Desenvolvimento de Sistemas

Lógica de programação

A questão mais importante no início dos estudos em programação é a **lógica**. Os estudos de lógica remetem à Antiguidade, com registros e citações de Aristóteles (384 a.C.) à lógica matemática. A lógica também está presente na filosofia, quando o ser humano reflete sobre o próprio pensamento.

Entender exatamente como desmembrar um problema, organizar um pensamento e ordenar passos para obter algum resultado (o qual leva aos algoritmos) é necessário para chegar a uma solução computacional desejada e para entregar exatamente o que foi pedido.

Ainda nesse contexto, existe uma série de caminhos que precisam ser percorridos para entender e estruturar os algoritmos. A lógica, antes da programação, entra em pauta e precisa ser tratada, assim como o **pensamento computacional**, o qual possibilita entender o que é um algoritmo. Contudo, você verá mais detalhes no decorrer deste conteúdo.

Lógica analítica

A lógica é uma disciplina originalmente da filosofia, que se desdobra e tem relevância em diferentes áreas do conhecimento, como matemática, computação etc. Em função da ocorrência em diferentes áreas, são encontradas diversas explicações sobre o que é lógica, dependendo da área de estudo de quem está a definindo. Porém, buscando encontrar um ponto em comum, pode-se resumir a lógica como a ação de **analisar um problema e julgar se ele é verdadeiro ou falso**. Observe um exemplo:

- Todos os dias o sol nasce e se põe.
- Hoje é um dia.
- Hoje o sol vai nascer e se pôr.

O que se observa nas frases é um raciocínio lógico. As duas primeiras frases apresentam os fatos, e a terceira frase é a conclusão baseada neles. Só se chega a uma conclusão porque as duas frases anteriores embasam a afirmação. Entretanto, note que os fatos podem ser interpretados de forma errada. Por exemplo:

- Ontem choveu.
- Hoje está chovendo.
- Amanhã vai chover.

Chover não é uma constância em todos os dias do ano, e sim um evento que pode acontecer. Logo, não se pode afirmar que de fato amanhã vai chover. Pelas frases, pode-se considerar que há mais chances de amanhã chover também, mas não se pode afirmar que vai chover.

Olhando para o primeiro caso, a conclusão está correta porque as duas regras (“Todos os dias o sol nasce e se põe” e “Hoje é um dia”) não se contradizem, da mesma maneira que a frase “Hoje o sol vai nascer e se pôr” também não contradiz as outras duas.

Além disso, as frases do primeiro caso respeitam as regras de funcionamento ao observar o mundo – nesse caso, o mundo real –, pois todos os dias o sol vai nascer e se pôr em função de como o planeta Terra se movimenta. Em resumo, as duas primeiras frases embasam as informações para a conclusão da última.

Nem sempre você estará observando o mundo real quando está verificando alguma coisa. Veja um exemplo:

- **Os bruxos utilizam uma varinha para fazer feitiços.**

Esse fato é verdadeiro no mundo ficcional de Harry Potter, mas é falso no mundo real. É sempre importante saber o contexto que está sendo analisado, porque pode ser um contexto que não necessariamente respeita as regras do mundo real. O importante é fazer sentido no contexto observado.

Perceba que os fatos apresentados estão sendo verificados, buscando compreender se são verdadeiros ou não. Não faz parte do que está sendo avaliado o motivo de o sol nascer e se pôr. Isso não é o que se faz todos os dias quando se tem contato com alguma fala, alguma informação etc. A lógica faz parte da vida das pessoas cotidianamente, nas avaliações e nos processos de decisão, por exemplo. Mesmo sem perceber, a lógica está acontecendo.

Observe o desafio a seguir, utilizando a lógica para responder a ele. Providencie um papel e uma caneta para fazer anotações, se necessário.

Desafio:

Duas mães levaram suas filhas para comprar chocolates. Cada filha e cada mãe pegam um chocolate. Chegando em casa, há apenas três chocolates, e nenhum deles foi comido durante o caminho. Então, por que há apenas três chocolates?

Resposta:

Foram ao mercado três pessoas: a avó, a mãe e a filha.

Você conseguiu perceber como é importante ler as informações com atenção, separar informações de deduções e considerar que um mesmo elemento pode ter mais de uma informação?

Em nenhum momento foi estabelecido que existiam quatro pessoas indo comprar chocolates; é uma dedução errada feita pelo leitor. A informação real é que “duas mães levaram suas filhas”, mas sem especificar a quantidade de pessoas. Outra questão importante é que, muitas vezes, não se considera que um mesmo objeto de análise pode ter mais de uma característica – no caso, uma mesma pessoa ser mãe e filha ao mesmo tempo –, o que não significa desobedecer às regras estabelecidas.

Até no processo de falar algo a lógica está sendo aplicada. É preciso escolher as palavras que serão ditas, a posição de cada uma delas etc. Isso acontece porque cada mudança altera o sentido do que está sendo falado, e a ideia é comunicar exatamente o que está sendo pensado. É aí que entra a lógica. Ela é necessária para elaborar ideias que façam sentido, para organizar a maneira de se expressar e, ainda mais importante, para analisar informações.

Desafio:

Você é professor e cuida de 12 crianças que estudam em uma creche no turno da manhã. Durante o período da tarde, 8 crianças voltam para casa, e mais 7 crianças chegam à escola. Qual é o tamanho do sapato do professor?

