

Eletrônico



Estratégia
CONCURSOS

Aula

Conteúdo em Informática IV Banco do Brasil (Escritório) Com videoaulas

Professor: Diego Cavallini, Equipe Informática: Tiago de Paiva, Thiago Rodrigues Cavallini, Vinicius Dalma

AULA 00

SUMÁRIO	PÁGINA
Apresentação	01
- Lógica de Programação	08
- Funções e Procedimentos	62
Lista de Exercícios Comentados	65
Gabarito	79

ATENÇÃO:
VÍDEO-AULAS
COMEÇARÃO A SER
DISPONIBILIZADAS
DIA 14/03



O PROFESSOR...

PROF. DIEGO CARVALHO

FORMADO EM CIÊNCIA DA COMPUTAÇÃO PELA UNIVERSIDADE DE BRASÍLIA (UNB), PÓS-GRADUADO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO NA ADMINISTRAÇÃO PÚBLICA E, ATUALMENTE, AUDITOR FEDERAL DE FINANÇAS E CONTROLE DA SECRETARIA DO TESOIRO NACIONAL.

ESTRATÉGIA CONCURSOS

Já ministrei mais de 120 cursos de Tecnologia da Informação no Estratégia Concursos, incluindo concursos importantes como: TCU; PF; Senado e Câmara Federal; etc. Nossa equipe já possui vasta experiência em concursos, de forma que você não precise procurar mais nenhum outro material de estudos além dos nossos para fazer sua prova.

ENTRE EM CONTATO:



PROFESSORDIEGOCARVALHO@GMAIL.COM



FACEBOOK.COM/PROFESSORDIEGOCARVALHO



ATENÇÃO

ESSA AULA É APENAS UMA DEMONSTRAÇÃO DO MÉTODO, ESTILO E ESCRITA DO PROFESSOR. CASO ADQUIRA O CURSO, O RESTANTE DAS AULAS E EXERCÍCIOS SERÃO DISPONIBILIZADOS.



A AVALIAÇÃO DOS PROFESSORES...



Avaliação do Professor

SIM! APÓS O FIM DO CURSO, VOCÊS PODEM NOS AVALIAR E DIZER O QUE **GOSTARAM E O QUE NÃO GOSTARAM**. CONFIRAM ABAIXO O QUE ALGUNS ALUNOS DISSERAM SOBRE MEUS CURSOS:

Comentário sobre o curso
O curso está excelente para a preparação para o concurso do TCE-SP e outros que tem abordado os mesmos temas. Bem focado nos temas e apresenta os aspectos gerais de cada um deles. O professor comentou muitos exemplos e exercícios. Nem senti falta de vídeos.
Olá professor! de forma geral gostei do curso. Eis alguns pontos para melhorar: - o fato de limitar o tamanho dos parágrafos em poucas linhas facilita, mas acho que não deveriam ser quebrados no meio do mesmo assunto...algumas vezes me perdi... - a observação a respeito do modelo espiral, no qual alguns autores consideram incremental e outros não, poderia ser reforçado no tópico do modelo espiral, mais para o fim da aula (já não lembrava mais desta informação quando cheguei neste modelo...) Desejo muito sucesso!
Achei o material muito bom, objetivo e com muitos exercícios. São resumos sobre os assuntos, então só vou saber se a profundidade é adequada para concursos depois que fizer a prova, mas no geral me pareceu que cobriu bem os temas. Att, Nádia.
Muito bom o curso.
okjok
DADA A QUANTIDADE DE TEMAS ABORDADOS E EXPLICADOS, ACHEI UM CURSO BEM COMPACTO E PRÁTICO.
Apresentou um número de questões comentadas muito bom e com bons esclarecimentos.
Muito bom mesmo. Muitas questões do CESPE mesmo que o concurso fosse da Vunesp. Será que a FCC, FGV e Cesgranrio não tem questões desses assuntos? As questões CESPE são "diferentes"... Não tive problemas com isso porque foco o TCU mas fica a dica. Discursivas sobre BI e big data. Sugestões, por favor.
O curso foi excelente e atendeu perfeitamente o meu plano de estudo. Pois precisava de mobilidade e consegui estudar tanto no smartfone, tablet e notebook principalmente no status Off Line porque nem sempre tinha acesso a internet disponível. A didática do professor foi um ponto forte neste curso: A explicação da teoria com uma linguagem clara, cheias de questões comentadas e bom humor como se eu estivesse em uma aula presencial!! Os assuntos-chaves destacados em vermelho otimizou muito as minhas revisões!
Curso com um conteúdo excelente, bem produtivo e rico nos detalhes passados que me ajudaram muito.
Gostei muito da linguagem com certeza compraria outros cursos
Não tenho referência de outros cursos porque este é o primeiro que estou estudando pela internet mas pra mim foi muito bom. A linguagem não é cansativa e acredito que o que se estuda é somente o necessário. Uma coisa que não entendi é que as vezes havia algumas questões que havia a pergunta e a letra da resposta mas não havia as respostas para escolher. Acho que poderia melhorar isso ou então não entendi o conceito.
O professor tinha uma abordagem informal do assunto que facilitava a memorização.
Melhor professor de TI que já vi na área de concursos!

Comentário sobre o curso
Excelente curso! Excelente material, excelente professor, excelente equipe. Recomendo a todos!
"Excelente" é pouco para esse curso. Eu adquiri outras aulas para me preparar para esse concurso, mas nenhuma outra agregou tanto. Imagino o trabalho que deve ter dado para "compilar" todos esses assuntos. Além disso, colocá-los de forma tão didática em aulas relativamente pequenas. Isso é altíssima qualidade! O conteúdo foi muito bem elaborado! E eu reparei isso porque depois que estudei por ele consegui resolver muitas questões que procurei sobre esses assuntos. Questões que antes pareciam ser de "outro mundo" pra mim. A organização da aula também merece destaque... As figuras, os esquemas, as questões, as notas de rodapé e a evolução do conteúdo estudado de acordo com os itens no edital... Sem falar na didática! Quanto ao professor, sem palavras. Respondeu todas minhas dúvidas e sempre se mostrou bastante receptivo. Bom, não sei como irei me sair na prova domingo, mas devo MUITO da minha preparação ao professor Diego Carvalho. Ele nem imagina o quanto me ajudou. Com certeza recomendo esse curso!
Eu me considero uma pessoa leiga em TI, pois sou formado em Engenharia Mecânica e pos graduado em economia, além de nunca ter trabalhado em departamentos de TI nas empresas onde trabalhei. Por isso, apesar de ter tido dificuldades, achei muito adequada a linguagem adotada pelo professor. Os assuntos são muito complexos para serem ensinados a pessoas com pouca base de conhecimento na área, e ainda assim eu entendo que o curso tenha atingido seu objetivo de permitir a transmissão do conhecimento proposto aos alunos. Como pontos de melhoria, sugeriria: - utilização de videoaulas, pois diversos assuntos poderiam ter seu entendimento facilitado com a utilização de imagens - redistribuição na sequência de assuntos. Ainda que eu entenda que vários assuntos sejam interligados, o que dificulta a escolha sobre "o que ensinar primeiro", acho que determinados assuntos poderiam ter sua ordem invertida, ainda que isto não tenha impedido o entendimento da matéria. Muito obrigado. André.
NULL
Excelente curso, muito didático, material de alta qualidade.
Ok

Comentário sobre o curso
ótima didática do Professor Diego!!
"Sempre podemos melhorar" Está muito bom o curso, o pessoal da Estratégia está de parabéns. abs. Airton.
O curso ajuda demais a estruturar e ordenar o conteúdo dos cursos, coisa quase impossível de fazer sozinho. Será o primeiro concurso que irei fazer para avaliar realmente a eficácia do curso. Sempre lembrando que é muito importante e a exemplificação da teoria com exemplos práticos reais e ou fictícios que ajudam a fixar o conteúdo, principalmente para aqueles cursos que tem provas discursivas ou questões abertas. Marco Costa
Aulas muito bem explicadas. Respostas muito rápidas. Conteúdo abrangente, focado e muito bem baseado bibliograficamente. Parabéns. Excelente curso.
Esta foi a matéria que mais gostei dentre as abordadas para o concurso do TRT. Gostei muito da linguagem que o professor utilizou e os exercícios no meio do conteúdo deixam a aula mais dinâmica!
Excelente material. Focado para o concurso e muita questão para fixação.
Melhor didática dos pacotes de TI pro TRT.
É uma pena que a última aula chegou tão perto da prova, mas eu entendo a complicação e é fato que o professor fez um trabalho muito bom. Vou revizar o último PDF amanhã, mas até o momento, de longe, foi o melhor material que eu encontrei sobre os outros assuntos. Uma sugestão de melhoria é o próprio site. O sistema de perguntas e respostas é horrível, feio e de baixíssima usabilidade.
Excelente didática! As apostilas dão vontade de ler até quando o assunto não é dos mais legais. Parabéns pelo curso!
O curso é excelente. Sugiro ao Professor lançar um curso regular de Java tratando todos assuntos (programação, JEE, etc.).
Muito bom o material, apostilas com muito exercícios.



alinhando expectativas...

O curso que eu proponho abrangerá todo o conteúdo do meu cronograma, entretanto é impossível e inviável esgotar cada ponto do edital em uma aula escrita. Como se ministra Java em uma aula? Teríamos uma aula de 800 páginas e não chegaríamos nem perto de matar todo conteúdo! Imaginem agora cada ponto do Edital.

01

Portanto, vou direcioná-los pelo conteúdo da melhor maneira possível. O nosso foco é ter uma visão geral, mas objetiva do que de fato cai em prova e, não, elucubrações sobre cada tema. Meu foco aqui é te fazer passar!

Eu sei como é complicado ler muita coisa (ainda mais de TI) e vocês têm outras disciplinas para estudar. Logo, vou ser simples e objetivo! Tranquilo? ;)



02

Além disso, o cronograma será seguido com a maior fidelidade possível, mas ele não é estático e poderá haver alterações no decorrer do curso.

