Lógica de Programação

Aula 04 - Expressões

Apresentação

Bem-vindo à Aula 04! Nas aulas anteriores, você se familiarizou com o uso dos algoritmos. Nesta aula, verá as diversas formas de representá-los, conhecendo três das representações mais utilizadas: **descrição narrativa, fluxograma** e **pseudocódigo**. E, para cada uma delas, conhecerá as vantagens e desvantagens. Porém, qualquer que seja a escolha, ela deverá permitir um nível de clareza quanto ao fluxo de execução.

Além disso, você construirá algoritmos com estruturas de decisão aninhadas, pois, à medida que os programas e algoritmos tornam-se mais completos (e complexos), faz-se necessário construí-los com mais desvios de fluxos de execução. Muitos desses fluxos podem ser definidos a partir do aninhamento das estruturas de decisão.

Você também conhecerá a estrutura de decisão **escolha**, que poderá ser uma alternativa aos encadeamentos da estrutura **se...senãose...então**.

Pronto pra começar?

Objetivos

Conhecer três das mais utilizadas Representações de Algoritmos;

Aninhar as estruturas de decisão compostas **se...senãose...então**;

Definir e construir algoritmos com estruturas de decisão compostas;

Construir algoritmos por meio da estrutura de múltipla **escolha**.

Descrição Narrativa

Lembra-se do algoritmo "Receita de Pudim" que você viu no início da disciplina? Não!? Para você relembrar, veja abaixo:

Algoritmo: Receita de Pudim

1. Reunir os ingredientes necessários: 3 Ovos, 200ml Leite, 1 Lata de Leite Condensado, 200g de Açúcar;
2. Colocar os ovos (sem casca) dentro do copo do liquidificador;
3. Colocar o leite condensado dentro do copo do liquidificador;
4. Colocar o leite dentro do copo do liquidificador;
5. Tampar o liquidificador;
6. Acionar o motor por 3 minutos;
7. Colocar o açúcar na fôrma do pudim;
8. Preparar a calda do pudim;
9. Colocar o conteúdo do copo do liquidificador na fôrma;
10. Cozinhar o pudim em banho-maria por 1h30;
11. Esperar o pudim esfriar;
12. Retirar o pudim da fôrma.

Nesse algoritmo, você utilizou o português coloquial (português coloquial é empregado em situações informais, com os amigos, familiares e em ambientes e/ou situações em que o uso da norma padrão da língua possa ser dispensado.) para descrever a sequência de como deveria ser executada a receita, de maneira que qualquer pessoa, sob as mesmas condições, pudesse atingir o mesmo objetivo. O algoritmo acima é representado através da **Descrição Narrativa**, que é uma forma de representar os algoritmos expressos por meio da linguagem natural a partir do uso de textos descritivos.

A principal vantagem da **Descrição narrativa** é que, em razão de você já conhecer a linguagem utilizada (o português), não lhe foi necessário aprender nenhuma nova linguagem ou técnica. Assim, foi possível entender o algoritmo do pudim facilmente, não foi mesmo?

Já por outro lado, o uso de **Descrição narrativa** possui sua desvantagem, pois permite que algumas informações possam ser interpretadas de diferentes modos. Por esse motivo, ela é considerada imprecisa, visto que permite ambiguidades e, em alguns casos, diferentes interpretações. Observe novamente o algoritmo do pudim, faça uma leitura atenta e tente encontrar ao menos um dos passos que pode ser interpretado de mais de uma maneira ou não deixa claro como deve ser executado.

Por exemplo, no **passo de número 12** nos é informado para "Retirar o pudim da fôrma". Mas como esse passo deve ser realizado? Cortando o pudim? Com uma colher? Com uma espátula? Ou colocando a fôrma de cabeça para baixo em um novo recipiente que possa comportar todo o pudim?