Resposta:

Você é o professor. Qual é o tamanho do seu sapato?

Essa é uma situação perfeita para perceber uma questão que é aparentemente óbvia, mas muito importante: ao analisar alguma coisa, é preciso separar as informações que são relevantes das que não são. Nem todas as informações são importantes para o que você está avaliando.

No exemplo citado, há uma série de informações sobre a quantidade de crianças que entraram e saíram da creche. Por mais que números sejam importantes e que se queira saber, no fim, um valor numérico, tais informações não ajudam a chegar mais próximo do valor pretendido.

Os três pilares da lógica: dedução, indução e abdução

A lógica foi pensada e conceituada por Aristóteles no século V a.C., para justamente se contrapor aos grandes filósofos da época, conhecidos por dominarem a arte da retórica e do discurso e também por cobrarem para desenvolver tais pensamentos. Aristóteles preferiu então desenvolver a lógica para que o acesso fosse permitido a todos, sem nem ao menos imaginar que influenciaria até hoje o pensamento e a formação de pensamentos de lógica, até mesmo na tecnologia.

Confira agora os três pilares da lógica:

Dedução

É utilizada para determinar uma conclusão, de maneira lógica e exata. Por exemplo:

- Todo animal é mortal.
- Todo homem é animal.
- **Logo, todo homem é mortal.**

Indução

Observa o mundo como ele é, do geral, e somente a partir daí se pode desenvolver uma lógica ou chegar a uma conclusão (verdadeira ou falsa). Por exemplo:

- Eu tenho visto muitos gatos, e todos eles eram pretos. Portanto, todos os gatos são pretos.

A premissa está correta, porém a conclusão é falsa. É de conhecimento geral que não são todos os gatos que são pretos.

- A grama ficou molhada todas as vezes em que choveu. Então, se chover amanhã, a grama ficará molhada.

Em resumo, o raciocínio indutivo parte do particular para o geral, seguindo determinado ciclo: observação e/ou experimentação; formulação de hipóteses explicativas; aplicação de teorias, enunciados, leis universais etc. Assim, a dedução e a indução podem ser resumidas da seguinte forma:

Dedução	Indução
◆ Ocorre do geral para o particular ◆ Baseia-se em premissas verdadeiras, que garantem uma conclusão verdadeira ◆ Não amplia o conhecimento	◆ Ocorre do particular para o geral ◆ Baseia-se em premissas verdadeiras, que não garantem uma conclusão verdadeira ◆ Amplia o conhecimento

Abdução

É a terceira via da lógica. O raciocínio abdutivo atua entre os dois extremos anteriores: o que sempre busca 100% de confiabilidade e o que busca 100% de validez. Esse meio-termo utiliza características de ambos para concluir a melhor explicação de algo. Cabe ressaltar ainda que melhor explicação é diferente de maior probabilidade, e a abdução, por sua vez, tem caráter explicativo e intuitivo, procurando concluir a melhor explicação e também utilizando seu conhecimento de fundo (repertório de conhecimento), e não a melhor probabilidade matemática.

Para finalizar, relembre o exemplo da chuva e da grama usado na indução e o traga para os três pilares novamente:

1. **Dedução**

- Quando chove, a grama fica molhada.
- Choveu hoje.
- Portanto, a grama está molhada.

2. Indução

- A grama ficou molhada todas as vezes em que choveu.
- Então, se chover amanhã, a grama ficará molhada.

3. Abdução

- Quando chove, a grama fica molhada.
- A grama está molhada, então pode ter chovido.

Lógica booliana

Você viu até agora como analisar as informações para determinar se elas são verdadeiras ou falsas, o que é a base da lógica booliana (ou booleana).

George Boole foi um matemático que criou um sistema numérico de valores finitos para representar a lógica em equações matemáticas. Por serem finitos, só existem dois valores possíveis: um e zero. O zero significa **falso**, que um aparelho está desligado. O um representa a **verdade**, um mecanismo ligado. Esses dois valores são oposições um para o outro e, na lógica booliana, são os dois únicos valores possíveis.

A lógica booliana é utilizada em diversas áreas do conhecimento, como linguagens de programação, inteligência artificial, circuitos lógicos, bancos de dados relacionais, sistemas distribuídos etc.

Olhando esse sistema, ele pode não parecer muito útil, porque geralmente há variações das coisas. Por exemplo, se você perguntar para alguém: “Está frio?”, não receberá necessariamente apenas um “sim” ou um “não” como resposta, porque as pessoas podem julgar a temperatura com valores intermediários. A resposta pode ser, por exemplo: “Está ameno”.

“Está ameno” quer dizer que a temperatura não está nem fria, nem quente, então não se pode julgar a informação como zero ou um utilizando apenas a linguagem. Todavia, ao colocar as informações dentro de um sistema de regras, aquelas podem ser julgadas como verdadeiras ou falsas, logo: um ou zero. Assim, podem-se estabelecer os graus que indicam se a temperatura está fria.

Voltando para o sistema booliano, ele se mostrou muito eficiente para computadores. Os computadores não se comunicam da mesma maneira que os seres humanos; na verdade, tais equipamentos têm uma comunicação muito restrita, qual seja de desligado e ligado.