Eventualmente, posso tirar o conteúdo de uma aula e colocar em outra de forma que o estudo de vocês fique mais lógico, coeso e fácil de acompanhar; posso também inverter a ordem das aulas (adiantar uma aula e atrasar outra) - sem prejudicá-los.

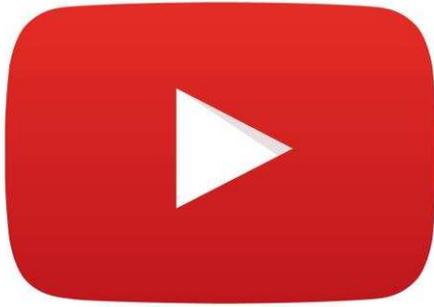
03

Ademais, vamos usar questões de diversas bancas. Enfim, confiem em mim: o curso vai ajudar bastante! Qualquer dúvida, é só me chamar! Caso haja alguma reclamação, problema, sugestão, comentários, erros de digitação, etc, podem enviar para o nosso fórum que eu tento responder da maneira mais tempestiva possível. Ainda duvidam que PDF não dá certo com Concursos de TI?



04





CONHEÇA O 6º LUGAR - ISS/SALVADOR

www.youtube.com/watch?v=blw4H3l6mC4#t=1678



CONHEÇA O 1º LUGAR - TRT/RJ

www.facebook.com/video.php?v=790616534367672



CONHEÇA O 3º LUGAR - ALERJ

www.youtube.com/watch?v=SZSvcy4QM8E



CONHEÇA O 1º LUGAR - DATAPREV

www.estrategiaconcursos.com.br/blog/entrevista-andre-furtado-aprovado-em-lo-lugar-no-concurso-dataprev-para-o-cargo-de-analistaarea-de-tecnologia-da-informacao



CONHEÇA O 2º LUGAR - ISS/SALVADOR

www.youtube.com/watch?v=vmUlnlJ-aqQ



CONHEÇA NOSSO GRUPO NO FACEBOOK

www.facebook.com/groups/EstrategiaConcursosdeTI



AS AULAS E AS DICAS...

<p>Parágrafos pequenos</p> <p>Observem que os parágrafos têm, no máximo, cinco linhas. Isso serve para que a leitura não fique cansativa e para que vocês não desanimem no meio do material! Para tal, eu tento dividir as disciplinas de maneira que as aulas fiquem objetivas e pequenas (em termos de teoria), mas extensa (em termos de exercícios).</p>	<p>Visão Geral</p> <p>Não se atenham a detalhes antes de entender o básico. <i>Por que?</i> Ora, não há nada mais irritante do que ir para uma prova que vai cair, por exemplo, RUP, saber vários detalhes, mas não saber as fases e disciplinas. Portanto, caso estejam iniciando os estudos sobre uma matéria, foquem em saber o básico para depois se especializarem.</p>
<p>Destaques em vermelho</p> <p>Quase todos os parágrafos possuem alguma palavra ou frase destacada em negrito e em vermelho. Isso ocorre por suas razões: primeiro, para enfatizar alguma informação importante; segundo, para facilitar a leitura vertical, i.e., após uma primeira leitura, a segunda pode ser passando apenas pelos pontos em destaque.</p>	<p>Façam muitos exercícios</p> <p>Ler várias bibliografias é muito trabalhoso e, geralmente, não vale o custo-benefício. Acredito que o que funciona mesmo é entender o básico, depois fazer muitos exercícios e, eventualmente, caso encontrarem algo que não souberem, pesquisem-no separadamente. Além disso, você vai pegando as “manhas” da banca.</p>
<p>Linguagem natural</p> <p>Essa é uma aula para ser lida, o que por si só já pode ser cansativo. Tentarei colocar a linguagem mais coloquial possível, simulando uma conversa. Caso virem frases ou palavras em itálico, ou é uma palavra estrangeira ou é a simulação de uma conversa com vocês. <i>Tranquilo?</i></p>	<p>Façam resumos</p> <p>Essa dica somente serve caso vocês tenham disponibilidade. Caso haja pouco tempo para estudar ou pouco tempo até a prova, não compensa! Se não, façam resumos organizados, pois eles economizarão um bom tempo de estudo em suas próximas provas e sempre que descobrirem novas informações, insiram-nas no resumo.</p>
<p>Diversas figuras</p> <p>Essas aulas estarão em constante evolução, sempre à procura de explicar as matérias de maneira mais compreensível e com novas informações/questões. Para tal, na minha opinião, é fundamental a utilização de figuras, gráficos, painéis, etc. Em minha experiência, é bem mais fácil memorizar a partir de imagens.</p>	<p>Revisem antes da prova</p> <p>Não adianta querer estudar coisas novas até o último minuto antes da prova e não revisar o que estudou há um mês. Vocês irão esquecer e irão se irritar na hora da prova por não lembrarem de conceitos simples. Tirem uma semana para revisar seus resumos, decorarem algumas coisas e, certamente, irão mais confiantes para a prova.</p>
<p>Fazer Exercícios</p> <p>Muitos exercícios é o meio pelo qual vocês se situarão. <i>Como assim, professor?</i> É na hora de fazer os exercícios que vocês descobrirão se estão bem ou mal e avaliarão se precisam estudar mais ou menos. Para tal, há um quadrinho ao final de cada bloco de exercícios para vocês anotarem a quantidade de questões respondidas corretamente ou incorretamente.</p>	<p>Simulado Final</p> <p>Ora, fazer um bloco de questões depois de estudar a teoria é tranquilo. No entanto, lembrem-se que a memória de vocês não é infinita e vocês têm um milhão de outras coisas para estudar e decorar. Portanto, se possível, ao fim do curso faremos um simulado com questões escolhidas que foram comentadas dentro das aulas.</p>



Vamos falar sobre Lógica de Programação! *Em primeiro lugar, por que chamamos de lógica?* Porque é necessário utilizar a lógica para resolver um problema computacional. *Como assim?* **Precisamos de um encadeamento ou uma sequência de pensamentos para alcançar um determinado objetivo.** Nós podemos descrever esses pensamentos como uma sequência de instruções ou passos.

Professor, o que seria uma instrução? **É um conjunto de regras ou normas definidas para a realização ou emprego de algo, indicando ao computador uma ação elementar a ser executada.** Galera, um computador não pensa, ele é burro, ele recebe ordens e executa! O programador de computador é o camarada que vai dar as ordens. Quando temos um conjunto de instruções, elas formam um algoritmo:

Algoritmo: conjunto predeterminado e bem definido de passos destinados à solução de um problema, com um número finito de etapas.



Professor, você pode dar um exemplo? Sim, o exemplo mais comum da bibliografia é mostrado acima: uma receita de bolo. Observem que para fazer um bolo (solucionar um problema), é necessário seguir uma sequência de passos finitos e predeterminados. **No fim das contas, grosso modo, um software nada mais é do que a representação de um algoritmo.**

Professor, todos os programas que eu utilizo no meu computador são representações de algoritmos? Sim! *Inclusive o joguinho de Paciência que eu curto?* Sim! *Mas até mesmo os apps que eu utilizo no celular?* Eles também! **Todos os softwares (de desktop, notebook, smartphone, tablet, geladeira, relógio, foguete da NASA, entre outros) são representações de algoritmos.**

Então, basta que eu escreva um conjunto de passos em qualquer língua que o meu computador realiza a tarefa que eu quiser? Claro que não! Computadores não entendem, por exemplo, português – eles entendem 0 e 1 (na verdade meeeeeesmo, eles entendem presença ou ausência de tensão elétrica), **portanto é necessário representar esses algoritmos por meio de uma linguagem de programação.**

```
1 // class declaration
2 public class ProgrammingExample {
3
4     // method declaration
5     public void sayHello() {
6
7         // method output
8         System.out.println("Hello World!");
9     }
10 }
```

Como assim, professor? Pessoal, um computador é uma grande calculadora. No entanto, ele é “burro”, ele só calcula o que o mandam calcular. **As linguagens de programação surgem como uma solução para abstrair a comunicação entre seres humanos e computadores.** Na imagem ao lado, a ordem do programador era: “Computador, escreva na tela: Hello World!”.

Bem, acho que todo mundo já ouviu falar alguma vez na vida em Código-Fonte. Se você não sabe o que é um Código-Fonte, abra o Internet Explorer ou Google Chrome ou Mozilla Firefox, entre em qualquer site que você queira e pressione a Tecla F12. Pronto, você verá o Código-Fonte por trás do site bonitinho que você está vendo...

Todo software ou site possui um código-fonte, que é um conjunto de palavras organizadas de acordo com regras específicas, formando um ou mais algoritmos. Essas palavras que formam o algoritmo são escritas utilizando uma linguagem de programação. Esse código-fonte é traduzido e posteriormente executado pelo usuário.