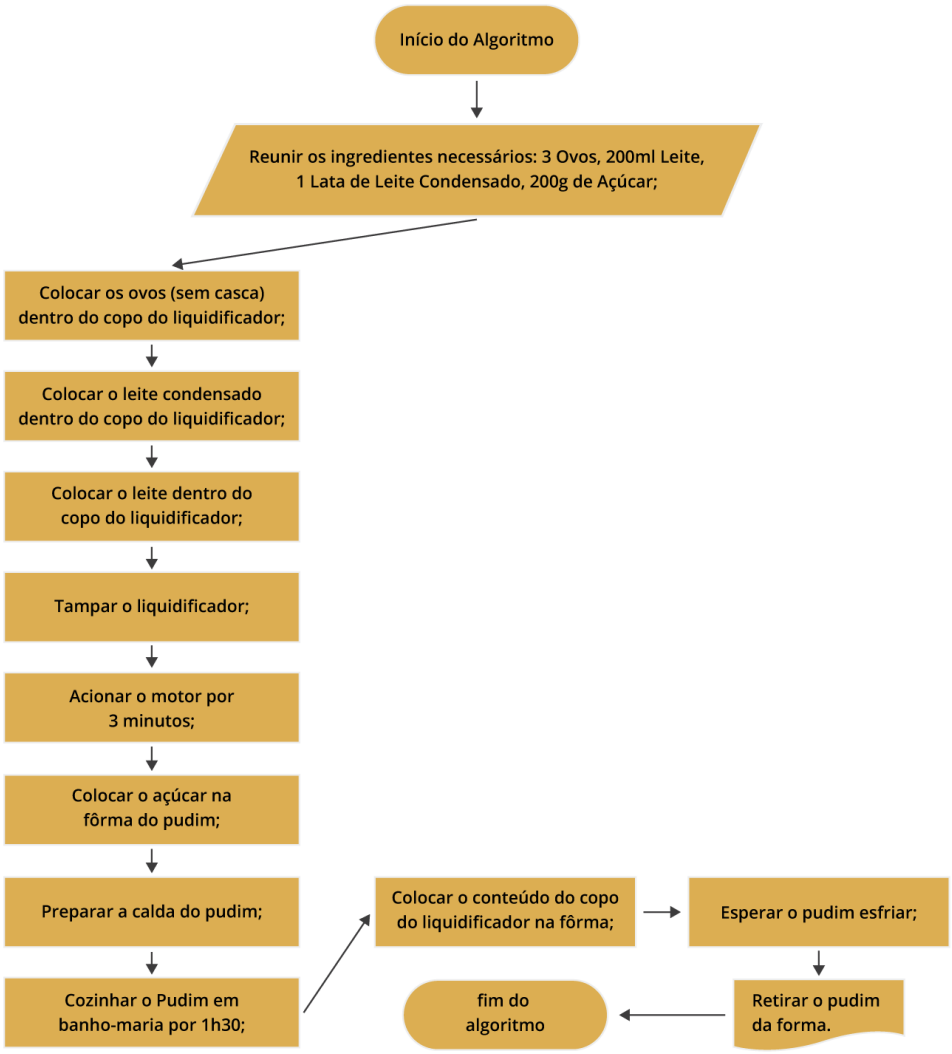
Além do exemplo descrito acima, em algum momento o algoritmo deu a instrução de acoplar o copo do liquidificador no respectivo motor? Não. Ou seja, se ele não estiver acoplado antes de iniciar a execução dos passos, não realizará a mistura dos ingredientes; porém, o algoritmo apenas dá a instrução de acionar o motor por 3 minutos, sem determinar que o copo do liquidificador deva estar acoplado.

Acredito que você já conseguiu compreender a representação de algoritmos através da **Descrição Narrativa** e que, apesar da facilidade de se entender os algoritmos mediante essa técnica, baseada no uso da sua linguagem (português, inglês, alemão, russo, francês, etc.), há desvantagem que limita a sua eficácia.

Fluxogramas

Até o momento, os algoritmos foram apresentados na forma textual. Agora, você conhecerá uma outra forma de representação, que utiliza elementos gráficos. Veja abaixo o algoritmo "Receita de Pudim" de uma maneira diferente.

Figura 01 - Exemplo 01: Algoritmo representado por **Fluxograma**



Viu como ficou a "Receita de Pudim" com essa técnica de representação de algoritmos? O que achou? Sua compreensão do algoritmo ficou mais clara? Você consegue encontrar alguma ambiguidade nos passos executados?

Fluxograma é uma representação que permite descrever de maneira clara e precisa os passos e fluxos necessários para a realização de algo. É uma forma padronizada e eficaz para representar os passos lógicos de um **algoritmo**.

Os **fluxogramas** possuem como conceito a técnica de representação gráfica dos algoritmos, realizada através de símbolos (figuras geométricas) com significados previamente definidos.

Por meio da representação de algoritmos com a utilização de **fluxogramas**, é possível definir passos e interconexões. Dessa forma, torna-se mais fácil a visualização dos fluxos existentes em um algoritmo.

Os **fluxogramas** utilizados para a representação dos algoritmos seguem um padrão em que as figuras geométricas possuem diferentes significados. Observe a Figura 02:

1	Representa início ou fim do algoritmo

2 Representa uma entrada de dados no algoritmo

3 Representa ação de processamento e/ou atribuição

4 Representa uma decisão (desvio de fluxo)

5 Representa exibição de dados ou mensagem

6 Representa exibição de dados ou mensagem

7 Representa início ou fim do algoritmo

8 Representa o sentido da execução, fluxo do algoritmo

9 Representa o sentido da execução, fluxo do algoritmo

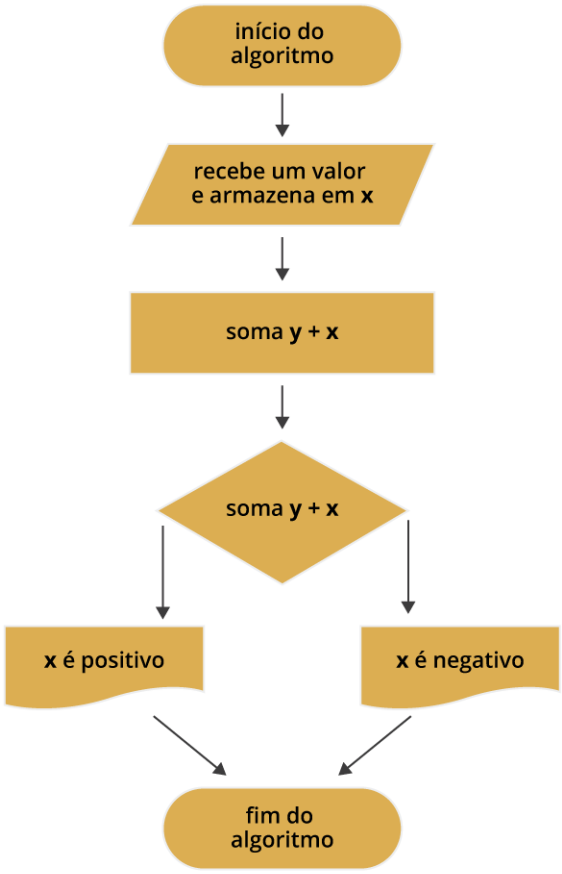
10 Representa o sentido da execução, fluxo do algoritmo