Entradas e **saídas** são conceitos recorrentes no mundo do desenvolvimento. **Entrada** é a informação recebida pelo *software*, a qual pode ser fornecida pelo usuário por meio de cliques no *mouse*, por meio de um texto. É uma informação que você sabe que vai receber, mas ainda não sabe como ela é. Já a **saída** é a resposta que retorna ao usuário depois de processar a informação de entrada. Essa resposta pode ser em formato de texto, de imagem etc. O importante desses conceitos é receber informações do usuário, trabalhá-las e depois retornar para o usuário alguma coisa.

Para ajudar a entender melhor, você trabalhará a seguir um problema maior, resolvendo-a à medida que progredir no próximo assunto. Sinta-se à vontade para tentar resolver o problema antes de continuar.

Segue o problema:

Eduardo tem intolerância à lactose e está passando mal. Para saber se Eduardo precisa tomar um remédio, é necessário saber se ele comeu alguma coisa e, se sim, o que comeu.

Existem cinco pessoas que estavam perto de Eduardo e podem ter visto algo. Elas podem estar mentindo ou dizendo a verdade, mas não as duas coisas ao mesmo tempo. Veja o que cada pessoa disse:

- Gabriel: “Eduardo tomou leite”.
- Jerônimo: “Eduardo não tomou leite”.
- Matheus: “Eduardo comeu um pedaço de bolo”.
- Felipe: “Eduardo não tomou leite e não comeu um pedaço de bolo”.
- Nicolas: “Felipe é mentiroso ou Matheus fala a verdade”.

Depois de conversar mais um pouco, Gabriel disse:

- “Matheus fala a verdade ou Nicolas é mentiroso”.

Agora tente responder:

- Eduardo comeu e/ou tomou alguma coisa?
- Se sim, ele comeu e/ou tomou o quê?

Geralmente, ao se deparar com problemas que precisam ser analisados, estes não são pequenas frases, e sim, em geral, conjuntos de frases (proposições) complexas que podem até mesmo mudar o sentido ou a veracidade de uma frase anterior.

O caso do Gabriel é um exemplo, e por isso você precisar utilizar **operadores boolianos**, também conhecidos como conectivos, para resolver a questão.

Existem dois tipos principais de operadores:

- **Operador unário**: aplicado sobre apenas uma sentença. Um exemplo é o operador de negação.
- **Operador binário**: aplicado sobre duas sentenças. Um exemplo é o operador “e” (conjunção).

Observe os dois tipos citados analisando o problema apresentado, no qual as pessoas disseram as seguintes frases:

- Eduardo tomou leite.
- Eduardo não tomou leite.
- Eduardo comeu um pedaço de bolo.

As três frases são simples e não podem ser separadas em frases menores. Elas ajudam a construir a situação, mas não se relacionam diretamente com outras sentenças. Relendo o problema, você verá que as frases estão ligadas e que os **operadores lógicos** as conectam. Você trabalhará com a segunda afirmação do Gabriel:

- Matheus fala a verdade ou Nicolas é mentiroso.

Essa frase pode ser separada em duas outras frases:

- Matheus fala a verdade.
- Nicolas é mentiroso.

Por mais que possam ser separadas, as frases têm uma ligação entre si e devem ser tratadas como uma única: o “**ou**”.

Disjunção: “ou”

O “ou” é um operador binário, ou seja, precisa de duas informações para conseguir ser utilizado. Quando tal operador é utilizado, desde que uma das informações seja verdadeira (1), ele resultará em verdadeiro (1). Independentemente

se a primeira ou a segunda informação for verdadeira, o retorno será verdadeiro. Se ambas forem falsas (0), o resultado será falso (0).

Relembre o exemplo, na fala do Nicolas:

- Felipe é mentiroso ou Matheus fala a verdade.

Note que são duas informações, que podem ser separadas assim:

- Felipe é mentiroso
- Matheus fala a verdade.

Porém, há a palavra “ou” entre as duas informações, e ambas precisam ser analisadas em conjunto. No enunciado do problema, foi dito que cada pessoa só pode estar falando a verdade ou mentindo, mas isso é o resultado final. No caso da palavra “ou”, está tudo bem se, na sentença, uma informação é falsa e outra é verdadeira. Desde que uma das informações seja verdadeira, o resultado final sobre a fala da pessoa será verdadeiro. Por isso, para julgar se Nicolas está falando a verdade, só uma das frases precisa ser verdadeira obrigatoriamente.

Conjunção: “e”

O “e” é um operador binário e, assim como o “ou”, precisa de duas informações para ser utilizado. O “e” é um operador exigente, porque só retornará verdadeiro (1) se ambas as informações forem verdadeiras (1).

Relembre o exemplo, na frase do Felipe:

- Eduardo não tomou leite e não comeu um pedaço de bolo.

Novamente, há uma grande frase que pode ser separada em duas informações distintas:

- Eduardo não tomou leite.
- Eduardo não comeu um pedaço de bolo.

Para saber se a informação dada por Felipe é verdadeira (1), primeiro é preciso analisar cada uma das frases, porque se uma delas for falsa (0), independentemente de qual seja, Felipe está mentindo (0). Se ambas forem verdadeiras, Felipe está falando a verdade (1).