Pessoal... se eu não souber uma linguagem de programação, eu posso escrever um algoritmo utilizando um pseudocódigo ou pseudolinguagem! *O que é isso?* É uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples sem necessidade de conhecer a sintaxe de nenhuma linguagem de programação. **Trata-se de um pseudocódigo, logo não pode ser executado em um sistema real.**

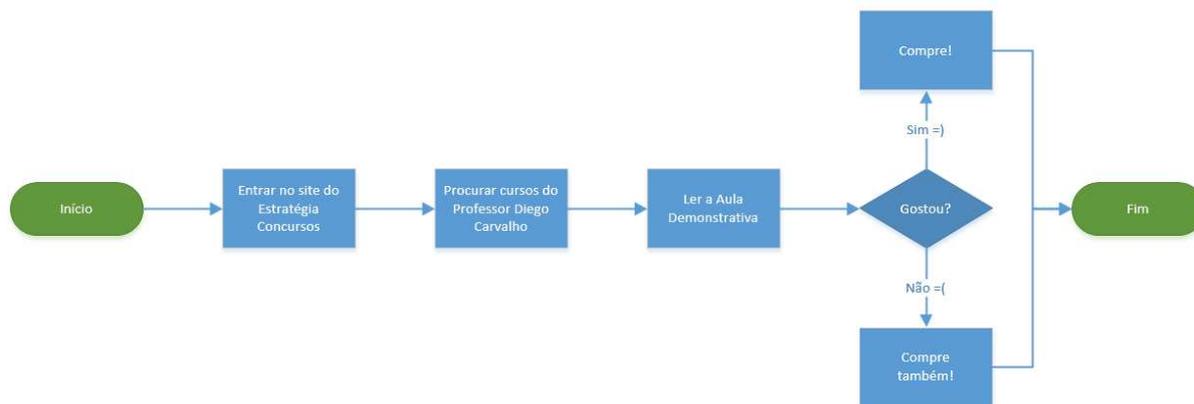
Um tipo de pseudocódigo é o Portugol (ou Português Estruturado)! Trata-se de uma simplificação extrema da língua portuguesa, limitada a pouquíssimas palavras e estruturas que têm significado pré-definido, na medida em que deve seguir um padrão. **Emprega uma linguagem intermediária entre a linguagem natural e a linguagem de programação para descrever os algoritmos.**

Embora o Portugol seja uma linguagem bastante simplificada, possui todos os elementos básicos e uma estrutura semelhante à de uma linguagem de programação de computadores. Portanto **resolver problemas com português estruturado pode ser uma tarefa tão complexa quanto a de escrever um programa em uma linguagem de programação qualquer.** Olha um exemplo:

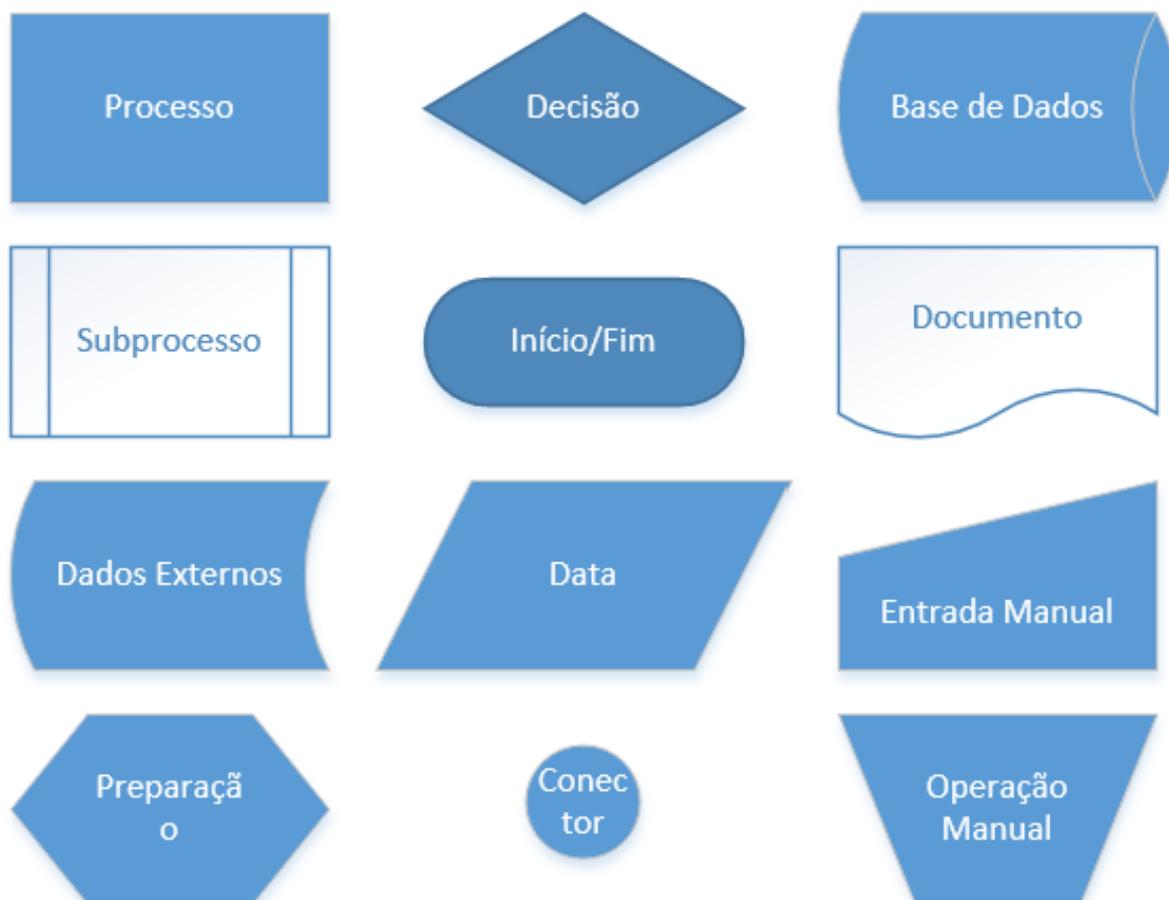
```
início
  <instruções>

  se <teste> então
    <instruções>
  senão
    <instruções>
  fim_se
fim
```

Além do português estruturado, é possível representar um algoritmo também por meio de um Fluxograma! *O que é isso, professor?* É uma espécie de diagrama utilizado para documentar processos, ajudando o leitor a visualizá-los, compreendê-los mais facilmente e encontrar falhas ou problemas de eficiência, como mostra a imagem abaixo.



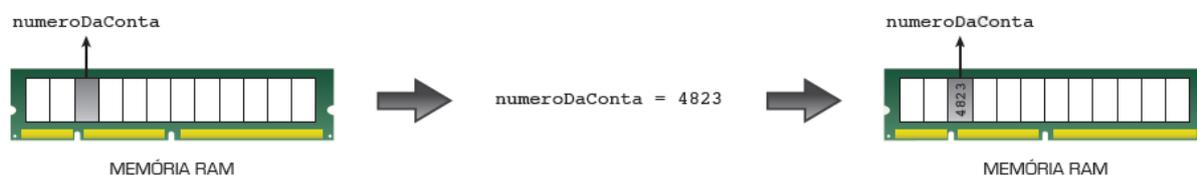
Os principais símbolos de um Fluxograma são:



Bem, nós vimos então que podemos representar algoritmos por meio de Linguagens de Programação, Pseudocódigos ou Fluxogramas! Em geral, os fluxogramas são mais utilizados para leigos; pseudocódigo para usuários um pouco mais avançados; e linguagens de programação para os avançados. **Agora vamos falar de conceitos mais específicos: constantes, variáveis e atribuições!**

Constantes são dados que simplesmente não variam com o tempo, i.e., possuem sempre um valor fixo invariável. Por exemplo: a constante matemática π é (sempre foi e sempre será) igual a 3.141592 – esse valor não mudará! *Professor, e o que seria uma variável?* São espaços na memória do computador reservados para guardar informações ou dados que podem variar.

Em geral, variáveis são associadas a posições na Memória RAM, armazenando diversos tipos de dados que veremos com detalhes à frente! Ela possui um nome identificador que abstrai o nome do endereço na memória que ela ocupa. Observem a imagem abaixo: existe uma variável (espaço em memória) chamada `numeroDaConta` em que se armazena o valor 4823.

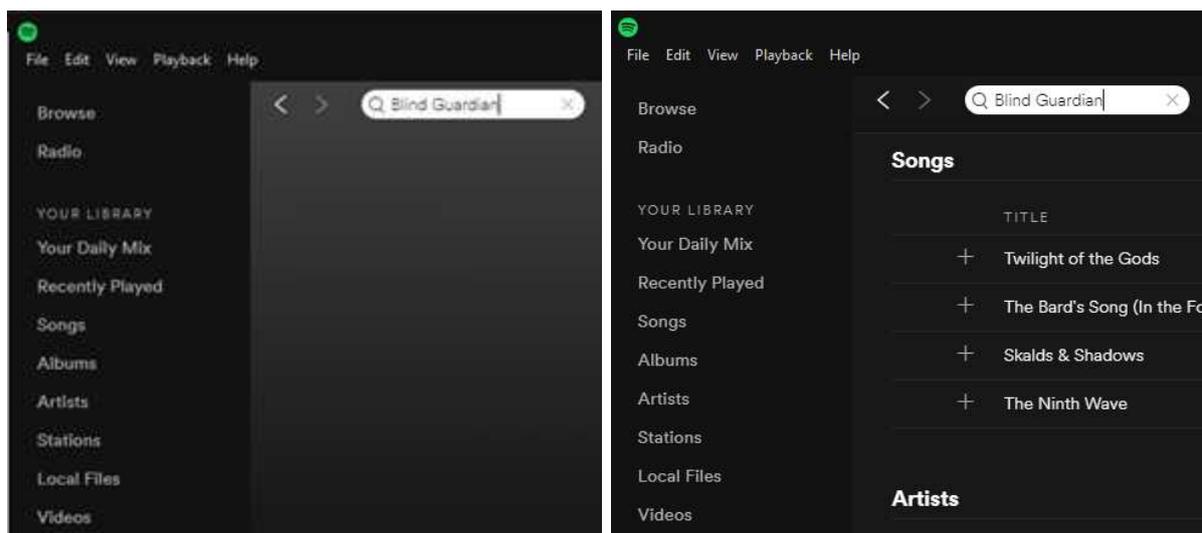


O conteúdo de uma variável pode ser alterado, consultado ou apagado diversas vezes durante a execução de um algoritmo, porém o valor apagado é perdido. *Bacana, mas e a atribuição?* Bem, **trata-se de uma notação para associar um valor a uma variável, i.e., armazenar o conteúdo no endereço de memória específico.** Cada linguagem de programação adotará uma maneira de representá-la.