11 Representa o sentido da execução, fluxo do algoritmo

12 Representa o sentido da execução, fluxo do algoritmo

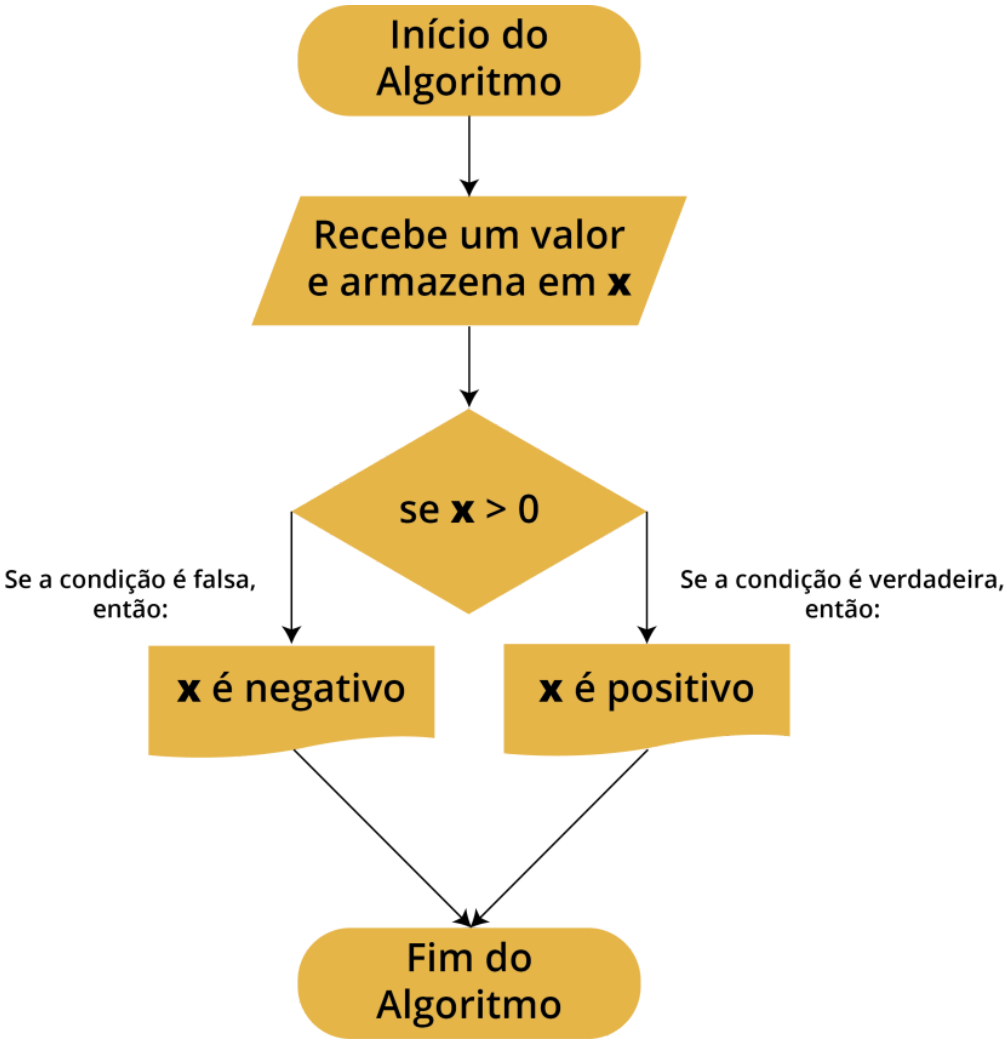
13 Representa o sentido da execução, fluxo do algoritmo

Figura 02 - As representações em um **Fluxograma**



Agora você vai rever um algoritmo exibido na aula anterior por meio de **Fluxograma**. Esse recurso foi utilizado para auxiliar na compreensão do algoritmo apresentado naquele momento, mesmo que você ainda não tivesse conhecimento sobre a construção de algoritmos por meio dos **fluxogramas**. Essa técnica de representação gráfica será explorada na Figura 03.