Negação

Diferentemente do “e” e do “ou”, a negação trabalha baseada em uma única informação, caracterizando-se assim com operador unário. Se a informação é verdadeira (1), torna-se falsa (0); se era falsa (0), torna-se verdadeira (1).

Relembre a frase do Jerônimo:

- Eduardo não tomou leite.

Essa frase é o contrário da do Gabriel, pois nega que Eduardo tenha consumido leite. Pode-se pensar na frase como: “Não é verdade que Eduardo tomou leite”.

Seguindo para a solução do problema de Eduardo, se **Gabriel está falando a verdade**, então há duas frases verdadeiras. Assim, considerando que Gabriel está falando a verdade, se Eduardo tomou leite, então, obrigatoriamente, Jerônimo e Felipe estão mentindo. Jerônimo mente porque diz o contrário do que Gabriel disse, e Felipe mente porque ambas as informações precisam ser verdadeiras no caso de “e”. Não se sabe ainda se Eduardo comeu o bolo, mas já se considera que Felipe é mentiroso.

Já se sabe, inclusive, que Nicolas está falando a verdade, pois ele diz que Felipe mente “ou” Matheus fala a verdade. Na hipótese de Gabriel estar certo, Felipe mente, e então existem duas opções: considerar Matheus mentiroso ou considerar Matheus verdadeiro. Veja:

Se Matheus fala a verdade	Se Matheus mente
◆ Verdades: Gabriel, Nicolas e Matheus ◆ Mentiras: Jerônimo e Felipe	◆ Verdades: Gabriel e Nicolas ◆ Mentiras: Gabriel, Jerônimo, Felipe e Matheus
A resposta é que Eduardo tomou leite e comeu um pedaço de bolo. Não há nenhum conflito entre as frases.	Há uma contradição: Gabriel, que é considerado verdadeiro, estaria mentindo em sua segunda frase, pois nem Matheus estaria dizendo a verdade, nem Nicolas seria mentiroso. Pela regra do “ou”, a informação da frase seria falsa.

Levando em consideração agora que **Gabriel é mentiroso**, deve-se considerar que ambas as frases dele são falsas. Como a frase do Jerônimo é o contrário da do Gabriel, considera-se que Jerônimo fala a verdade. Até aqui, sabe-se que Eduardo tomou leite, então a primeira informação que Felipe disse é verdadeira (não quer dizer que ele está falando a verdade). Confira as hipóteses:

Se Felipe falou a verdade	Se Felipe mentiu
Então Eduardo não comeu bolo, e obrigatoriamente Matheus está mentindo. Nesse caso, logicamente Nicolas está mentindo, porque Matheus mentiu e Felipe falou a verdade. Porém, se Nicolas está mentindo, então Gabriel está falando a verdade porque Nicolas é mentiroso – mas já se definiu que Gabriel é mentiroso. As informações não condizem uma com a outra.	Então Eduardo comeu um pedaço de bolo, porque, já que se está assumindo que Gabriel mente, a primeira parte é verdadeira (“Eduardo não tomou leite”). O que impede o “e” de ser verdadeiro para que Felipe minta é a segunda parte (“Não comeu bolo”). Logo, Matheus está certo. Nicolas falou a verdade porque Matheus falou a verdade e Felipe mentiu. O problema é que, se Matheus fala a verdade, Gabriel também está falando a verdade, e não mentindo. As informações entram novamente em conflito.
◆ Mentiras: Gabriel, Matheus e Nicolas ◆ Verdades: Gabriel, Jerônimo e Felipe	◆ Mentiras: Felipe e Gabriel ◆ Verdades: Gabriel, Jerônimo, Matheus e Nicolas

A única situação que não gera um conflito de informação é quando **Gabriel fala a verdade**. Como Gabriel e Jerônimo tinham dito frases opostas, por meio dos desdobramentos de cada uma delas, conseguiu-se descobrir qual é o caminho que não viola nenhuma outra informação.

Desafio

Algumas pessoas estão trocando figurinhas repetidas do álbum de futebol do ano. Maria e José estão negociando. Depois da negociação, José diz para Maria que, se ela der de graça para ele mais uma figurinha, ele terá o dobro dela. Maria diz que, se José der uma figurinha de graça, eles terão a mesma quantidade de figurinhas. Quantas figurinhas cada um tem?

Resposta

Maria tem 5 figurinhas e José tem 7. Usa-se aqui o operador “e” sem perceber. O objetivo é chegar a dois valores que respeitem **ambas** as regras:

• Se retirar 1 do valor B, o valor A será o dobro de B.
• Se retirar 1 do valor A, o valor A e o valor B serão iguais.

Por exemplo, o valor B é 3 e o valor A é 3. Logo, se for retirado 1 de B, o valor A será 4, que é o dobro do valor A em questão (2). Isso significa que a primeira regra está sendo respeitada. Entretanto, se for retirado 1 de A, o valor fica 2, e o valor B fica 4, o que não respeita a segunda regra. Logo, os valores não estão corretos.

Algoritmos

Sempre que se deseja que o computador faça alguma coisa específica, devem-se fornecer para ele instruções de **como fazer**. Com isso em mente, o objetivo é escrever um **programa** explicando passo a passo quais são as tarefas que o computador tem que fazer e como elas devem ser feitas, e é justamente nesta etapa que a palavra "**algoritmo**" toma forma.

