1

TEXTO BASE

LINGUAGEM DE PROGRAMAÇÃO

Faculdade IMPACTA

Texto base

1

Introdução à disciplina e conceitos iniciais

Prof. Me. Lucio Nunes de Lira
Prof. MSc. Rafael Maximo Carreira Ribeiro

Resumo

Nesta aula os objetivos são: (I) introduzir as diretrizes da disciplina, como conteúdo programático, metodologia de ensino, critérios de avaliação e bibliografia; (II) conceituar os termos "algoritmo", "lógica de programação", "linguagem de programação", "código-fonte" e "programa"; (III) simular o processo para resolução de problemas computacionais sem o uso de computadores (computação desplugada).

1.1. Motivação

A vida das pessoas é constantemente alterada pela evolução das tecnologias, inclusive pela tecnologia computacional. Diversos aspectos da vida são impactados pelo progresso das descobertas relacionadas à área da Ciência da Computação. Você pode imaginar o quão diferente seria sua rotina sem a existência de *notebooks* e *smartphones*?

Brookshear (2008, p. 17) menciona que, como uma disciplina, "[a Ciência da Computação] busca construir uma base científica para diversos tópicos, tais como a construção e a programação de computadores, o processamento de informações, as soluções algorítmicas de problemas e o processo algorítmico propriamente dito.". Podemos resolver problemas com a construção de *algoritmos* e, para automatizá-los, aplicaremos *lógica de programação* com instruções em *linguagens de programação*, criando *códigos-fonte* que se tornarão *programas* executados por computadores.

1.2. Diretrizes da disciplina

Para verificar as diretrizes da disciplina, conteúdo programático, metodologia de ensino, critérios de avaliação e bibliografia, consulte o documento "Plano de Ensino".

1.3. Algoritmos

Conforme descrito por Ziviani (1999, p. 1) "[algoritmo é] uma sequência de passos de ações executáveis para a obtenção de uma solução para um determinado tipo de problema". Podemos imaginar um algoritmo como um procedimento que visa resolver um problema, formado por uma sequência ordenada e finita de passos passíveis de serem executados, assim como conceituado por Pereira (2010).

O termo algoritmo não está diretamente relacionado aos computadores, muito menos depende deles, pois podemos criar algoritmos que não serão, necessariamente, executados em um computador. Imagine, por exemplo, uma receita de bolo:

1) Deixe disponível em sua bancada os ingredientes necessários: açúcar, farinha de trigo, margarina, ovos, leite e fermento em pó;
2) Bata as claras dos ovos para formar claras em neve;
3) Misture a margarina com as gemas e o açúcar de modo a obter uma massa homogênea;
4) Aos poucos coloque leite e farinha de trigo na massa, continue batendo;
5) Acrescente as claras em neve e o fermento na massa;
6) Despeje a massa em uma forma adequada, untada e enfarinhada;
7) Coloque a forma em um forno em temperatura média, por volta de 180 ºC, por cerca de 40 minutos.

Essa receita de bolo é um algoritmo! Note como os algoritmos fazem parte do dia a dia de qualquer pessoa. Existem algoritmos que são executados inconscientemente, respirar é um deles. Quando você se desloca para o trabalho, executa um algoritmo, pois segue uma sequência de passos logicamente ordenados para resolver um problema, que podemos interpretar como atingir um objetivo. Desde trocar de roupa, passando por embarcar em um transporte, até registrar sua entrada no local da empresa.

Assim como ficou evidente no algoritmo para construir um bolo, é possível criar algoritmos com maior ou menor detalhamento. Você reparou que não foram definidas as quantidades dos ingredientes? Isso pode ser um problema para algumas pessoas que tentarão executar esse algoritmo e não tem conhecimento prévio. Já para outras pessoas mais experientes, que apenas precisam lembrar dos passos gerais, pode ser suficiente.

Uma característica importantíssima sobre os algoritmos é que a construção pode variar de acordo com quem os elabora. Portanto, nem todos os algoritmos que objetivam resolver o mesmo problema terão a mesma sequência de passos, afinal dependerá, dentre outros fatores, da experiência, vocabulário, concisão e *lógica* do autor. Em contraste, a execução dos algoritmos, se forem bem formulados, independe de quem os executará, bastando que os passos sejam rigorosamente seguidos.