Figura 03 - Exemplo 02: Algoritmo representado por **Fluxograma**

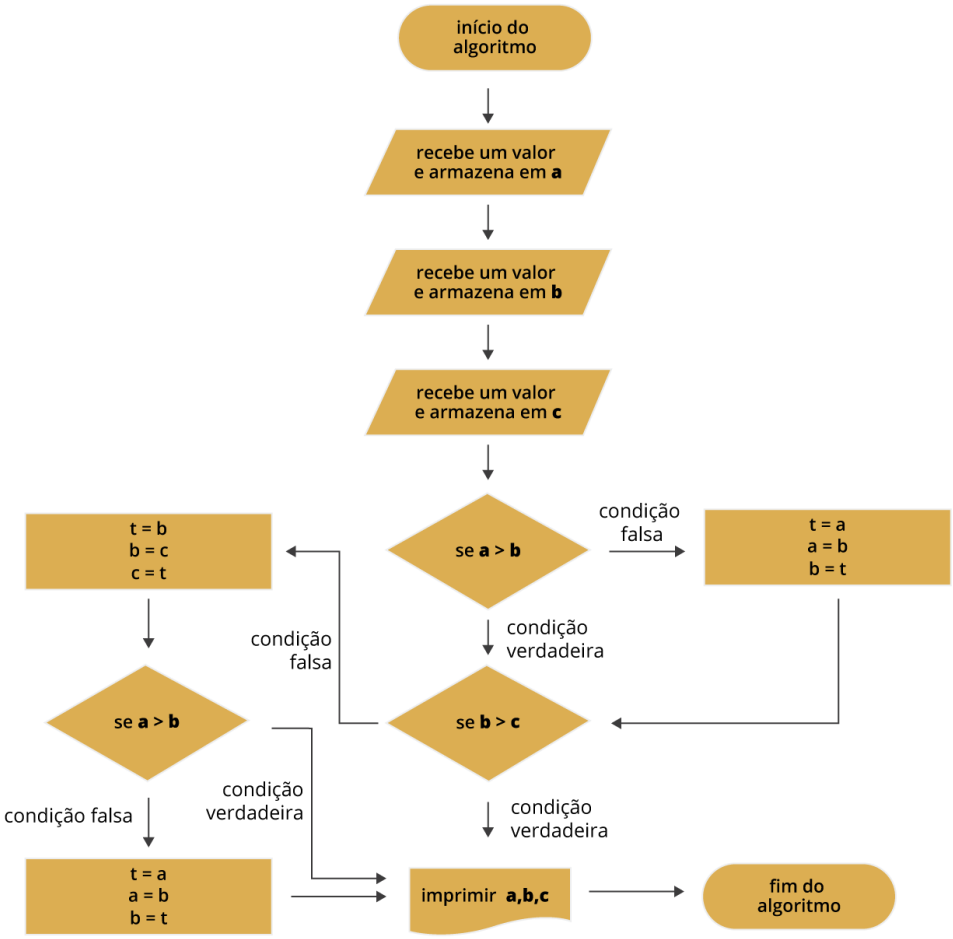


O exemplo apresentado recebe um número qualquer, armazena na variável **x** e, em seguida, verifica se o número recebido é positivo. Para algoritmos com poucos fluxos e de tamanho reduzido, o **Fluxograma** torna-se adequado e permite seu fácil entendimento.

Veja abaixo um outro algoritmo também representado por **Fluxograma**. Este já possui mais de um fluxo de decisão, ou seja, dependendo dos valores fornecidos no início do algoritmo, ele poderá executar trechos (blocos) diferentes. Pode-se assumir que, neste exemplo, há regiões do algoritmo que não serão executadas.

Analise com atenção o exemplo da Figura 04 e a partir dos fluxos procure compreender qual o trabalho realizado pelo algoritmo. Em seguida, discuta com o seu professor mediador a função do algoritmo observado.

Figura 04 - Exemplo 03: Algoritmo representado por **Fluxograma**



Você notou que nesse novo algoritmo há possibilidades de desvios de fluxos? Note que foram utilizadas três estruturas de decisão **se...então...senão**. Nesse caso, o fluxo seguirá por caminhos diferentes, dependendo dos valores fornecidos para as variáveis **a, b** e **c**.

Como uma das desvantagens, quando os **fluxogramas** são utilizados para algoritmos extensos ou muito complexos, há um aumento expressivo em seu tamanho, devido à necessidade de se definir todos os fluxos.

Por exemplo: ao comparar a Figura 04, com três estruturas de decisão, com a Figura 03, em que há apenas uma, nota-se que a quantidade de elementos geométricos foi ampliada consideravelmente. Assim, os **fluxogramas** em algoritmos maiores ou mais complexos tendem a tornarem-se extensos, o que poderá dificultar o seu entendimento.

Além da extensão do **fluxograma**, outra desvantagem é que as informações apresentadas podem não ser claras. Para você, o que representa o **t** presente nos elementos de processamento? Pense a respeito e discuta no fórum da disciplina qual o trabalho realizado pelo **fluxograma** da Figura 04 e a importância de **t** nesse algoritmo.

Um outro ponto é que nessa estratégia há a necessidade de familiarizar-se com os símbolos (figuras geométricas e seus significados) para um correto entendimento do **fluxograma**. Esse padrão não é universal e poderá existir pequenas diferenças. Nesta disciplina, serão sempre adotados os significados representados na Figura 02.

Pseudolinguagens

Como você pode ter percebido, nesta disciplina, você já teve a oportunidade de conhecer três formas de representar os algoritmos. A primeira representação foi a **Descrição narrativa** vista na Aula 01, através da receita de pudim. A segunda, a **Pseudolinguagem**, na Aula 02, através de exemplos de código, e, por fim, o **Fluxograma**, na Aula 03, em que foi apresentada a estrutura de decisão.