Em resumo, algoritmos são uma série de instruções usadas por computadores visando a resolver dado problema para que se chegue a uma solução.

Claro que as instruções têm que ser extremamente bem pensadas e definidas (e até refinadas) para que se possa chegar a uma maneira satisfatória de resolver os problemas que foram dados.

Existem, assim, diversas formas de escrever um **algoritmo**. Algumas são mais informais, algumas formais e matemáticas e outras até mesmo por meio de gráficos. Por exemplo, se for preciso conectar um videogame em uma televisão, usando cabos e energia elétrica, é necessária uma série de passos para solucionar esse problema, isto é, é necessário utilizar um **algoritmo**.

O desenvolvimento de todo e qualquer algoritmo consiste em desenvolver um planejamento. Assim que se tem o algoritmo em mãos, é possível traduzi-lo para a linguagem de programação. Contudo, antes, veja quatro passos para resolver os problemas de forma mais simples:

Passo 1: descrever o problema

Querendo ou não, o primeiro passo é um dos mais complicados. Você lidará aqui com duas palavras muito importantes: “cliente” e “desenvolvedor” (você). O **cliente** é sempre o responsável por fornecer a descrição de um problema (algoritmo), o que, na maioria das vezes, é a parte mais complicada do processo. Geralmente, são encontrados alguns problemas:

1. A descrição está ambígua e não diz exatamente o que é necessário.
2. A descrição está incompleta.
3. A descrição não condiz com alguns passos que foram pedidos.

Logo, é justamente o papel do **desenvolvedor** compreender e analisar tais problemas e remediar a situação com o **cliente** antes do início do desenvolvimento. Veja um exemplo bem simples de alguns problemas em uma confeitaria:

Cenário: você é dono de uma confeitaria famosa por produzir bolos. Durante esta semana, alguns clientes fizeram encomendas:

- **Cliente 1**: “Eu quero um bolo”.
- **Cliente 2**: “Eu quero um bolo de chocolate”.
- **Cliente 3**: “Eu quero um bolo de chocolate, mas sem cacau”.
- **Cliente 4**: “Eu quero um bolo de chocolate, com cobertura de chantili e que sirva 20 pessoas”.

Percebeu a grande diferença de informações que são necessárias para que de fato sejam entendidos os problemas e a solução que os clientes precisam? Aliás, não esqueça que você está apenas no primeiro passo.

Passo 2: analisar o problema

A proposta deste passo é justamente determinar os pontos inicial e final para solucionar o problema. Se for feito um bom trabalho no passo 1, será muito mais fácil chegar a uma análise com a maior quantidade de informações e insumos.

- Que informações estão disponíveis?
- Onde elas estão?
- Existe alguma regra para manipulá-las?

Voltando à confeitaria:

- Qual é o sabor do bolo? Ele precisa de cobertura? Qual é a quantidade de pessoas? Qual é a data de entrega?
- O contato é feito apenas com o cliente? Ele é feito por *e-mail*, servidor, sistema da loja?
- O cliente comentou se tem alguma alergia? O bolo será entregue ou o cliente vai retirá-lo na loja?

Como pôde ser visto, são diversas **variáveis** e **informações** que deverão ser guardadas e pensadas conforme você for construindo o seu algoritmo apenas neste ponto inicial. É sempre melhor pensar nessas condições **no início do processo** para que, **no final ou na entrega, não existam alterações vitais que possam atrasar o pedido dos clientes**.

Passo 3: desenvolver um algoritmo de alto nível

Chegou o momento de repensar um pouco. Você já entendeu que um algoritmo é um plano para resolver problemas, porém este precisa ser “macro” para poder lidar com os problemas menores de uma forma um pouco mais simples durante o desenvolvimento. Observe o exemplo:

Problema: você precisa fazer um bolo com recheio de morango e chocolate, para 20 pessoas, e entregá-lo amanhã pela manhã.

Algoritmo de alto nível:

- Separar os ingredientes
- Confeccionar o bolo
- Embalar o bolo para entrega

Esse algoritmo é satisfatório para ser usado diariamente, porém ainda faltam alguns detalhes a serem implementados para que ele se torne de fato uma solução não só para problemas simples. Por exemplo:

Detalhes:

- Qual é o tipo de chocolate solicitado pelo cliente: amargo, meio amargo, ao leite, branco?
- Que horas da manhã o cliente buscará o bolo? (O horário pode impactar o processo de confecção.)
- Como o cliente levará o bolo: de carro, moto, bicicleta?
- Qual é a melhor forma de embalar o produto para que chegue ao destino com qualidade?

Esses detalhes se relacionam diretamente com o próximo passo, pois é nele que você aprenderá a **refinar** os seus algoritmos para buscar soluções mais práticas e completas para os clientes e até mesmo soluções que nem foram pensadas durante o pedido.

Passo 4: refinar o algoritmo

O algoritmo de alto nível mostra os passos mais importantes e **essenciais** para resolver o problema que foi dado. Agora, o que você precisa é justamente colocar detalhes que supram outras necessidades e complementem todo o seu **algoritmo**.

Considere **o que** será implementado no algoritmo e quanto o **cliente** realmente precisará desses detalhes. É importante ressaltar que isso remonta ao passo 1, no qual a conversa entre cliente e desenvolvedor deve ser sempre bem detalhada e com um contato próximo.