Genericamente, podemos dizer que lógica é uma parte da filosofia que trata das formas do pensamento em geral (dedução, indução, inferência, hipótese etc.) e das

operações intelectuais que visam a determinação do que é verdadeiro ou falso. Faremos uso de lógica para escrever as instruções de nossos algoritmos, de modo que a execução de cada instrução faça sentido para atingir o objetivo final.

Segundo um dos mais importantes nomes da Computação, Knuth (1997), um algoritmo tem cinco características fundamentais:

1) *Finitude:* um algoritmo precisa terminar após um número finito de passos;
2) *Definição:* cada passo do algoritmo precisa ser precisamente definido, as ações devem ser especificadas clara e rigorosamente evitando ambiguidade;
3) *Entrada:* um algoritmo tem zero ou mais entradas, que são dados fornecidos ao algoritmo ao ser iniciado, ou dinamicamente conforme for executado;
4) *Saída:* um algoritmo tem zero ou mais saídas, que geralmente estão relacionadas com os dados fornecidos nas entradas;
5) *Eficácia:* é esperado que um algoritmo seja eficaz, no sentido que suas operações devem ser suficientemente básicas para que possam, a princípio, ser feitas precisamente e em um tempo finito por alguém com lápis e papel.

Nesta disciplina nossa atenção será direcionada para a *lógica de programação*, para que possamos estudar as estruturas básicas para a construção de algoritmos destinados a serem executados em computadores com uso de uma *linguagem de programação*. Porém, antes de direcionarmos nossos estudos aos algoritmos propostos para solucionar problemas computacionais, é preciso entender o que é a *abstração* e como criar *representações* de problemas por meio da abstração.

1.4. Abstração e representação

Para resolver problemas computacionalmente são necessários pelo menos dois instrumentos: (I) uma *representação* que capte todos os aspectos relevantes do problema, descartando detalhes insignificantes para a sua resolução e; (II) um *algoritmo* que resolva o problema com base em ações aplicadas na representação construída.

Compreenderemos a importância da representação no desenvolvimento de algoritmos analisando um clássico problema de lógica, que chamaremos de “o homem e o rio”. Este problema está ilustrado na capa da obra de Dierbach (2012), conforme a figura à direita.

Um homem vive no lado leste de um rio e deseja vender um lobo, um bode e um repolho em uma vila no lado oeste. Porém, seu barco comporta apenas ele e mais um personagem. Além disso, o homem não pode deixar o lobo sozinho com o bode, porque o lobo comeria o bode, também não pode deixar o bode sozinho com o repolho, porque o bode comeria o repolho. Como o homem poderá atravessar todos em segurança?

Há uma abordagem óbvia para resolver este problema: escrever um algoritmo que teste diversas sequências de combinações de duplas e viagens de uma margem à outra do rio, até encontrar uma sequência válida, ou seja, que resolva o problema obedecendo às restrições. Tentar todas as possibilidades buscando uma solução para um problema é conhecido como "abordagem por força bruta". Geralmente não é uma boa estratégia, por ser ineficiente, consumindo muitos recursos como, por exemplo, tempo.

O que seria uma representação adequada para este problema? Uma vez que apenas os aspectos relevantes do problema precisam ser representados, todos os detalhes irrelevantes podem ser omitidos. Uma representação que deixa de fora detalhes do que está sendo representado é uma forma de *abstração*.

O uso de abstração é predominante na computação. Pense, a cor do barco é relevante? A largura do rio? O nome do homem? Não! A única informação relevante é onde está cada personagem/entidade desse contexto, a cada momento antes e após a execução de um passo/instrução da solução proposta para o problema. A localização coletiva de cada entidade, neste caso, refere-se ao *estado* do problema. Assim, o estado inicial do problema, que chamamos de α, pode ser representado conforme a Tabela 1.1.

Tabela 1.1: Estado inicial α.