A segunda forma de representação, a **Pseudolinguagem**, que você já conheceu por meio de exemplos, é a maneira de se escrever algoritmos através do 'português estruturado', ou seja, escrever utilizando uma estrutura semelhante à de uma linguagem de programação e, nesse caso, utilizando a língua portuguesa.

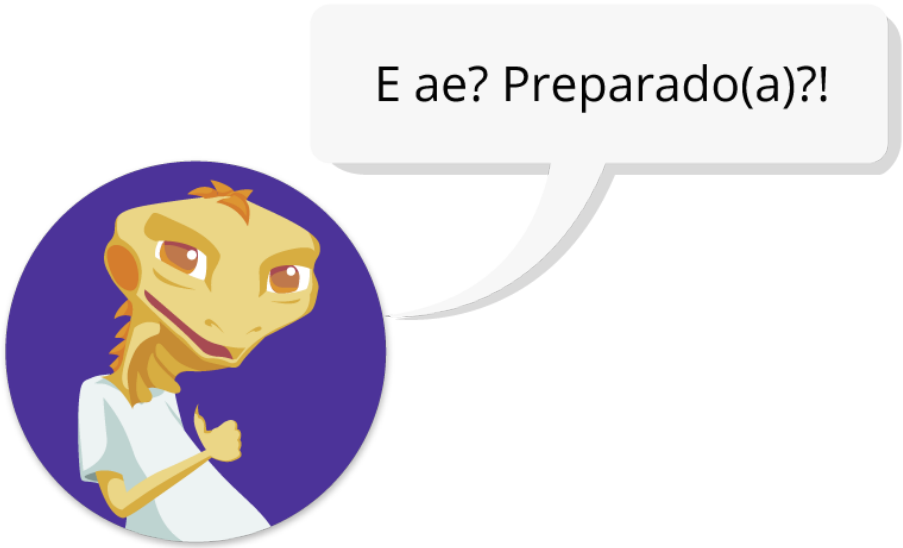
Ela apresenta duas vantagens: a primeira é a precisão relacionada aos **fluxogramas**, e a segunda, a possibilidade de descrever com maior riqueza, como na **Descrição narrativa**. Assim, mesmo que o **Pseudocódigo** seja escrito independentemente de uma linguagem de programação, sua estruturação permite, caso necessário, transcrever o código com maior facilidade para a maioria das linguagens de programação. Dessa forma, pode-se dizer que o uso do **Pseudocódigo** permite escrever algoritmos em estruturas mais próximas dessas linguagens.

Já a desvantagem do seu uso está relacionada à necessidade de se aprender um padrão, ou seja, as regras da **pseudolinguagem** utilizada. Essas regras dizem respeito à sintaxe da pseudolinguagem e dos comandos existentes nela, as quais

podem, ainda, ser diferentes entre pseudolinguagens distintas. Mas não se preocupe: apesar das diferenças que você pode encontrar, as variações são facilmente contornadas.

Como dito anteriormente, você já viu na disciplina três formas em que são representados os algoritmos: a **Descrição Narrativa**, os **Fluxogramas** e a **Pseudolinguagem** em português Potigol.

Agora, no próximo passo, você conhecerá umas das estratégias de programação, chamada de **aninhamento**.



Estruturas Aninhadas

Na aula anterior, você viu e utilizou os operadores Aritméticos, Relacionais e Lógicos, como também construiu algoritmos por intermédio da estrutura de decisão **se...então...senão**. Nesta, conheceu algumas das formas de representação dos algoritmos. Que tal continuar com a construção deles?

Para criar desvios de fluxo, você faz uso de estruturas de decisão em seus algoritmos. Nos algoritmos mais simples, a quantidade de fluxos é reduzida e possui poucos desvios. Nesse sentido, à medida que você cria algoritmos com diversos

fluxos, em sua construção, o uso de **aninhamentos** é frequente.

O **aninhamento** ocorre quando se utiliza uma estrutura dentro de uma mesma estrutura. Por exemplo, se você utiliza a estrutura de decisão **se...então** dentro de uma outra estrutura **se...então**, você terá um **aninhamento** dessa estrutura **se...então**. Assim, diz-se que temos um **aninhamento de se's**, também conhecido por **Desvio Condicional Aninhado**.