Galera, nós não vamos nos prender a uma linguagem de programação específica, vamos adotar o Português Estruturado, também conhecido – como dito anteriormente – **Portugol**. *Como é a notação de atribuição?* A notação de atribuição é representada por uma seta apontando para a esquerda do valor para o identificador, podendo ser:

```
Constante: dataDeNascimento ← 1988  
Variável:  endereço ← cidade  
Expressão: idade ← (anoAtual - anoDeNascimento)
```

E a Entrada/Saída de dados? Galera, um modelo computacional é baseado em uma **ENTRADA → PROCESSAMENTO → SAÍDA**. Entradas e saídas fazem parte da interação do programa com o mundo real, i.e., a forma com que o programa recebe os dados a serem processados do mundo real e devolve uma resposta. Esse é um conceito básico da primeira aula de um curso de computação.



Vamos falar agora sobre tipos de dados! Em geral, eles se dividem **em dois grupos: Dados Elementares e Dados Estruturados**. Só uma informação antes de começar: os dados elementares também podem ser chamados de simples, básicos, nativos ou primitivos. Já os dados estruturados também podem ser chamados de compostos. *Bacana? Vamos para as definições!*

- **Tipos Elementares:** são aqueles que não podem ser decompostos. *Ora, se eu disser que o Pedrinho tem 10 anos, é possível decompor esse valor de idade? Não, logo é um tipo elementar.* Há diversos tipos elementares, dependendo da linguagem de programação utilizada. No entanto, os principais são:
 - **Inteiro:** também conhecido como Integer, são similares aos números inteiros da matemática, i.e., sem parte fracionária. Podem ser positivos, negativos ou nulos. Ex: -2% de crescimento do PIB; 174 km de distância; 0 °C de Temperatura; etc.
 - **Real:** também conhecido como Float (Ponto Flutuante), são similares aos números reais da matemática, i.e., possuem parte fracionária. Ex: 3,141592 é a constante de PI; 9,81 m/s² de Aceleração Gravitacional; raiz quadrada de 7; etc.
 - **Caractere:** também conhecido como Literal ou Char, são representações de letras, dígitos e símbolos. Quando colocadas em conjunto, formam um tipo estruturado chamado String ou Cadeia de Caracteres. Ex: 'a', '\$', '5', 'D', etc.

- **Lógico:** também conhecido como Boolean, são representações de valores lógicos – verdadeiro/falso, ligado/desligado, sim/não, etc. São extremamente importantes na programação, principalmente na verificação de condições.
- **Tipos Estruturados:** são aqueles que podem ser decompostos. *Ora, se eu disser que o nome da bola da copa é Brazuca, é possível decompor esse nome? Sim, basta dividi-lo em caracteres: 'B', 'r', 'a', 'z', 'u', 'c', 'a'. Há infinitos tipos estruturados¹, pois eles são a combinação de vários outros, o mais comum é:*
 - **Cadeia de Caracteres:** também conhecido como String, são representações de sequências de caracteres, incluindo ou não símbolos. Pode ser uma palavra, frase, código, etc, por exemplo: "O rato roeu a roupa do rei de Roma".

Pessoal, quase todas as linguagens de programação possuem instruções para **Leitura e Escrita de dados**. A Leitura é uma Entrada de Dados (Input) que busca ler dados do usuário por meio teclado geralmente (Ex: busca do Google). A Escrita é uma Saída de Dados (Output) que busca mostrar informações ao usuário na tela do computador (Ex: resultado da busca do Google).

Pesquisa Google

Estou com sorte



Descubra de onde vem a magia de Harry Potter

Disponibilizado pelo Google em: [English](#)

¹ Uma música em .mp3, um texto em .pdf, uma imagem em jpg são todos tipos estruturados.

Todas Imagens Vídeos Shopping Notícias Mais Configurações Ferramentas

Aproximadamente 34.300.000 resultados (0,41 segundos)

Entrada/saída, sigla E/S (em inglês: Input/output, sigla I/O) é um termo utilizado quase que exclusivamente no ramo da computação (ou informática), indicando **entrada** (inserção) de **dados** por meio de algum código ou programa, para algum outro programa ou hardware, bem como a sua saída (obtenção de **dados**) ou retorno de ...

[Entrada/saída – Wikipédia, a enciclopédia livre](https://pt.wikipedia.org/wiki/Entrada/saída)

<https://pt.wikipedia.org/wiki/Entrada/saída>

 Sobre este resultado  Feedback

[Entrada/saída – Wikipédia, a enciclopédia livre](https://pt.wikipedia.org/wiki/Entrada/saída)

<https://pt.wikipedia.org/wiki/Entrada/saída> ▼

Entrada/saída, sigla E/S (em inglês: Input/output, sigla I/O) é um termo utilizado quase que exclusivamente no ramo da computação (ou informática), indicando **entrada** (inserção) de **dados** por meio de algum código ou programa, para algum outro programa ou hardware, bem como a sua saída (obtenção de **dados**) ou ...

Professor, e como eu faço para manipular dados? **Temos alguns operadores:**

Operadores Aritméticos: são utilizados para obter resultados numéricos, preocupando-se com a priorização².

Operador	Símbolo	Prioridade
Multiplicação	*	2º
Divisão	/	2º
Adição	+	3º
Subtração	-	3º
Exponenciação	^	1º

Operadores Relacionais: são utilizados para comparar números e literais, retornando valores lógicos.

Operador	Símbolo
Igual a	=
Diferente de	<> ou !=

² Em operadores que possuem a mesma prioridade, o que aparecer primeiro deve ser priorizado! Além disso, parênteses possuem sempre a maior prioridade!

Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Operadores Lógicos: servem para combinar resultados de expressões, retornando valores lógicos (verdadeiro ou falso).

Operador/Símbolo
E/And
Ou/Or
Não/Not

Por fim, antes de passarmos para as estruturas de controle, vamos entender o que é um **Bloco de Comandos**. Um Bloco de Comandos é um conjunto de comandos limitados por dois delimitadores que marcam o início e o fim do bloco. O bloco pode um ou mais comandos (quando tem apenas um, é opcional utilizar os dois delimitadores) – geralmente é: {...} ou `begin ... end`.

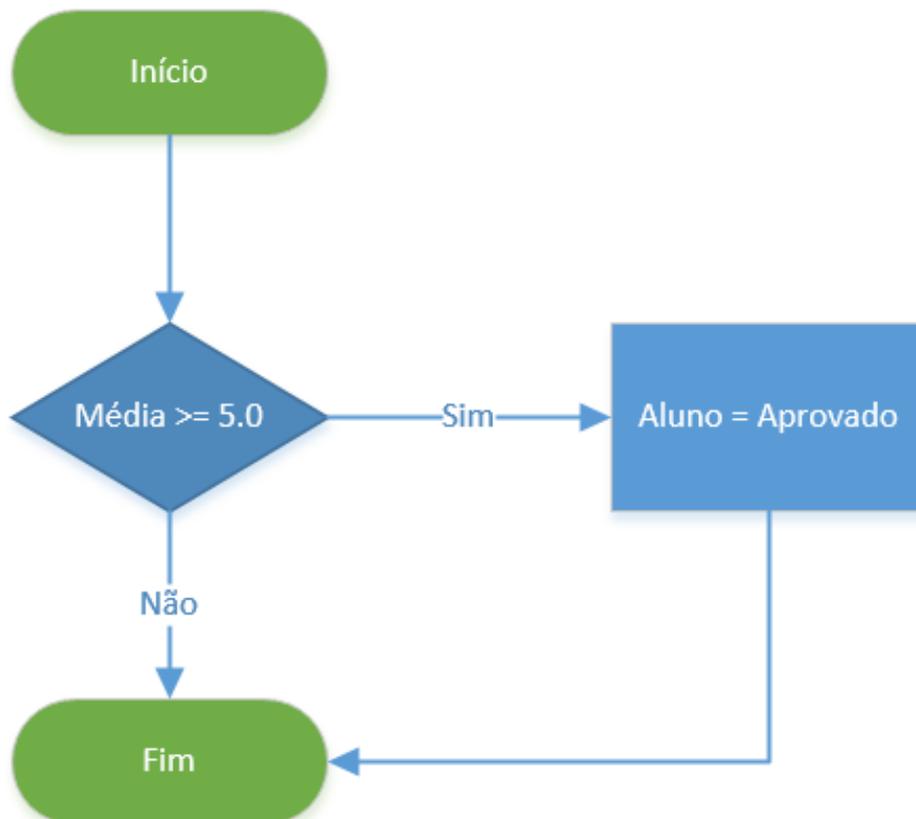
ESTRUTURAS DE CONTROLE: DECISÃO E REPETIÇÃO

Bacana! Com isso, já podemos falar sobre Estruturas de Decisão e Repetição! Galera, as Estruturas de Decisão (também chamadas de Estruturas de Seleção, inclusive no edital) permitem interferir na sequência de instruções executadas dependendo de uma condição de teste, i.e., o algoritmo terá caminhos diferentes para decisões diferentes. Bacana?

O contrário dessa afirmação é a execução sequencial das instruções, sem loops ou desvios. Quando terminarmos, todos devem saber cantar "Hey Jude" na ordem certinha, seguindo os comandos que nós estamos aprendendo no fim da aula! Beleza? A melhor maneira de se visualizar uma estrutura de decisão é com o uso de fluxogramas. Então, vamos ver...