Homem	Lobo	Bode	Repolho
0	0	0	0

Note que nesta representação criamos uma tabela em que o nome das colunas são as entidades relevantes no contexto do problema, e os valores das colunas são indicações sobre os respectivos estados de cada entidade.

Arbitrariamente, definimos que colunas com 0 (zero) indicam que a respectiva entidade está no lado leste do rio. Já colunas com 1 (um) indicam que a entidade está no lado oeste do rio. Por exemplo, se a primeira instrução do algoritmo fosse "homem atravessa com o bode de leste para oeste", a representação do novo estado do problema, gerado após a execução desta instrução, seria como ilustrado na Tabela 1.2. Neste novo estado apenas homem e bode estão no lado oeste e as localizações do lobo e do repolho não foram alteradas, pois a instrução executada não os impactou.

Tabela 1.2: Estado atual após a execução de uma instrução.

Homem	Lobo	Bode	Repolho
1	0	1	0

Com essa representação plenamente assimilada, podemos estabelecer que uma solução para este problema é a aplicação de um algoritmo que tenha como entrada o estado inicial α (Tabela 1.1), processe α com uma sequência de instruções logicamente ordenadas e que conduzam à saída do estado final β, ilustrado na Tabela 1.3.

Tabela 1.3: Estado final β.

Homem	Lobo	Bode	Repolho
1	1	1	1

Portanto, podemos generalizar nossos algoritmos em três etapas: (I) receber entradas; (II) processá-las e; (III) gerar saídas que resolvam o problema de acordo com as entradas processadas. Esse conceito está ilustrado na Figura 1.1.

Figura 1.1: Generalização do processo algorítmico. Fonte: Elaborado pelo autor.

É importante diferenciar a "representação" do "algoritmo" que será aplicado sobre ela. A tabela é apenas uma forma de representar o "mundo do problema" que queremos resolver, já o algoritmo é uma sequência de ações que serão aplicadas neste "mundo" de forma a deixá-lo no estado que desejamos, solucionando o problema.

VAMOS PRATICAR!

1) A partir do estado inicial, com os personagens na margem leste do rio, escreva uma sequência de passos que leve-os para a margem oeste do rio, considerando tudo o que foi descrito anteriormente. Lembre-se que queremos um algoritmo, então os passos devem ser logicamente ordenados, não ambíguos e suficientemente simples para serem entendidos e executados por outra pessoa. Faça da melhor forma que puder.

2) Reescreva seu algoritmo do exercício anterior, porém sem utilizar palavras. Adote que poderá usar apenas os símbolos H, L, B, R, 0, 1 (representando os personagens Homem, Lobo, Bode e Repolho e os lados leste e oeste, respectivamente) e → (para indicar que, naquela instrução, os personagens à esquerda da seta irão juntos de barco para o lado do lago indicado à direita da seta). Faça uma representação e atualize-a a cada novo estado gerado pela execução de uma instrução do algoritmo.

1.5. Linguagens de Programação

Até agora discutimos sobre algoritmos de forma genérica, porém, o foco da disciplina é o estudo de algoritmos que serão executados em computadores, ou seja, serão transformados em programas. Thomas Cormen, conceitua esse tipo de algoritmo:

> Também podemos considerar um algoritmo como uma ferramenta para resolver um problema computacional bem especificado. O enunciado do problema especifica em termos gerais a relação desejada entre entrada e saída. O algoritmo descreve um procedimento computacional específico para se conseguir essa relação entre entrada e saída. (CORMEN et al., 2012, p. 3).

Para que os algoritmos possam ser executados automaticamente é necessário que sejam escritos em linguagens compreensíveis aos computadores, que são as linguagens de programação. A codificação, também chamada de implementação, é a fase em que o programador escreve seus algoritmos em uma linguagem de programação.

Como escrito por Lira (2019, p. 29), "Linguagens de programação são conjuntos de símbolos e regras de sintaxe que permitem a construção de instruções que descrevem, de forma não ambígua, ações que podem ser entendidas e executadas por meio de computadores". Cada linguagem de programação define um padrão para a escrita de suas instruções, esses padrões têm motivações diversas, algumas linguagens favorecem a legibilidade para humanos, outras a concisão, ou o desempenho na execução etc.

