz e mais eficiente:

1. Separar 1 ovo, 1 colher de sobremesa de manteiga e sal a gosto;
2. Quebrar o ovo em uma tigela e colocar sal nesta tigela;
3. Misturar levemente o ovo e o sal, com um garfo;
4. Ligar o fogão em fogo baixo;
5. Aquecer a manteiga na frigideira até que comece a derreter;
6. Jogar o ovo na frigideira, misturando com uma colher até ficar firme;
7. Retirar da frigideira e servir.

Conforme já mencionado no início desta seção, focamos nos algoritmos computacionais que são executados em computador. Diante disso, precisamos entender como o computador executa as instruções descritas por meio de nossos algoritmos.

3 Computadores

Um sistema computacional é constituído basicamente de processador, memória para armazenamento de informações, dispositivos de entrada e saída (E/S) e barramentos para a interconexão entre esses componentes principais.

O processador, também chamado de Unidade Central de Processamento (UCP), é um componente essencial para o funcionamento do sistema computacional. Ele é responsável pela execução de programas (instruções) armazenados na memória principal e processamento de operações lógicas e aritméticas, devolvendo os resultados apropriados para uso do usuário do sistema.

A UCP realiza as operações lógicas e aritméticas por meio da Unidade Lógica e Aritmética (ULA), faz o controle dessas operações por meio da Unidade de Controle (UC) e armazena os dados ou resultados temporários usando registradores.

3.1 Arquitetura de um Computador

Na Figura 2 [Tocci et al. 2019] é apresentada uma arquitetura típica de um computador com os dispositivos de entrada e saída, memórias, e a UCP, composta pela ULA E pela UC, além de registradores de armazenamento temporário.

A UCP é responsável pela execução de todas as instruções dos programas armazenados na memória principal e o processamento de operações lógicas e aritméticas, devolvendo os resultados apropriados para uso dos usuários do sistema. Essas instruções primitivas são denominadas instruções de máquina e, quando agrupadas, formam os programas.

A UCP realiza constantemente as seguintes tarefas [Stallings 2010]:

- **Buscar instrução**: o processador busca na memória a instrução a ser executada.
- **Interpretar a instrução**: a instrução é decodificada para determinar a ação que deve ser executada.
- **Obter os dados**: a execução da instrução pode necessitar da leitura de dados da memória ou dos dispositivos de entrada.
- **Processar os dados**: a execução da instrução pode necessitar de alguma operação aritmética ou lógica com os dados.
- **Gravar os dados**: a execução da instrução pode requerer a gravação dos dados na memória ou em um dispositivo de saída.

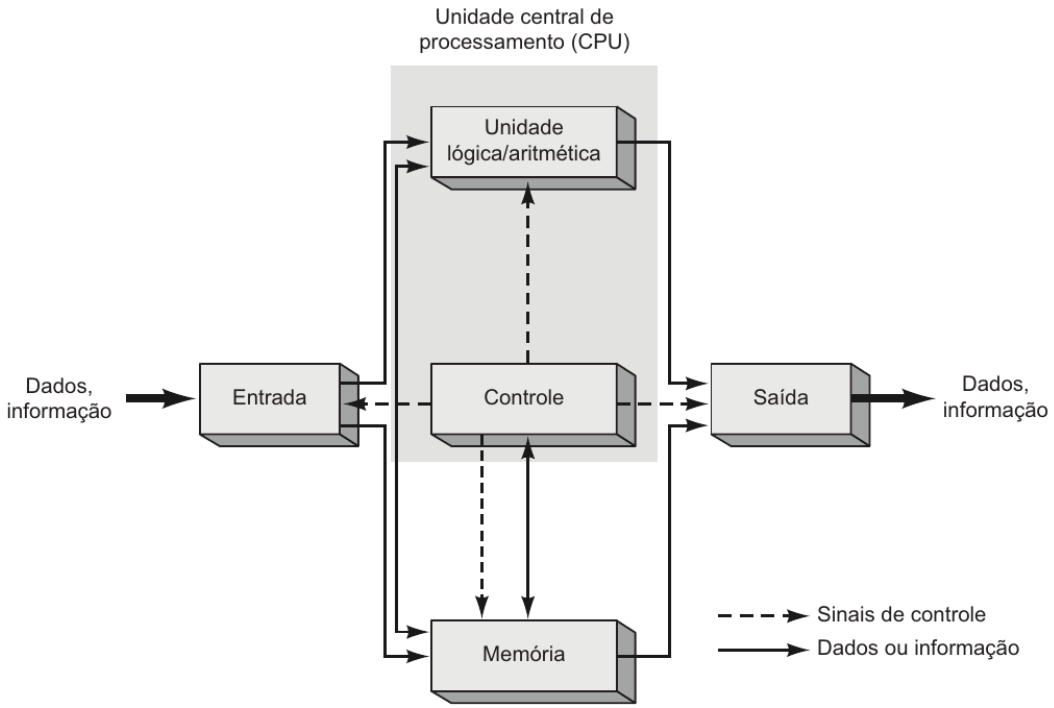

Figura 2: Arquitetura de um computador [Tocci et al. 2019]

Todas as operações lógicas e aritméticas são realizadas na ULA de um computador. O objetivo principal de uma ULA é receber dados binários armazenados na memória e executar operações aritméticas e lógicas sobre eles, de acordo com instruções provenientes da UC.