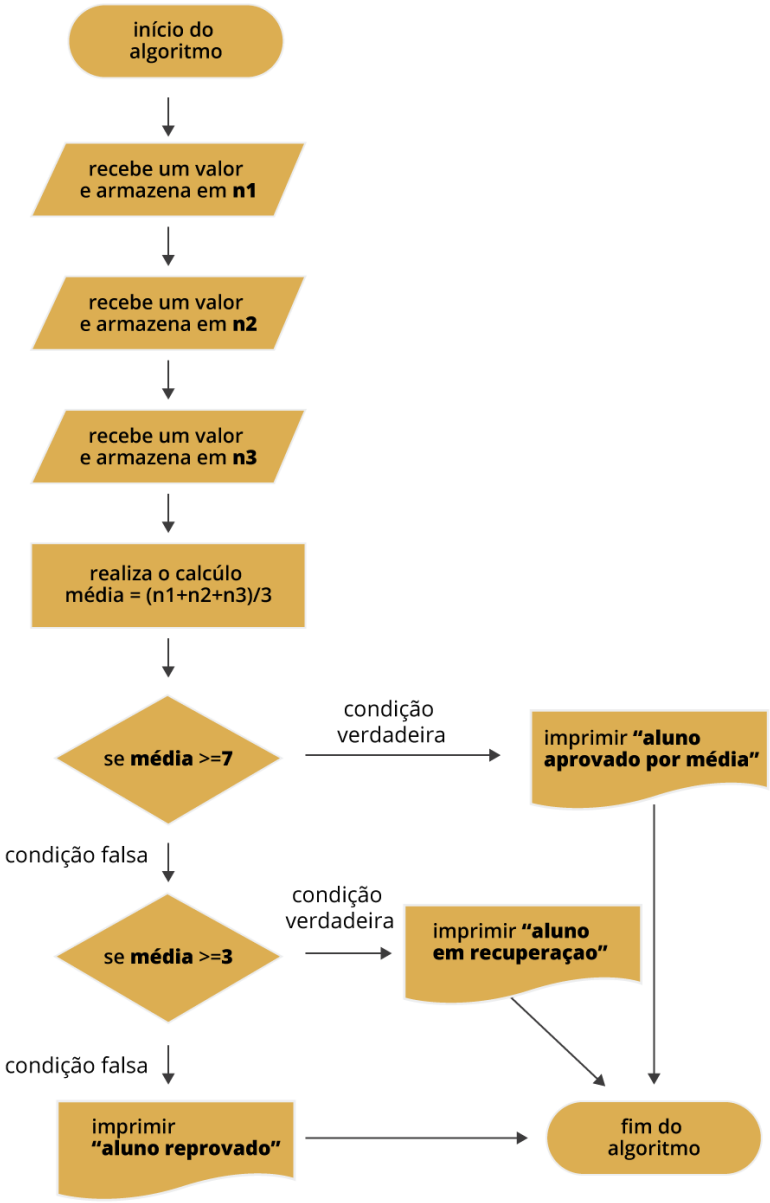
No algoritmo abaixo foi utilizado o **aninhamento** da estrutura de decisão **se...então...senão**. Observe que o código do algoritmo recebe três números, que representam as notas do usuário na disciplina de Lógica de Programação, e, em seguida, calcula a média aritmética (A média aritmética é a soma de todos os números que irão compor a média; ao final, é realizada a divisão da soma total pela quantidade de números que foram somados.) desses números.

Conteúdo interativo, acesse o Material Didático.

No algoritmo acima, foram solicitados três números, foi realizado o cálculo da média e, então, o resultado do cálculo foi armazenado na variável **media**. Dependendo do valor presente nessa variável, o algoritmo realizará o desvio de fluxo uma ou duas vezes.

Veja a representação dos fluxos existentes desse algoritmo no **Fluxograma** a seguir:

Figura 05 - Exemplo 04: Fluxograma com cálculo de média.



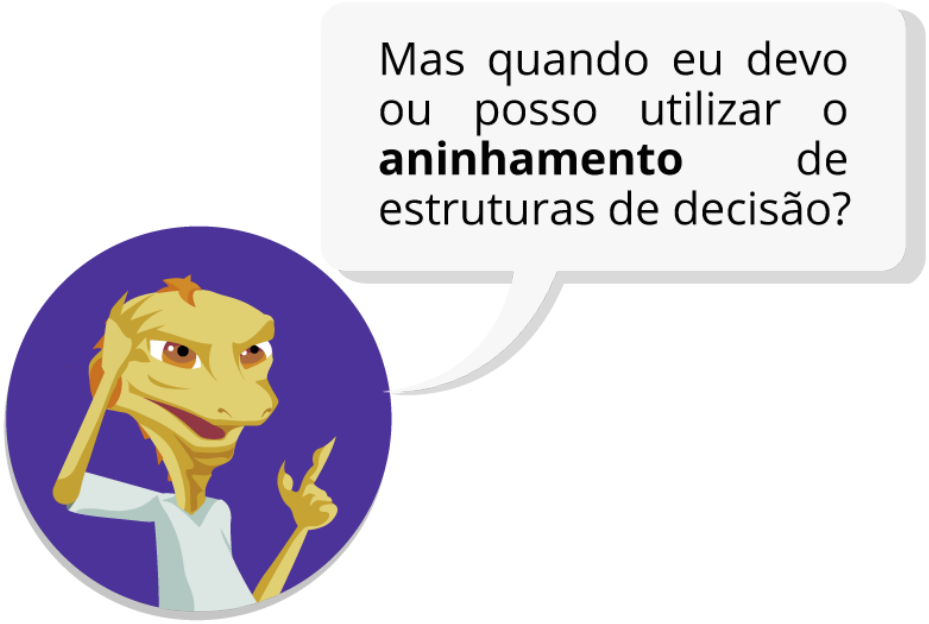
Nesse **Fluxograma**, se a média for igual ou superior a 7, haverá somente um desvio de fluxo, mediante o qual o algoritmo será direcionado para o bloco de códigos referente à condição **verdadeira** da primeira estrutura de decisão **se** e, então, fará a impressão da mensagem, informando que o aluno foi aprovado por média e finalizando o algoritmo logo em seguida.