Em nossa disciplina utilizaremos a linguagem de programação Python 3, que dentre as vantagens destacam-se: (I) sintaxe relativamente simples, o que pode ajudar no ensino e aprendizagem; (II) crescimento de popularidade no meio acadêmico e profissional, facilitando a busca de materiais de apoio, ajuda em fóruns e empregabilidade; (III) é portátil, ou seja, pode ser usada em diversos sistemas operacionais, como Windows, MacOS e Linux; (IV) é uma linguagem de código aberto e suportada por diversos ambientes gratuitos para programação.

Para entender como criar algoritmos que serão implementados em linguagens de programação, é útil começar com a apresentação dos principais componentes envolvidos nos sistemas computacionais, porém com objetos comuns, de forma lúdica, sem abordar diretamente um computador. Para isso, conheceremos o Computador Simplificado.

1.6. Computador Simplificado

Podemos simular procedimentos computacionais sem a necessidade de um computador, aprendendo conceitos básicos e usando objetos do dia a dia que, em seguida, serão mapeados para correspondentes de um sistema computacional eletrônico. O Computador Simplificado (que abreviaremos como CS) se encaixa neste contexto!

À direita da Figura 1.2 está um algoritmo com seis instruções, à esquerda há uma representação simplificada de um sistema computacional sem referência direta aos componentes de um computador, apenas objetos do cotidiano com função equivalente.

No CS, temos: (I) ao centro, uma pessoa chamada Megan, responsável por executar as instruções do algoritmo; (II) à esquerda, papéis descartáveis, cada um representando uma entrada de dados; (III) à direita, uma tela para exibir dados para outras pessoas; (IV) acima, uma lousa em que cada parte tem um nome, em nosso caso uma única letra, e pode guardar dados temporariamente, afinal pode ser facilmente apagada; (V) abaixo, um diário para registrar dados, de modo a armazená-los por longos períodos. As setas ao redor de Megan indicam o fluxo em que os dados podem trafegar.

ALGORITMO

Nº	Descrição
1	Leia um papel e guarde o valor em A.
2	Leia um papel e guarde o valor em C.
3	Some o conteúdo de A com C e guarde em Z.
4	Exiba o conteúdo de Z.
5	Registre no diário o conteúdo de Z dividido por 2.
6	Leia o registro do diário e guarde o valor em X.

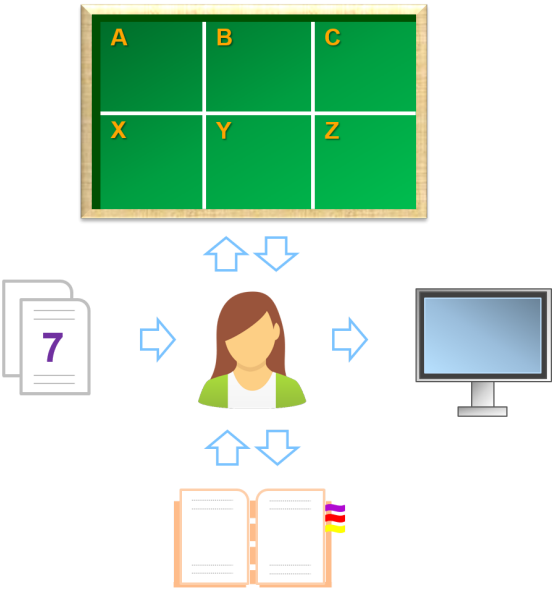

Figura 1.2: Algoritmo e CS (nota sobre direitos autorais[1]). Fonte: Elaborado pelo autor.

Suponha que o algoritmo comece a ser executado. A primeira instrução solicita que Megan leia um papel e guarde o valor na parte da lousa chamada A. Após a execução desta instrução, o estado será alterado conforme o ilustrado na Figura 1.3.