CASO 1:

Se (Média \geq 5.0) Então
Aluno = Aprovado

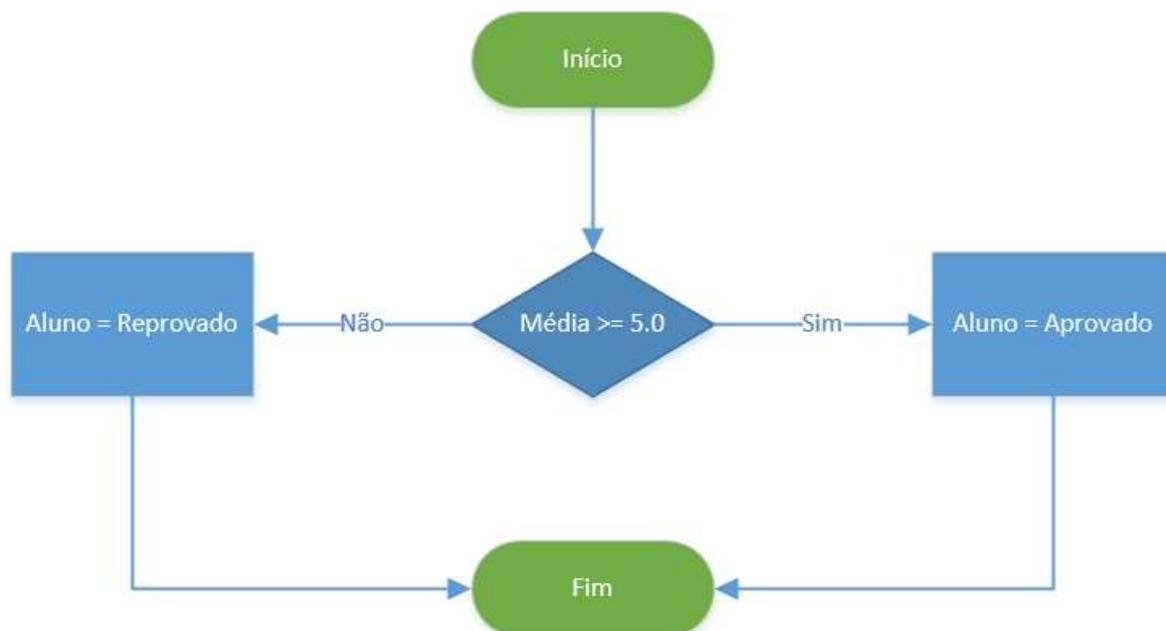


O primeiro caso, e mais simples, é uma estrutura de decisão com um teste (Média ≥ 5.0). O programa avalia se a variável Média tem o valor maior ou igual a 5.0, no caso de o teste ser verdadeiro a instrução "Aluno = Aprovado" é executada. Em caso negativo, o programa pula a instrução "Aluno = Aprovado" e continua logo após o final do bloco de decisão.

Quando uma estrutura de decisão (seleção) possui somente o bloco "Se Então", é chamada de simples. Isso já foi cobrado em provas da FCC, galera! Prestem bastante atenção nesse quesito! Algumas provas também cobram os nomes das estruturas em inglês, ou seja, ao invés de "Se Então", eles também podem cobrar como "If Then". *Tudo tranquilo?*

CASO 2:

```
Se (Média  $\geq 5.0$ ) Então
    Aluno = Aprovado
Senão
    Aluno = Reprovado
```

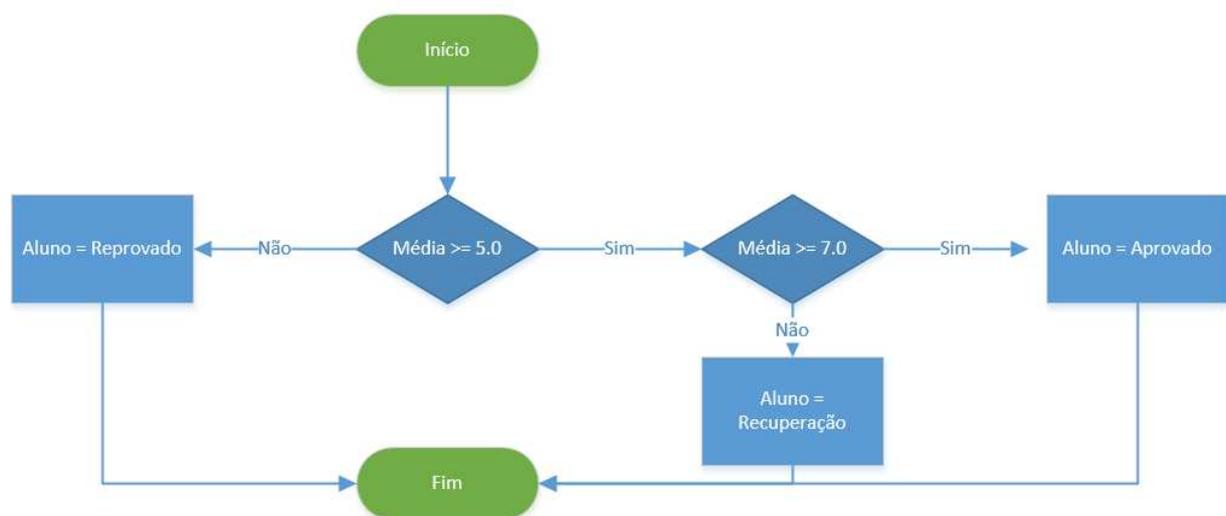


O segundo caso é só um pouquinho mais complexo, onde temos dois fluxos possíveis, um que passa pelo comando "Aluno = Aprovado" e outro que passa pelo "Aluno = Reprovado". **Importante ressaltar que, nesse caso, sempre alguma dessas instruções serão executadas**, pois, ou Média é maior ou igual a 5.0 ou é menor, sempre. *Bacana?*

Esse exemplo de estrutura de decisão possui os blocos Se-Então-Senão. **Quando a estrutura de decisão possui todos os blocos (Se-Então-Senão), ela é chamada de composta.** Em inglês a estrutura composta é conhecida como "If-Then-Else". Então, nós vimos já o Se-Então, agora vimos o Se-Então-Senão e agora vamos ver o Se-Então-Senão dentro de outro Se-Então-Senão.

CASO 3:

```
Se (Média >= 5.0) Então
  Se (Média >= 7.0) Então
    Aluno = Aprovado
  Senão
    Aluno = Recuperação
Senão
  Aluno = Reprovado
```



Vejam o código anterior! Nesse caso, o programa realiza um primeiro teste (Média >= 5.0). No caso de a Média ser menor que 5.0, o comando "Aluno = Reprovado" será executado e o fluxo termina. **Caso contrário, ou seja, a Média sendo maior ou igual a 5.0 executamos os comandos do bloco "Então", que, nesse caso possui mais uma estrutura de decisão.**

O segundo teste avalia a expressão "Média >= 7.0". **Chamamos esse tipo de utilização em diferentes níveis de estrutura de decisão (seleção) aninhada.** Antes de aprendermos a nossa última estrutura de decisão (seleção), "Caso Selecione",

temos que nos atentar para o fato de que a condição testada nas estruturas de decisão não precisa ter uma expressão somente.

Qualquer uma que retorne um valor verdadeiro ou falso é válida como condição de teste. Expressões como (Média > 10 ou Média < 5) ou ainda (Aluno == Reprovado e Média < 3.0 ^ NumeroFaltas > 5) poderiam ser utilizadas. *Sabe uma utilizada muitas vezes em programação? (1 = 1). Isso mesmo, é óbvio, mas é muito utilizado em programação para algumas coisas específicas – é óbvio 1 = 1 é verdadeiro.*

CASO 4:

Selecione (Número)

Caso 13: Presidente = "Dilma"

Caso 20: Presidente = "Pastor Everaldo"

Caso 28: Presidente = "Levy Fidelix"

Caso 43: Presidente = "Eduardo Jorge"

Caso 45: Presidente = "Aécio Neves"

Caso 50: Presidente = "Luciana Genro"

Caso Outro: Presidente = "Nulo"

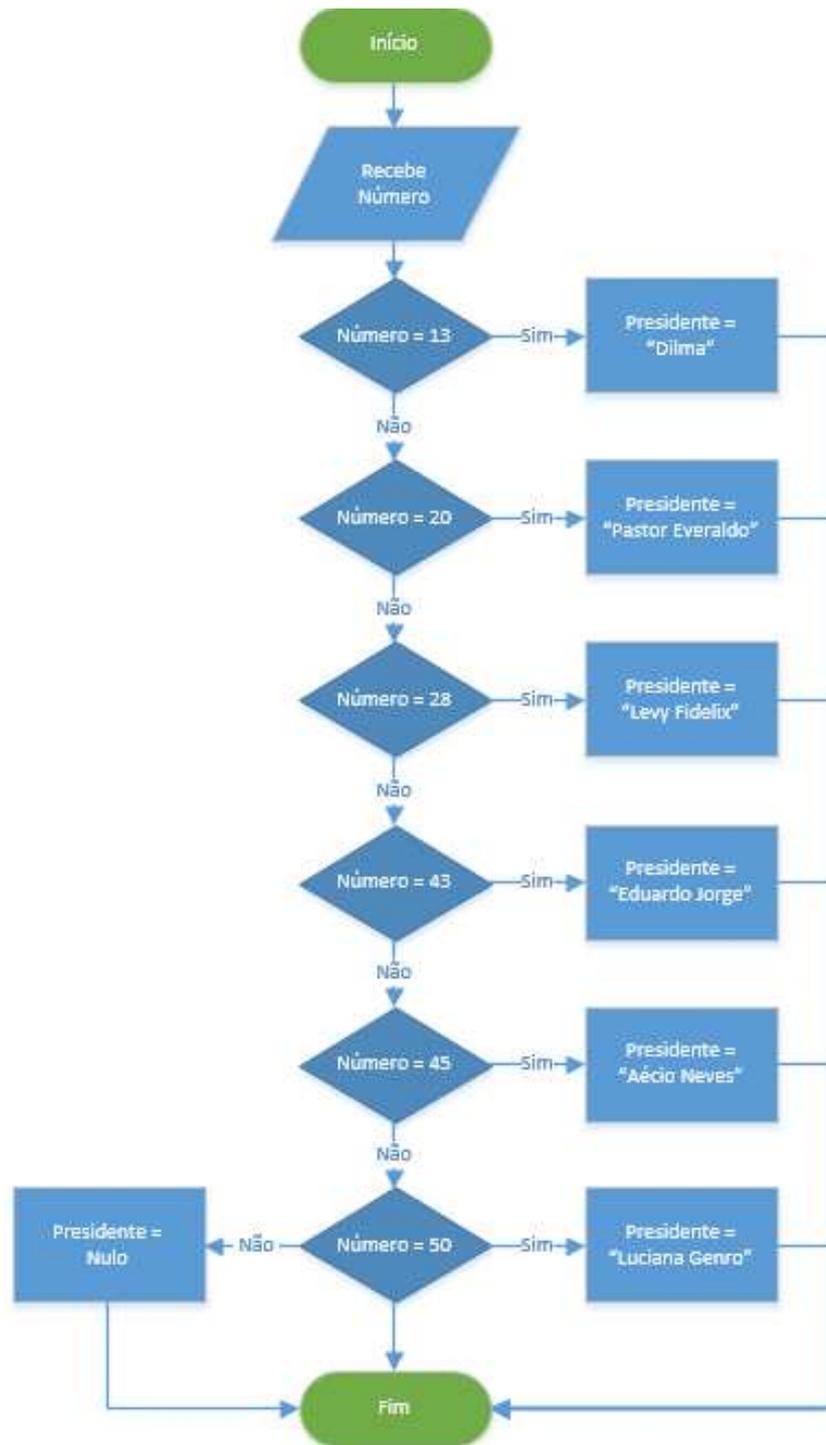
Fim Seleccione

A estrutura "Caso-Seleccione" também é conhecida como de decisão múltipla escolha. Diferente das outras que vimos, não parte de um teste de condição para definir para onde o fluxo deve ser desviado, mas sim avalia o valor de uma variável e dependendo de qual seja o conteúdo, encaminha para o rótulo indicado e executa as instruções ali apresentadas.