Ainda no primeiro **se**, quando a média for inferior a 7, o fluxo será desviado para o bloco de códigos referente à condição **falsa**. Internamente, nesse bloco há uma segunda estrutura de decisão **se**, e nela verifica-se novamente o valor da variável **media**. Caso o valor seja maior ou igual a 3, o desvio será realizado, agora, para o

bloco de códigos referente à condição **verdadeira** da segunda estrutura de decisão **'se'**. Nesse bloco de código, será impressa a mensagem informando que o aluno está em recuperação e, logo após, finalizando o algoritmo.

Porém, se o valor da variável **media** for inferior a 3, o algoritmo será desviado para o bloco de códigos de condição falsa do segundo **se** e, nesse caso, será impressa a mensagem informando que o aluno foi reprovado e, assim, o algoritmo será finalizado.

A partir desse exemplo, foi apresentado o **aninhamento** de uma estrutura de decisão **se...então...senão** no bloco de condição falsa de um primeiro **se..então...senão**. Talvez você esteja se perguntando: "Posso colocar somente uma segunda estrutura de decisão no bloco de condição **falsa**?" A resposta é não, isto é, também é possível aninhar a estrutura de decisão **se** dentro do bloco de condição **verdadeira**. Inclusive, há a possibilidade de ter **aninhamento** de várias estruturas de decisão no bloco da condição **verdadeira** ou **falsa**, como também é possível ter mais de um nível de **aninhamento**.



Geralmente, é preciso aninhar estruturas de decisão quando é necessário realizar mais de uma verificação, ou seja, mais de um 'teste'. Nesse exemplo, foi necessário verificar em qual situação o aluno seria enquadrado e, para isso, a média

deveria ser verificada em duas etapas: primeiramente, foi verificado se o aluno estava aprovado por média e, caso contrário, seguia-se para o segundo passo, conferindo se o aluno possuía a média necessária para recuperação ou não.

Um novo exemplo!

O algoritmo a seguir poderá ser utilizado para verificar se o aluno de um curso superior da UFRN está aprovado por média ou necessitará fazer uma prova de reposição. Para que esse aluno seja aprovado por média, ele deve possuir média igual ou maior que 7, contudo, se esse aluno possuir média igual ou superior a 5 e não tiver nenhuma nota abaixo de 3, ele também estará aprovado.

Se o aluno tiver média igual ou superior a 5, mas qualquer uma das 3 notas que ele obteve durante suas aulas for menor que 3, ele deverá fazer a prova de reposição. Agora, se a média for menor do que 5 e maior ou igual a 3, então ele fará a prova de recuperação. Em último caso, se a média for menor que 3, o aluno não poderá fazer a recuperação e estará reprovado por média.

Conteúdo interativo, acesse o Material Didático.

Você reparou que no algoritmo apresentado há diversos **aninhamentos** do **se...então...senão**? Como dito, esses **aninhamentos** podem ser tanto no bloco da condição **verdadeira** como no bloco da condição **falsa**. Nesse mesmo algoritmo, foi possível inserir mais de um nível de aninhamento através do uso de vários **se...então...senão**, um dentro do outro.

Agora, modifiquei esse algoritmo utilizando o operador lógico '**|e|**' que você conheceu na aula anterior. Neste exemplo, é analisada a situação do aluno nos mesmos requisitos que o algoritmo apresentado anteriormente. Observe o código seguinte com atenção e perceba que foi diminuída a quantidade de estruturas **se...então...senão**, de modo a ser obtido um algoritmo mais fácil de ler e entender.

Conteúdo interativo, acesse o Material Didático.

E aí? Conseguiu perceber que utilizando o operador lógico '**|e|**' foi possível reescrever o algoritmo com três estruturas de decisão **se...então...senão** a menos? Apesar disso, os algoritmos de cálculo e de verificação da situação do aluno são

equivalentes e, para os mesmos valores de notas **n1**, **n2** e **n3**, ambos irão gerar as mesmas respostas.

Conheça uma outra forma de escrever algoritmos com a estrutura **se...senãose** aninhada.

Conteúdo interativo, acesse o Material Didático.

Nesse exemplo, foi apresentada uma forma diferente de se encadear a estrutura de decisão **se...então...senão**. Repare que no primeiro comando **se**, caso a variável **media** seja menor que 7, é realizado um novo teste com o comando **senãose** (uma junção dos comandos **senão + se**). Nessa nova verificação, o teste compara se o valor armazenado em **media** é maior ou igual a 5.

#FicaDica

É possível construir algoritmos diferentes para resolver um mesmo problema e obter o mesmo resultado, dessa forma, não há somente uma solução correta. Em muitos casos, há infinitas soluções.

Atividade 01

Crie os **fluxogramas** dos exemplos apresentados em **Pseudocódigo** nesta aula e apresente os **fluxogramas** no encontro presencial ao seu professor mediador.

Atividade 02

Conteúdo interativo, acesse o Material Didático.

Estrutura de Decisão Escolha

Na aula anterior, você estudou a estrutura de decisão **se...então**, agora você conhecerá outra estrutura de decisão, denominada de estrutura de decisão **escolha**. No que diz respeito ao âmbito das linguagens de programação, essa estrutura recebe o nome de **switch case**.