ALGORITMO

Nº	Descrição
1	Leia um papel e guarde o valor em A.
2	Leia um papel e guarde o valor em C.
3	Some o conteúdo de A com C e guarde em Z.
4	Exiba o conteúdo de Z.
5	Registre no diário o conteúdo de Z dividido por 2.
6	Leia o registro do diário e guarde o valor em X.

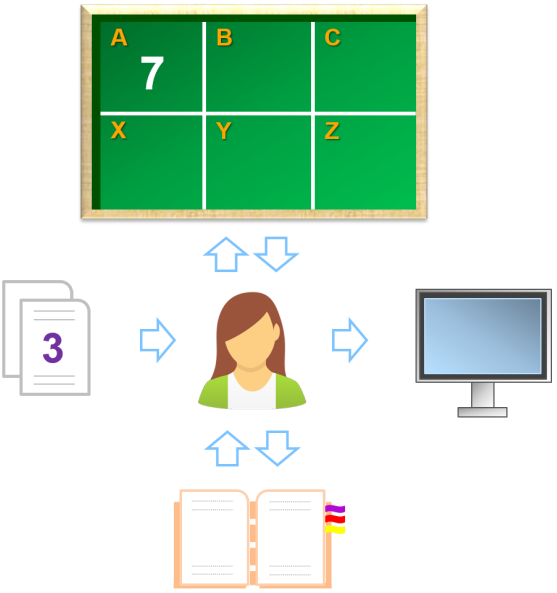

Figura 1.3: CS após a execução da 1ª instrução. Fonte: Elaborado pelo autor.

A segunda instrução solicita que Megan leia outro papel e guarde o valor na parte C da lousa. Se a instrução indicasse que o valor lido deveria ser guardado em A, o valor prévio em A seria sobrescrito, afinal não pode haver dois valores ocupando o

[1] A figura utilizada para representar Megan no Computador Simplificado é de autoria de Fredy Sujono (https://www.iconfinder.com/freud) e está autorizada para uso comercial mediante indicação de créditos.

mesmo espaço ao mesmo tempo. Após a execução desta instrução, o estado será alterado conforme o ilustrado na Figura 1.4.

ALGORITMO

Nº	Descrição
1	Leia um papel e guarde o valor em A.
2	Leia um papel e guarde o valor em C.
3	Some o conteúdo de A com C e guarde em Z.
4	Exiba o conteúdo de Z.
5	Registre no diário o conteúdo de Z dividido por 2.
6	Leia o registro do diário e guarde o valor em X.

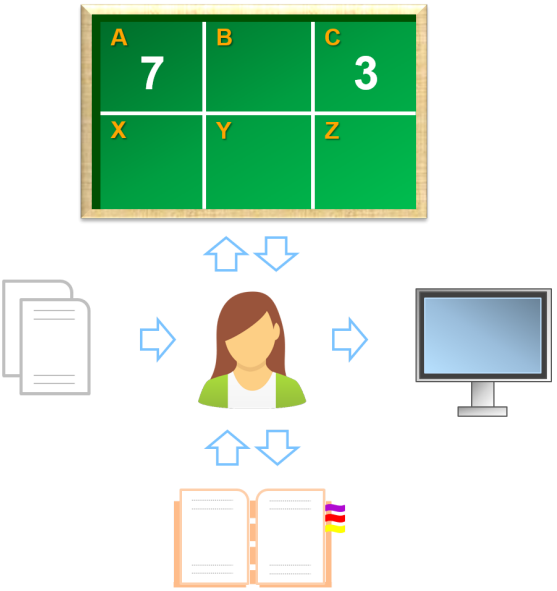

Figura 1.4: CS após a execução da 2ª instrução. Fonte: Elaborado pelo autor.

Após a execução da quarta instrução (propositalmente pulamos a terceira), o estado do CS estará conforme a Figura 1.5. É importante observar que Megan pode tanto escrever na lousa quanto consultá-la, assim como no diário. A exibição de dados na tela corresponde ao que anteriormente denominamos como "saída" do algoritmo.