Temos, então, quatro Estruturas de Decisão (Seleção): Se-Então, Se-Então-Senão, Se-Então (Aninhados) e Caso-Seleccione.³

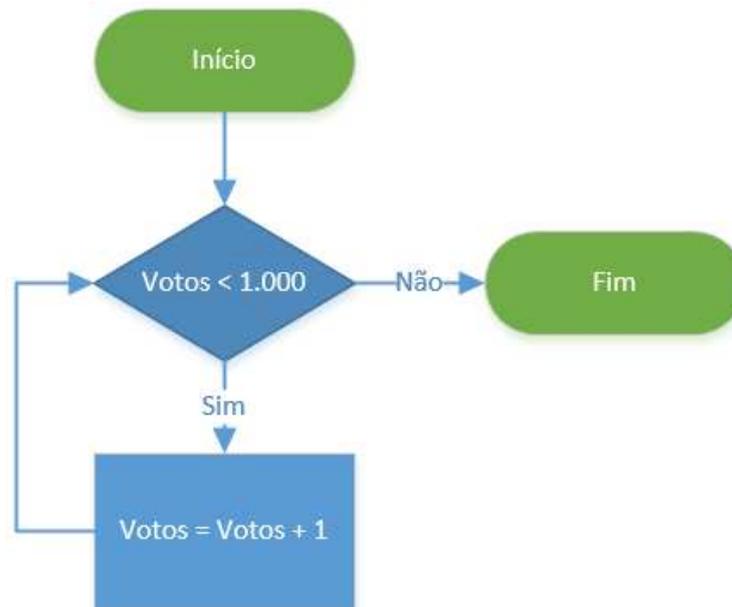
Agora veremos as famosas Estruturas de Repetição! Essas estruturas são utilizadas quando se deseja que um determinado conjunto de instruções ou comandos sejam executados por mais de uma vez. A quantidade pode ser definida, indefinida, ou enquanto um estado se mantenha ou seja alcançado. Parece confuso, mas não é! Vamos ver, pessoal...

³ Em inglês: If-Then, If-Then-Else, If-Then (nested) e Switch-Case.



CASO 1: REPETIÇÃO PRÉ-TESTADA (TESTA-SE ANTES DE PROCESSAR!)

Enquanto (Votos < 1.000) Faça
 Votos = Votos + 1
Fim-Enquanto



Esse primeiro tipo de estrutura de repetição realiza um teste antes de executar qualquer instrução dentro de seu bloco. Desse modo, as instruções podem nem ser executadas, caso o teste da condição inicial falhe. No exemplo acima, se a variável "Votos" for maior ou igual a 1.000, a instrução "Votos = Votos + 1" não é executada uma vez sequer.

A variável sendo testada deve sofrer alguma alteração dentro do bloco da estrutura de repetição **sob pena de a execução não ter fim**. Exemplo:

```
Votos = 500
```

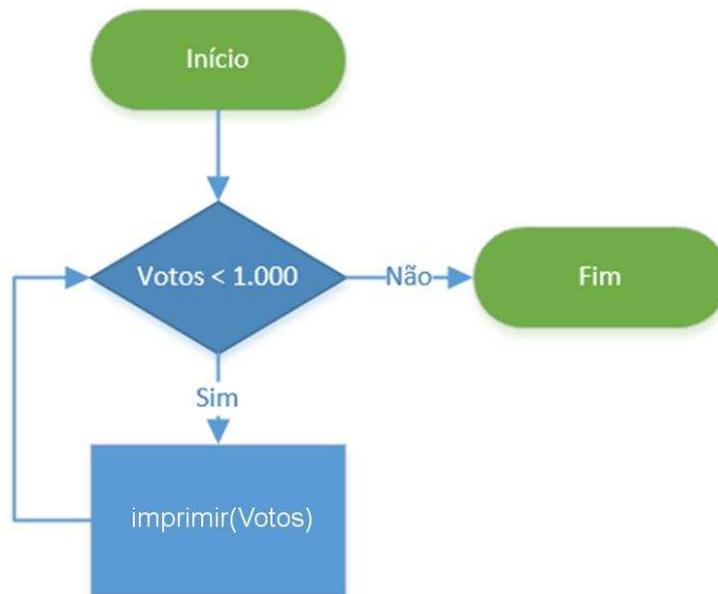
```
Enquanto (Votos < 1.000) Faça  
    imprimir("Eu nunca mais vou sair daqui! Muahahaha")  
Fim-Enquanto
```

A variável votos tem o valor estipulado em 500, ou seja, é menor que 1.000. Quando é executado o primeiro teste "Votos < 1000" o resultado é verdadeiro, ou seja, o fluxo é direcionado para dentro do bloco e a instrução "imprimir" é executada. Na segunda repetição, como não houve alteração da variável "Voto" vai executar novamente a instrução "imprimir" e assim por diante.

Dá a obrigatoriedade de se alterar o valor da variável utilizada no teste de condição de entrada e saída da estrutura. Em inglês, a estrutura é "While Do".

CASO 2: REPETIÇÃO COM VARIÁVEL DE CONTROLE

Para Votos de 1 até 1000 Faça
imprimir(Votos)
Fim-Enquanto



Esse tipo de estrutura é um pouco diferente da primeira, já que define de antemão quantas vezes será repetida as instruções dentro do bloco. No nosso exemplo, a instrução "imprimir (Votos)" será executada 1000 vezes, pois assim determina a expressão "Para Votos de 1 até 1000 Faça". Essa expressão é simplificada, mas realiza várias instruções internas:

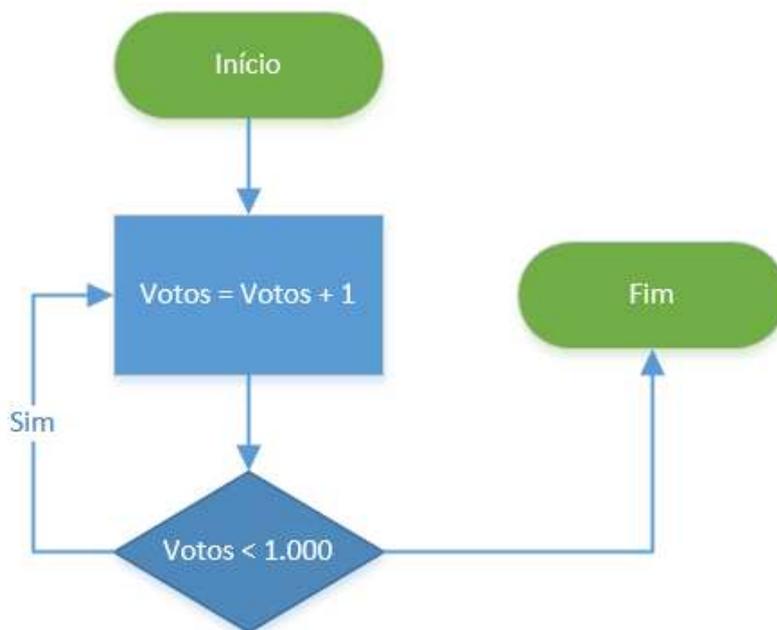
- Testa se Votos é igual a 1000;
- Se é igual, encerra a repetição;
- Se é menor, soma 1 à variável de controle "Votos";
- Mais uma repetição

Perceba que, como não há outros testes de condição de saída a não ser a contagem da variável de controle, **as instruções que fazem parte do bloco de execução da estrutura de repetição não precisam alterar a variável de controle**, pois a própria estrutura o faz. É uma grande diferença para os casos de estrutura de repetição pré-testada e pós-testada, que necessitam de tal alteração.

CASO 3: REPETIÇÃO PÓS-TESTADA (TESTA-SE DEPOIS DE PROCESSAR!)

Faça

Votos = Votos + 1
Enquanto (Votos < 1.000)



O último caso é o pós-testado. **Como sugere o nome, nessa estrutura a instrução do bloco é executada sempre ao menos uma vez!** Isso porque o primeiro teste somente é feito ao final. Em inglês essa estrutura é conhecida como "Do While". Agora que sabemos tudo sobre estruturas de seleção (ou decisão) e de repetição, que tal tentar um exemplo de código que imprime a letra de "Hey Jude" dos Beatles.