A UC tem como função dirigir e coordenar as atividades das demais unidades do sistema. A UC é responsável pela busca das instruções na memória principal, sua decodificação e execução ([Tocci et al. 2019]):

- Busca da instrução que será executada, armazenando-a em um registrador da UCP;
- Interpretação das instruções a fim de saber quais operações deverão ser executadas pela ULA (ex.: soma, subtração, comparação) e como realizá-las;
- Geração de sinais de controle apropriados para a ativação das atividades necessárias à execução da instrução identificada. Esses sinais de controle são enviados aos diversos componentes do sistema, sejam eles internos à UCP (ex.: a ULA) ou externos (ex.: memória e dispositivos de entrada e saída).

Os computadores usam a memória principal para guardar:

- As informações que serão executadas pela UCP;
- As informações que estão sendo acessadas muitas vezes; e
- Os valores dos programas que estão sendo executados naquele momento.

Por fim, os dispositivos de E/S, também chamados de periféricos são responsáveis por fazer a comunicação do sistema em si com o usuário ou meio externo e pelo armazenamento de dados para posterior recuperação. Um dispositivo de E/S é qualquer dispositivo conectado a um computador de forma a possibilitar a interação do computador com o mundo externo. Atualmente, existe uma grande variedade de dispositivos, desde aqueles desenvolvidos para permitir a comunicação do homem com o computador (teclado, mouse, monitor de vídeo, etc) até dispositivos que possibilitam a comunicação entre computadores (modems, placas de redes, etc), ou ainda aqueles destinados ao armazenamento de informações (disco rígido, pen-drive, etc).

3.2 Linguagens de Programação

O computador trabalha com um alfabeto de símbolos composto apenas por 0 e 1, ou seja, sistema binário. Para representar quantidade, por exemplo, usamos a combinação de 0s e 1s, seguindo a mesma lógica dos números decimais no qual usamos a combinação de números entre 0 a 9 para formar números maiores que 9. A Tabela ilustra números de 0 a 12 em decimal e seus valores correspondentes em binário.

Tabela 2: Números em decimal e binário

Decimal	Binário
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0

Se computador só entende binário, como devemos escrever nossas instruções (programas) de forma que ele entenda? Certamente, escrever em binário não é a melhor escolha. Para isso, usamos uma linguagem de programação que é mais fácil de entendermos – um meio termo entre linguagem de máquina e linguagem humana.

3.3 Linguagem C

Ainda precisamos converter o que escrevemos em uma linguagem de programação para uma "linguagem binária" – linguagem de máquina. Existem duas formas de fazer essa conversão: por meio do uso de um compilador ou interpretador. Em programas escritos em linguagem de programação C, por exemplo, usamos um compilador para criar um executável nativo para o programa. O compilar lê o programa inteiro e converte-o em um código-objeto, que é uma tradução do código-fonte em uma forma que o computador possa executar diretamente. Em outras linguagens de programação como por exemplo, Python, utiliza-se um interpretador que lê o código-fonte do programa uma linha por vez, executando a instrução específica contida nessa linha.

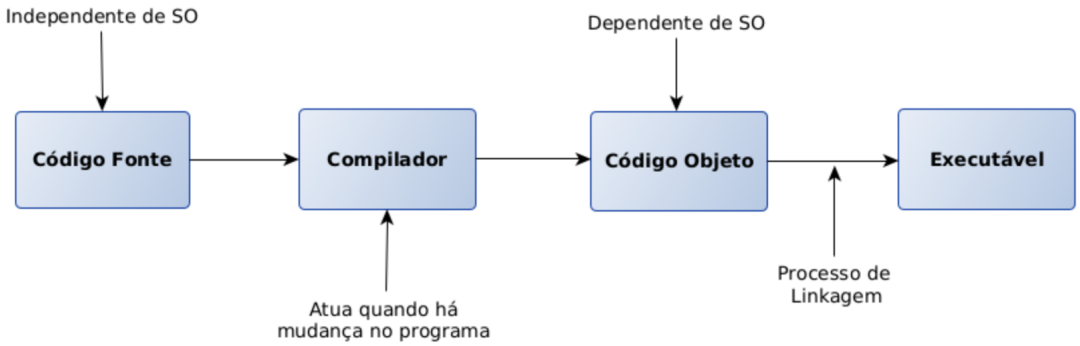

Figura 3: Linguagem compilada

Referências

[Edelweiss and Livi 2014] Edelweiss, N. and Livi, M. A. C. (2014). *Algoritmos e programação com exemplos em Pascal e C*. Bookman.

[Paes 2016] Paes, R. B. (2016). *Introdução à programação com a linguagem C*. Novatec.

[Raabe 2017] Raabe, A. L. A. (2017). Referenciais de formação em computação: Educação básica. In *XXXVII Congresso da Sociedade Brasileira de Computação (CSBC)*, pages 1–9. Sociedade Brasileira de Computação. julho/2017.

[Stallings 2010] Stallings, W. (2010). *Arquitetura e Organização de Computadores*. São Paulo: Pearson Prentice Hall, 8 ed. edition.

[Tocci et al. 2019] Tocci, R., Widmer, N., and Moss, G. (2019). *Sistemas Digitais: Princípios e aplicações*.