ALGORITMO

Nº	Descrição
1	Leia um papel e guarde o valor em A.
2	Leia um papel e guarde o valor em C.
3	Some o conteúdo de A com C e guarde em Z.
4	Exiba o conteúdo de Z.
5	Registre no diário o conteúdo de Z dividido por 2.
6	Leia o registro do diário e guarde o valor em X.

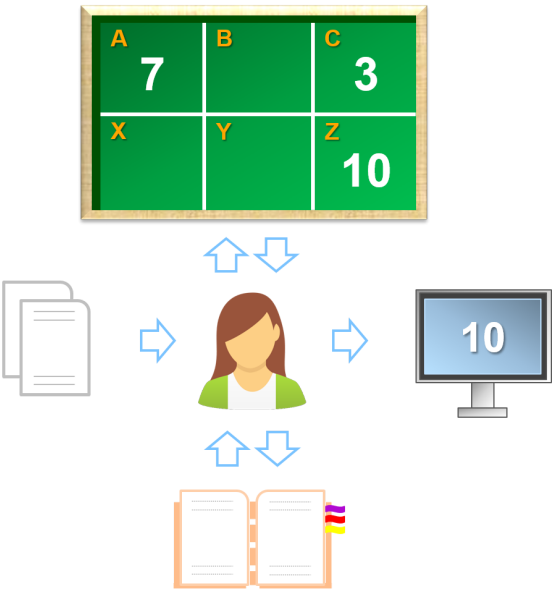

Figura 1.5: CS após a execução da 4ª instrução. Fonte: Elaborado pelo autor.

Na Figura 1.6 observamos o que ocorre após a execução da quinta instrução do algoritmo, que registra no diário o conteúdo de Z dividido por 2. Registrar dados

corresponde a guardá-los em um local em que não sejam apagados tão facilmente, diferentemente daqueles que estão na lousa, uma área de trabalho temporária.

ALGORITMO

Nº	Descrição
1	Leia um papel e guarde o valor em A.
2	Leia um papel e guarde o valor em C.
3	Some o conteúdo de A com C e guarde em Z.
4	Exiba o conteúdo de Z.
5	Registre no diário o conteúdo de Z dividido por 2.
6	Leia o registro do diário e guarde o valor em X.

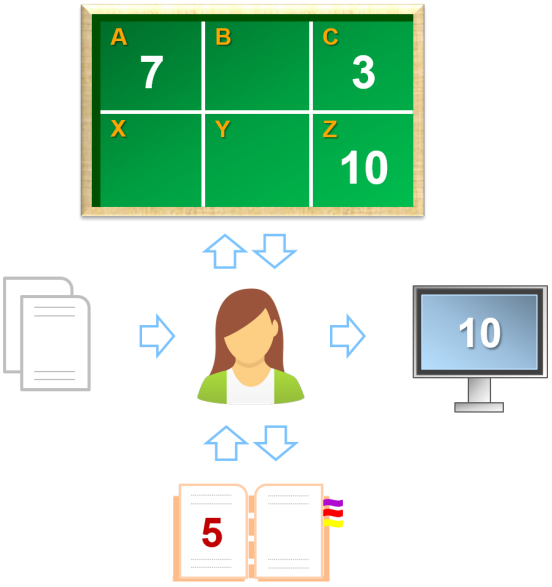

Figura 1.6: CS após a execução da 5ª instrução. Fonte: Elaborado pelo autor.

Com reconhecimento da importância de cada parte do CS, a Figura 1.7 ilustra a correspondência de seus itens em relação aos componentes de um computador real.