Fomos inspirados pelo fluxograma abaixo, **que destaca muito bem as repetições e decisões para saber que parte da letra cantar.** *Vamos nessa?*

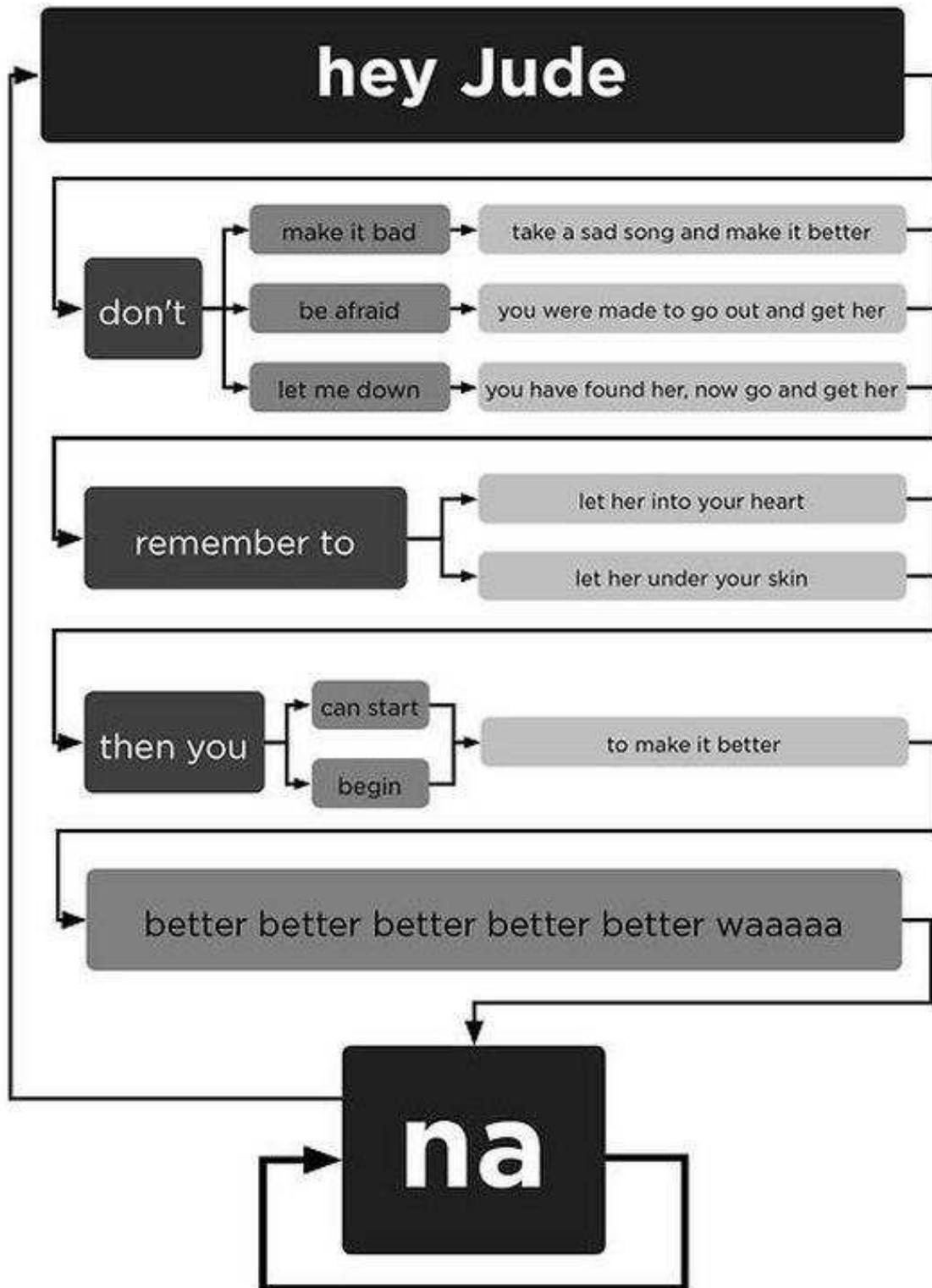
```
01 estrofe = 0
02
03 Enquanto (estrofe < 3) Repetir
04     estrofe == estrofe + 1
05     imprimir("Hey Jude, don't ")
06
07     Se (estrofe == 1) Então
08         imprimir("make it bad")
09         imprimir("take a sad song and make it better")
10     Senão Se (estrofe == 2) Então
11         imprimir("be afraid")
12         imprimir("you were made to go on and get her ")
13     Senão Se (estrofe == 3) Então
14         imprimir("let me down")
15         imprimir("you have found her, now go and get her")
16
```

```
17 imprimir("remember to let her ")
18
19 // Teste se estrofe é par ou ímpar
20 Se (estrofe % 2 == 1) Então
21     imprimir("into your heart")
22 Senão
23     imprimir("under your skin")
24
25 imprimir("then you ")
26
27 // Teste se estrofe é par ou ímpar
28 Se (estrofe % 2 == 1) Então
29     imprimir("can start")
30 Senão
31     imprimir("begin")
32
33 imprimir("to make it better")
34
35 Se (estrofe == 3) Então
36     imprimir("better better better better BETTER WAAAAAAAAAAAA")
37     Repetir
38         imprimir("NA NA NA NANANA NAAAAAAA")
39         imprimir("NANANA NAAAAAAA")
40         imprimir("Hey Jude")
41 Enquanto(Verdadeiro)
```

A música começa com uma estrutura que conhecemos, a repetição pré-testada (linha 03). **A condição de execução é se estrofe é menor que 3, isso porque somente temos três estrofes na música.** A seguir vemos uma estrutura de decisão composta (linha 07), todas avaliando a variável "estrofe". Para cada um dos valores de estrofe (1, 2 e 3) um bloco em separado é executado.

Após o primeiro "Se Então Senão", temos outro, que testa se estrofe é par ou ímpar (linha 28). Essa estrutura também é composta, pois possui um bloco "Senão". Por fim, temos uma estrutura de seleção (decisão) simples (linha 35), ou seja, somente com bloco "Se Então" e, dentro desse, encontramos uma estrutura de repetição pós-testada (linha 37).

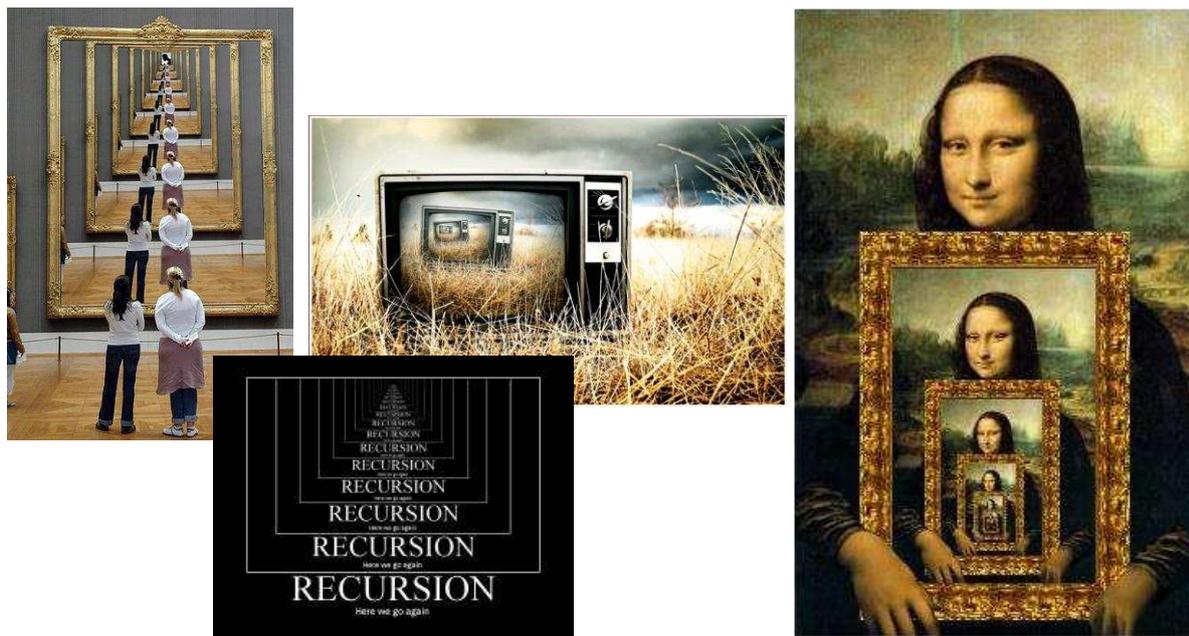
No exemplo lúdico essa repetição nunca termina (*pois ela termina em fade e parece continuar, lembram? :)*), pois a condição "Verdadeiro" sempre está satisfeita. Nos programas reais temos que lançar mão de algum escape. **Todo mundo cantou bem alto para o vizinho ouvir? Ficou difícil? Tirei do diagrama abaixo, assim acho mais fácil de visualizar. Abraços e até a próxima.**



lyrics © sony atv

RECURSIVIDADE

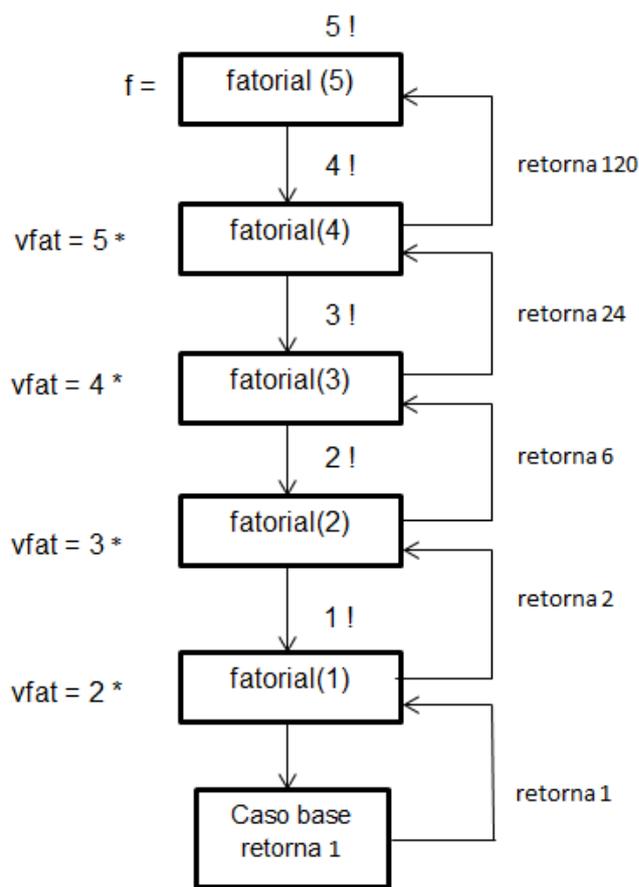
Por fim, vamos falar sobre recursividade! *O que é isso, professor?* Formalmente, é uma maneira de solucionar um problema por meio da instanciação de instâncias menores do mesmo problema. Grosso modo, **a recursão é uma técnica em que uma função chama a si mesma**. Eu gosto de pensar em recursividade por meio de imagens como as apresentadas abaixo.