Quando o objetivo é desenvolver algoritmos que serão interpretados pelos computadores, é necessário considerar as capacidades do sistema e prover detalhes suficientes para que o planejamento permita que o desenvolvedor se sente na frente da **interface de desenvolvimento** e não tenha que retornar aos passos anteriores.

Quando estiver com dúvidas, lembre-se de que é preferível haver muitos detalhes a poucos.

Que tal um desafio para explorar os quatro passos que você aprendeu, transformando um algoritmo com a melhor solução possível?

Um cliente trouxe uma solicitação de desenvolvimento de um sistema que deve servir para registrar todas as atividades do dia a dia dele. O cliente informou que esse sistema precisa ter uma agenda, na qual o usuário pode registrar o título da atividade, além de um alarme a cada **30 minutos**.

Com base nas informações citadas, pense nos **quatro passos** anteriores e comece a listar tudo que o sistema deve conter.

Leia atentamente a descrição do problema e analise as funcionalidades principais. Elabore um algoritmo de alto nível e depois o refine, apresentando melhorias e detalhes que não estão na descrição.

Resolvendo problemas

Partindo da lógica analítica, cotidiana ou mesmo booliana, pode-se olhar para um problema mais específico e começar a imaginar uma solução computacional, criando um algoritmo. Durante esse processo, no entanto, é importante atentar a alguns pontos. São eles:

Compreender o que é pedido

Tanto nos estudos quanto no trabalho, dedicar um tempo para interpretar o que é realmente solicitado é um passo, apesar de óbvio, essencial e muitas vezes negligenciado, por causa da pressa ou da ansiedade de quem está lendo. Ao resolver um problema, analise exatamente o que é esperado de resultado e quais são as restrições e as condições envolvidas no problema.

Para praticar, elabore algoritmos nos exercícios propostos. A tendência é ler brevemente o enunciado, escrever algum código e testar se a resposta condiz com a esperada pela atividade. Qualquer erro nessa resposta acaba se tornando algo obscuro, indecifrável; por isso, é importante entender os detalhes envolvidos no problema e na resposta, os passos que estão sendo tomados e o motivo de estes serem necessários.

O mesmo vale para a rotina de trabalho: os requisitos dos clientes precisarão ser interpretados com cuidado, considerando restrições e regras de negócio presentes na organização e no sistema. Programar sem a prévia atenção a detalhes é a certeza de ter retrabalho no futuro, gerando correções e adaptações (muitas vezes profundas) no código.

Realizar deduções na construção do programa

Você já viu neste conteúdo os conceitos de dedução e indução. Direta ou indiretamente, deduções serão aplicadas na resolução de algoritmos, formando parte do pensamento presente na lógica de programação. Veja o exemplo:

1. Se hoje fizer sol, nós iremos à praia.
2. Faz sol hoje.

Pode-se deduzir, então, que nós iremos à praia, pois a condição (“fazer sol”) parece satisfeita. Em programação, perguntas assim estarão presentes, mas, em vez de sentenças abstratas, como “faz sol”, serão utilizados valores definidos, como números ou palavras. Veja no exemplo:

1. O número é par se é divisível por 2.
2. 71.

Pode-se concluir que 71 não é um número par, pois não é divisível por 2.

Um último exemplo aplica conectores lógicos:

1. O livro está na cama ou na estante.
2. O livro não está na cama.

Considerando que ambas as frases são verdadeiras, pode-se deduzir que o livro, então, está na estante.

Enumerar as etapas a serem realizadas

Algumas ações realizadas por um programa seguem uma lógica sequencial. Como em uma receita, ao trocar a ordem de alguns passos, pode-se ter um resultado diferente.

Algoritmo: montar sanduíche de queijo e mortadela

1. Pegar uma fatia de pão de fôrma e colocá-la sobre o prato.
2. Pegar uma espátula e passá-la na manteiga.
3. Passar a espátula na fatia de pão.
4. Pegar uma fatia de queijo e colocá-la sobre a fatia de pão.

5. Pegar uma fatia de mortadela e colocá-la sobre a fatia de queijo do passo 4.
6. Despejar *ketchup* e mostarda sobre a fatia de mortadela do passo 5.
7. Pegar uma fatia de pão de fôrma e colocá-la sobre o preparo.

Mudar a ordem de algum desses passos (o passo 3 pelo 2, por exemplo) pode tornar o sanduíche um desastre, e o mesmo pode acontecer com programas de computador.

Analisar outras possibilidades de solução

Muitas vezes, as pessoas se atêm a um pensamento e ficam presas em uma solução que pode apresentar problemas ou ser menos eficiente. Por exemplo, considere este problema: “Havendo três pessoas, some a idade das duas mais velhas”. Como você pode resolvê-lo?

Uma solução é:

1. Encontrar a idade da pessoa mais velha
2. Encontrar a idade da segunda pessoa mais velha
3. Somar as duas idades

O problema será resolvido e estará correto, mas, ao pensar um pouco, ainda pode haver outra solução:

1. Encontrar a idade da pessoa mais nova
2. Somar a idade das duas pessoas que sobraram

Você também obterá a resposta correta e talvez de maneira mais rápida, e note que foi aplicada a lógica para a resolução. Assim, quando um algoritmo estiver muito complexo, pare, respire e tente imaginar uma nova solução.