Uma estrutura de decisão **escolha** permite "enfileirar" vários blocos de código, semelhantemente a um corredor cheio de portas. Assim, 'dependo da chave', uma dessas portas seria acessada. Algumas pessoas consideram essa estrutura de decisão semelhante ao **se**, porém, com várias possibilidades. Conheça agora a sintaxe da estrutura de decisão **escolha**.

Conteúdo interativo, acesse o Material Didático.

A estrutura de decisão **escolha** permite definir vários comandos **caso**. No exemplo de código acima, dependo do valor da variável **x**, será executado o bloco de comandos após o símbolo '**=>**' até o início do próximo comando **caso**. No exemplo, temos **caso 1**, **caso 2** e **caso 3**. Se o valor de **x** for igual a 1, o primeiro comando **escreva** será executado; se o valor for 2, então o segundo comando **escreva** será acionado; já na situação em que o valor de **x** seja 3, o terceiro comando **escreva** terá então sua vez.

E se o valor de **x** não for 1, nem 2 e nem 3? Então o '**caso_**' e o respectivo bloco de código serão acionados. Essa opção '**caso_**' é acionada quando nenhum dos outros comandos **caso** é acionado, em outras palavras, se o valor de **x** não coincidiu com nenhuma das outras situações, '**caso_**' será executado. Pode-se dizer que equivale ao "senão" de um último **se...então...senão**.

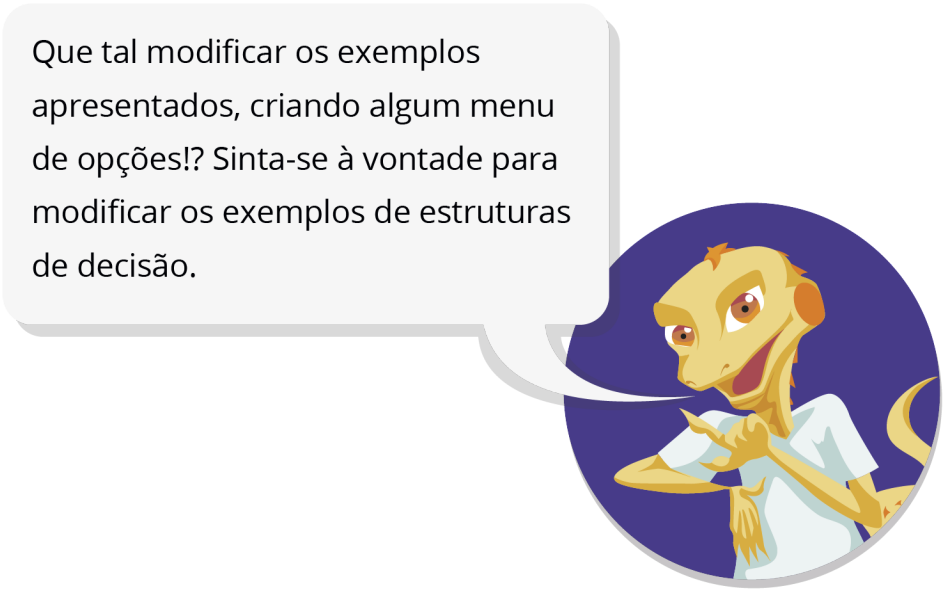
O uso da estrutura de decisão **escolha** permite simplificar algoritmos, principalmente quando for necessário sequenciar múltiplas estruturas **se...senãose...então**. Nesses casos, é provável que o algoritmo fique mais simples quando é adotada a estrutura de **escolha**. Veja um exemplo.

Conteúdo interativo, acesse o Material Didático.

Nesse código, o usuário do programa deverá informar o número do mês em que nasceu e as opções válidas são de 1 até 12. Qualquer valor diferente desse intervalo acionará a opção '**caso_**', que possui como conteúdo a mensagem de mês inválido.

#FicaDica

Esse exemplo pode fazer parte de um menu de opções de um programa e demonstra uma boa situação em que a estrutura de decisão **escolha** (switch case) é utilizada.



Atividade 03

Conteúdo interativo, acesse o Material Didático.

Resumo

Nesta aula você compreendeu as principais formas de representação dos algoritmos, passando pela **descrição narrativa**, pelos **fluxogramas** e pela **pseudolinguagem**. Viu ainda o que é o **aninhamento** de estruturas e como ele ocorre com a estrutura de decisão **se...então...senão**.

Além disso, conheceu também o encadeamento **se...senãose...então**. Por último, teve a oportunidade de conhecer as estruturas de decisão **escolha**.

Referências

Linguagem Potigol: Programação para todos. Disponível em: <http://potigol.github.io/>. Acesso em: 10 de nov. 2017.

Fluxogramas, diagrama de blocos e de Chapin no desenvolvimento de algoritmos. Disponível em: <https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550>. Acesso em: 23 dez. 2017.

Pseudocódigo. Disponível em: <http://www.hardware.com.br/termos/pseudocodigo>. Acesso em: 01 de jan. 2018.

Lógica de Programação – Desvio Condicional Aninhado. Disponível em: <http://www.bosontreinamentos.com.br/logica-de-programacao/12-logica-de-programacao-desvio-condicional-aninhado-se-entao-senao-se/>. Acesso em: 01 de jan. 2018.

Switch statement. Disponível em: <http://www.javaforstudents.co.uk/Switch_statement>. Acesso em: 25 de fev. 2018.