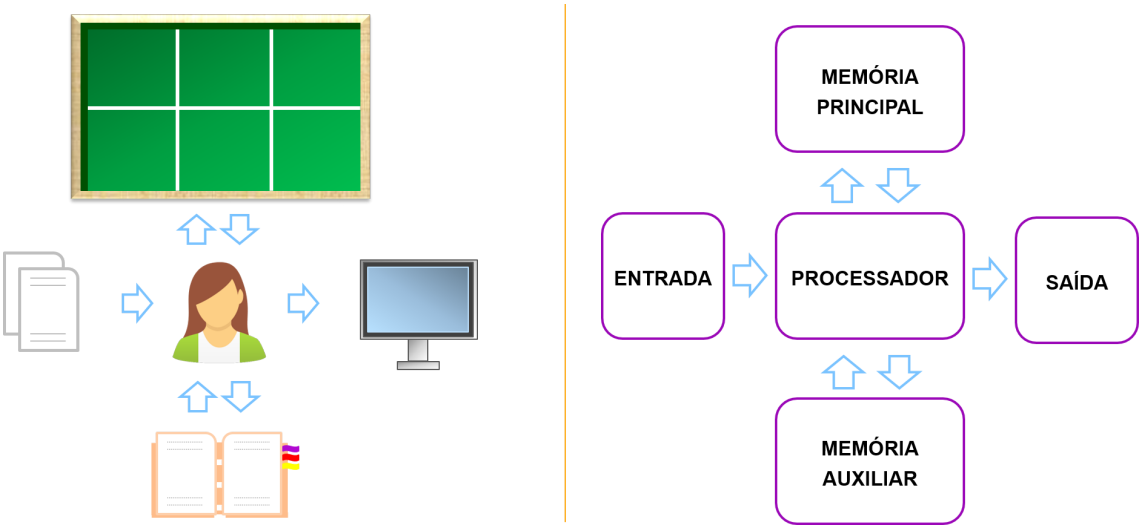

Figura 1.7: Correspondência entre o CS e um computador. Fonte: Elaborado pelo autor.

O mapeamento é dado assim: (a) Megan é o processador, pois é o componente que executa as instruções do algoritmo; (b) os papéis descartáveis são dados fornecidos como entrada, geralmente inseridos pelo usuário por meio de um teclado, mouse ou tela sensível ao toque; (c) a tela é a saída, que corresponde a um monitor ou impressora, por exemplo; (d) a lousa simboliza a memória principal, conhecida como Memória RAM, que serve para guardar dados temporariamente, pois ao cortar o fluxo de energia do

computador os dados são apagados e; (e) o diário é a memória auxiliar, que serve para persistir os dados, ou seja, mantê-los registrados para que possam ser consultados futuramente, como HDDs, SSDs, cartões de memória, *pendrives* etc.

Você pode estar se perguntando: "qual seria a equivalência do algoritmo em um sistema computacional real?". Resposta: ao *código-fonte* de um programa escrito em uma linguagem de programação. Aprender essa linguagem será um de nossos objetivos!

VAMOS LER!

Você sabia que existe um site dedicado a computação desplugada? Sim! Um site dedicado a propagação desse método para ensinar e aprender conceitos de computação e pensamento computacional sem o uso de computadores. Veja atividades e desafios interessantes sobre diversos assuntos: como os números binários funcionam? Como saber se um ano é bissexto? Como verificar se um número é par ou ímpar? É possível adivinhar o número que alguém está pensando com pouquíssimas tentativas?
Para quem ficou interessado, o endereço é: https://csunplugged.org/en/

Bibliografia e referências

BROOKSHEAR, J. G. **Ciência da Computação: uma abordagem abrangente**. 7 ed. Porto Alegre: Bookman, 2008.

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

DIERBACH, C. **Introduction to Computer Science Using Python: A Computational Problem-Solving Focus**. 1 ed. New York: Wiley, 2012.

KNUTH, D. E. **The art of computer programming**. 3 ed. Massachusetts: Addison-Wesley, 1997. ISBN: 978-0-201-89683-1.

LIRA, L. N. **Instrumentos de apoio ao ensino e aprendizagem de algoritmos e programação de computadores: implicações no desempenho discente em instituição de educação profissional**. 197f. Dissertação (Mestrado Profissional em Gestão e Desenvolvimento da Educação Profissional). Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2019.

PEREIRA, S. L. **Algoritmos e lógica de programação em C: uma abordagem didática**. 1 ed. São Paulo: Érica, 2010. ISBN: 978-85-365-0327-1.

ZIVIANI, N. **Projeto de algoritmos com implementações em Pascal e C**. 4 ed. São Paulo: Pioneira, 1999.