Ensinar ao computador uma solução

Alguns problemas parecem tão óbvios que muitas vezes não se para para pensar nos passos que constituem sua resolução. Por exemplo: “Estou dirigindo a 100 km/h e vou percorrer 50 km. Em quantos minutos vou chegar ao meu destino?”.

Sem pensar muito, você pode responder: “30 minutos”. Porém, como você conseguiu resolver o problema? É isso que deve ser ensinado ao computador:

1. Primeiro, considere o valor “velocidade”.
2. Depois, considere o valor “distância”.
3. Divida o valor “distância” pelo valor “velocidade”.
4. Multiplique o valor obtido por 60.

Como ressaltado anteriormente, pode haver mais de uma solução possível para um problema. O dobro de um número, por exemplo, pode ser obtido multiplicando-o por dois ou somando tal número com ele mesmo.

Pensar em todos os detalhes

Não é à toa que, no ensino de algoritmos, geralmente são feitas comparações com receitas culinárias. Imagine, por exemplo, a receita de um bolo. Pesquise na Internet uma receita qualquer. É muito provável que alguns detalhes estejam presentes, como a ordem dos ingredientes (primeiramente os sólidos, depois os líquidos e, por último, o fermento) e a necessidade de preaquecer o forno, além de temperatura e tempo de preparo. Sem esses detalhes, o bolo provavelmente não ficará como o esperado.

A mesma coisa acontece com a programação para computadores, pois detalhes sobre quais são os passos a serem tomados e como realizá-los precisam ser considerados. Iepsen (2018) apresenta um exemplo bastante didático de algoritmo

para acender um fósforo:

1. Pegar uma caixa de fósforo
2. Abrir a caixa de fósforo
3. Verificar se há palito na caixa. Se sim:
    a. Retirar o palito
    b. Fechar a caixa
    c. Riscar o palito
    d. Verificar se o palito acendeu. Se sim:
        i. Concluído
    e. Se não: retornar ao passo x
4. Se não houver palito: descartar a caixa e retornar ao passo 1

O passo “e” deixa uma incógnita sobre o passo a que se deve retornar. Seria possível retornar ao passo “c”, mas haveria problemas se o palito estivesse úmido, por exemplo. Além disso, quantas vezes seria repetido “riscar o palito”? Seria necessário um limite. Porém, depois desse limite (desistindo desse palito), para que passo se deve ir? Se você optasse pela instrução “a”, teria problemas, pois, por causa do passo “b”, a caixa estaria fechada.

Na montagem de um programa, é preciso testar hipóteses para cobrir o maior número possível de possibilidades de funcionamento. As estruturas de condições e repetição certamente estarão presentes, o que expande as possibilidades de caminhos na execução dos algoritmos.

Sugestões para praticar

Assim como qualquer habilidade, quanto mais você praticar, melhor você ficará. Contudo, no caso dos algoritmos, uma simples repetição dos mesmos algoritmos não ajudará a progredir no processo de aprendizado. A grande tarefa é justamente criar ou procurar novos algoritmos e começar a implementar e solucionar cada um deles, desde os mais básicos até os mais complexos.

Para avaliar se de fato você está melhorando, siga estas dicas:

- Cronometre seus progressos à medida que você for evoluindo. Conforme for realizando alterações, verifique se o tempo de desenvolvimento diminuiu.
- Peça para um desenvolvedor mais experiente avaliar o código que você elaborou.
- Pratique os desafios propostos nos materiais deste curso e procure mais testes e referências também em *sites* externos.

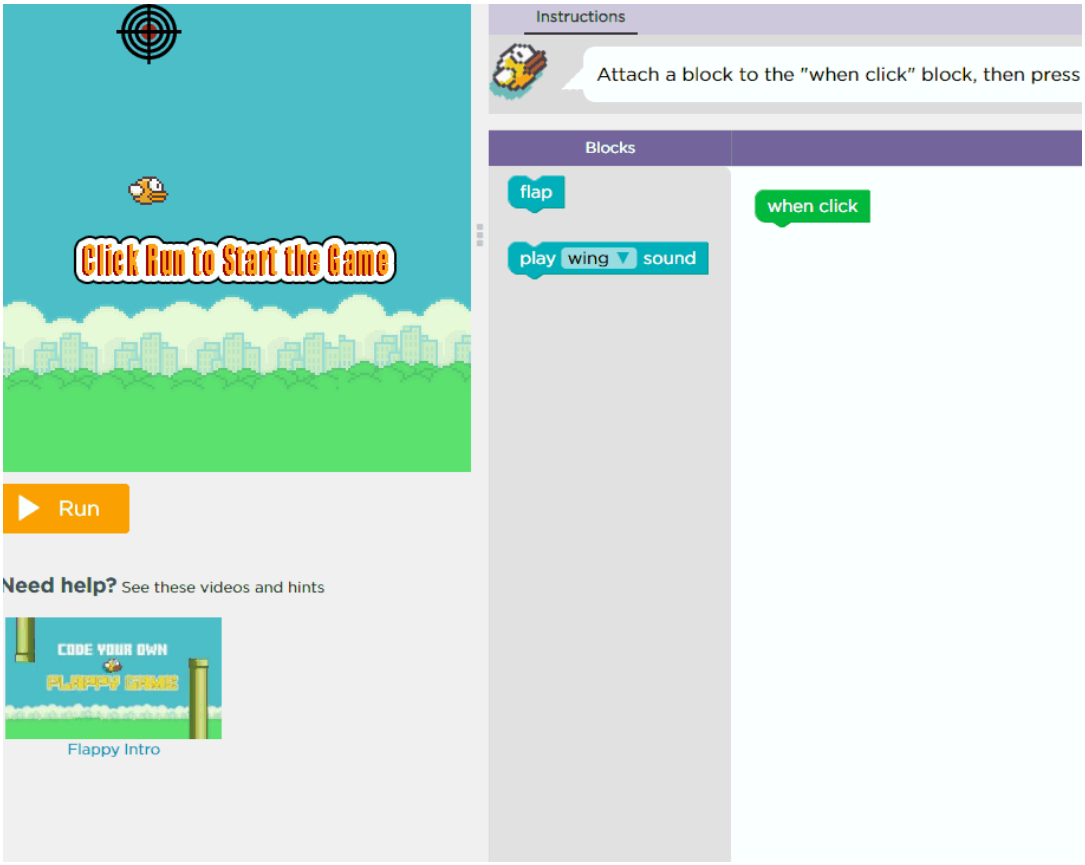
Seguem alguns exemplos que podem ajudar a entender visualmente algumas práticas:

- **Blockly de labirinto**: veja a série de passos necessários para chegar ao ponto final de cada fase do labirinto. Conforme você avança, os **problemas** ficam mais complexos e precisam de mais passos a serem ultrapassados. Pesquise “Maze Blockly” no buscador de sua preferência.

(objetos/gif-blocky_labirinto.gif)

Figura 1 – Funcionalidade do Blockly de labirinto

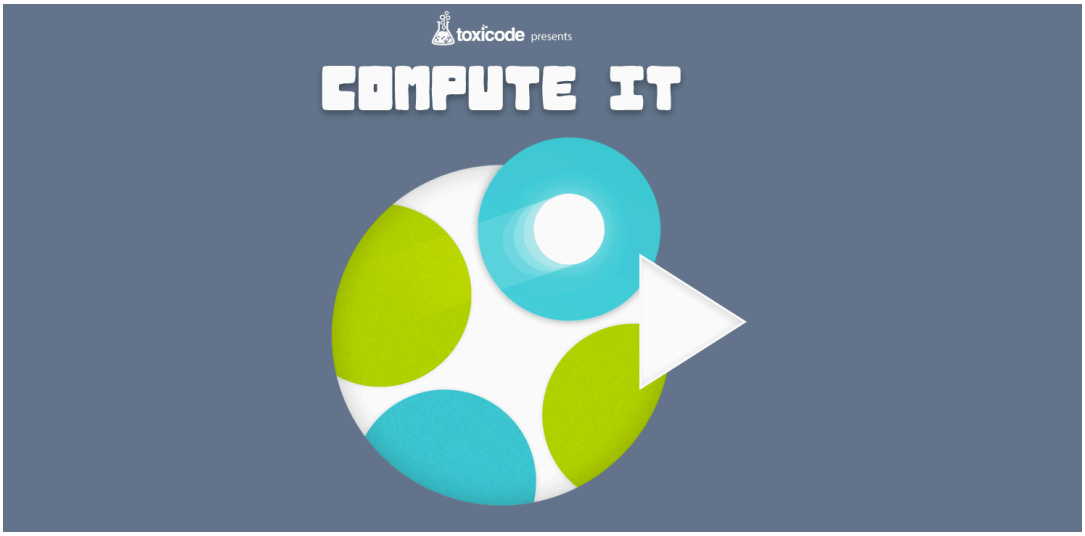
- **Flappy Code**: crie um jogo que simula a lógica de Flappy Bird. A complexidade que é pedida aumenta a cada nível até chegar a uma etapa de um algoritmo bem mais complexo. Pesquise “Studio Code Flappy” no buscador de sua preferência.



(objetos/gif-flappy_code.gif)

Figura 2 – Funcionalidade do Flappy Code

- **Toxicode**: siga os passos e avance de nível. Quanto maior for o nível, mais complexo ele ficará. O Toxicode é uma importante ferramenta para praticar laços de repetições e condicionais. Pesquise “Toxicode compute it” no buscador de sua preferência.



(objetos/gif-toxicode.gif)

Figura 3 – Funcionalidade do Toxicode

Encerramento

Assim como toda resolução de um algoritmo é construída por meio de uma solução de simples ou complexos passos, aqui se pôde explorar a maioria deles de uma forma mais detalhada. O mais importante, sem dúvida, é progredir gradualmente tanto no aprendizado quanto na construção deste.

Os algoritmos que serão encontrados no decorrer do curso e na futura vida profissional serão cada vez mais complexos e estruturados, mas é claro que todos eles podem ser trabalhados com os passos abordados neste material. As questões relacionadas a planejar, solucionar o problema, pensar em alto nível, refinar e finalizar o problema são naturais e surgem com muita prática em diversos algoritmos diferentes.

Você viu apenas os primeiros passos dentro do gigante mundo que é o da programação, mas saiba que há um caminho a seguir e que passos são importantes para cada problema. Com o auxílio deste conteúdo, sem dúvidas você estará mais preparado para lidar com tudo que vier pela frente.