Nós dissemos que se trata de uma função que chama a si mesma e, daí, temos o nosso primeiro problema. *Se ela chama a si mesma sempre, não entra em um loop infinito?* **Sim, por isso é necessário um caso-base (solução trivial), i.e., um caso mais simples em que é utilizada uma condição de parada, que resolve o problema.** Toda recursividade é composta por um caso base e pelas chamadas recursivas.

Vantagens da recursividade: torna a escrita do código mais simples, elegante e, muitas vezes, menor. Desvantagens da recursividade: quando o loop recursivo é muito grande, é consumida muita memória nas chamadas a diversos níveis de recursão, **tendo em vista que cada chamada recursiva aloca memória para os parâmetros, variáveis locais e de controle.**

Em muitos casos, uma solução iterativa gasta menos memória e torna-se mais eficiente em termos de performance do que usar recursão. *Galera, vocês se lembram que estudaram fatorial na escola?* Pois é, fatorial é uma operação matemática em que um número natural (representado por $n!$) é o produto de todos os inteiros positivos menores ou iguais a n .



fatorial.

Professor, facilita aí. Já saí da escola há muito tempo. Ok! Fatorial(5) = $5 \times 4 \times 3 \times 2 \times 1$; uma maneira diferente de escrever isso é: Fatorial(5) = $5 \times$ Fatorial(4), que é o mesmo que dizer $5 \times (4 \times$ Fatorial(3)), que é o mesmo que $5 \times (4 \times (3 \times$ Fatorial(2))), que é o mesmo que $5 \times (4 \times (3 \times (2 \times$ Fatorial(1))))). E agora, professor? Agora é a vez do caso-base! No caso do fatorial, o caso-base é: Fatorial(1) = 1. **Todas essas instâncias ficaram em memória aguardando a chegada do caso-base e agora retornamos com os resultados.** Dado o caso-base, temos que: $5 \times (4 \times (3 \times (2 \times$ Fatorial(1))))), logo $5 \times (4 \times (3 \times (2 \times 1))) = 120$. Vejam ao lado a imagem da execução de um código representando o

Por fim, gostaria de mencionar que existem dois tipos de recursividade: direta e indireta. De modo bem simples, a recursividade direta é aquela tradicional em que uma função chama a si mesma (Ex: Função A chama a Função A); a recursividade indireta é aquela alternativa em que uma função chama outra função que chama a primeira (Ex: Função A chama a Função B, que chama a Função A).



1. (CESGRANRIO – 2014 – EPE – Tecnologia da Informação) Analise o algoritmo abaixo, onde $a\%b$ representa o resto da divisão de a por b .

```
início
  inteiro x,y,i,r
  ler x
  ler y
  para i de 1 até x
    se (x%i=0) e (y%i=0) então
      r <- i
    fim se
  próximo
  escrever r
fim
```

Qual será a resposta, caso as entradas sejam 128, para x , e 56, para y ?

- a) 2
- b) 8
- c) 56
- d) 64
- e) 128

Comentários:

Galera, vejam como não é difícil. Nós, primeiro temos um comando **início** para indicar o início do algoritmo. Em seguida ele declara 4 variáveis (espaços na memória) para armazenar um dado do tipo **inteiro** e dá o nome de **x**, **y**, **i**, **r**. Depois nós temos o comando **ler** em que ele vai ler uma entrada do usuário e armazenar em **x** e outro comando **ler** em que ele vai ler outra entrada do usuário e armazenar em **y**.

Que valores são esses, professor? Ele diz no enunciado: 128 para x e 56 para y . Dito isso, o próximo comando tem uma estrutura de repetição **para**. E ela diz que para **$i = 1$ até $i = x$** (lembrem-se que $x = 128$), repita o que tem abaixo –

lembrando que no final do comando **para** tem um comando **próximo**, a cada repetição, a variável de controle i é incrementada em 1. Relaxa que é simples de entender: a cada **para** eu aumento o valor de i em 1. Aí, abaixo nós temos outro comando que diz: **se $x\%i = 0$ e $y\%i = 0$** , então $r \leftarrow i$. E ele disse no enunciado que **$a\%b$** representa o resto da divisão de a por b , logo $x\%i$ representa o resto da divisão de x por i . *Então, vamos testar?*

$x = 128$

$y = 56$

$x\%i = 128\%1$. Qual o resto da divisão de 128 por 1? Quociente 128, Resto 0.

$x\%i = 56\%1$. Qual o resto da divisão de 56 por 1? Quociente 56, Resto 0.

Então cumprimos as duas condições do comando **se $x\%i = 0$ e $y\%i = 0$** , logo executamos o comando abaixo: **$r \leftarrow i$** , ou seja, r agora será i , então $r = 1$.

--

Vejam agora o comando próximo. Isso significa que agora $i = 2$ e nós repetimos tudo de novo.

$x\%i = 128\%2$. Qual o resto da divisão de 128 por 2? Quociente 64, Resto 0.

$x\%i = 56\%2$. Qual o resto da divisão de 56 por 2? Quociente 28, Resto 0.

Então cumprimos as duas condições do comando **se $x\%i = 0$ e $y\%i = 0$** , logo executamos o comando abaixo: **$r \leftarrow i$** , ou seja, r agora será i , então $r = 2$.

--

Vejam agora o comando próximo. Isso significa que agora $i = 3$ e nós repetimos tudo de novo.

$x\%i = 128\%3$. Qual o resto da divisão de 128 por 3? Quociente 42, Resto 2.

$x\%i = 56\%3$. Qual o resto da divisão de 56 por 3? Quociente 18, Resto 2.

Então **não** cumprimos as duas condições do comando **se $x\%i = 0$ e $y\%i = 0$** , o resto não é zero para nenhum dos dois casos, logo **não** executamos o comando abaixo: **$r \leftarrow i$** , portanto r continua sendo igual a 2, então $r = 2$.

--

Vejam agora o comando próximo. Isso significa que agora $i = 4$ e nós repetimos tudo de novo.

$x\%i = 128\%4$. Qual o resto da divisão de 128 por 4? Quociente 32, Resto 0.

$x\%i = 56\%4$. Qual o resto da divisão de 56 por 4? Quociente 14, Resto 0.

Então cumprimos as duas condições do comando **se $x\%i = 0$ e $y\%i = 0$** , logo executamos o comando abaixo: **$r \leftarrow i$** , ou seja, r agora será i , então $r = 4$.

Pessoa, nós podemos continuar fazendo na mão até descobri o valor final de r , mas vocês perceberam que esse algoritmo está querendo descobrir o Máximo Divisor Comum (MDC) entre 128 e 56? Ele sempre está decompondo os dois números e fará isso até descobrir qual será o MDC entre eles. Uma vez descoberto isso, eu não preciso fazer na mão. Sabemos 128 é divisível por 1, 2, 4, 8, 16, 32 e 64. Sabemos que 56 é divisível por 1, 2, 4, 7 e 8. Vimos que ambos têm em comum os divisores 1, 2, 4 e 8. Então o MÁXIMO divisor comum é o 8.

Olha que legal! Se você coloca isso em uma linguagem de programação e manda o computador executar, ele sempre vai descobrir o MDC entre quaisquer dois números. E é isso que o computador faz, ele responde a um algoritmo criado por um programador.

Gabarito: B

-
2. (CESGRANRIO – 2014 – PETROBRÁS - Analista de Sistemas) Analise o algoritmo abaixo em português estruturado.

```
algoritmo segredo;
variáveis
  x,y,z : inteiro;
fim-variáveis
início
  x:=15;
  y:=10;
  z:=0;
  enquanto y>0 faça
    z:=z+x;
    y:=y-1;
  fim-enquanto
  imprima(z);
fim
```

Que número seria impresso caso esse programa executasse?

- a) 0
- b) 10
- c) 15
- d) 100
- e) 150

Comentários:

Vamos lá, galera... vimos que o nome do algoritmo é segredo e que ele tem três variáveis: x , y , z do tipo inteiro. No início, ele diz que: $x = 15$; $y = 10$ e $z = 0$. E aí vamos para a estrutura de repetição **enquanto**:

Sabemos que $y = 10$, ele diz: enquanto $y > 0$ repita o que está abaixo. E abaixo está:

$z = z + x$. Logo, $z = 0 + 15 = 15$. Agora $z = 15$.
 $y = y - 1$. Logo, $y = 10 - 1 = 9$. Agora $y = 9$.

E terminamos o **enquanto**, mas lembrem-se que é uma estrutura de repetição. Então, nós só saímos dela quando $y > 0$ (não é o caso, porque $y = 9$). Então, de novo:

$z = z + x$. Logo, $z = 15 + 15 = 30$. Agora $z = 30$.
 $y = y - 1$. Logo, $y = 9 - 1 = 8$. Agora $y = 8$.

E terminamos o **enquanto**, mas lembrem-se que é uma estrutura de repetição. Então, nós só saímos dela quando $y > 0$ (não é o caso, porque $y = 8$). Então, de novo:

$z = z + x$. Logo, $z = 30 + 15 = 45$. Agora $z = 45$.
 $y = y - 1$. Logo, $y = 8 - 1 = 7$. Agora $y = 7$.

Galera, vocês percebem um padrão? Esse algoritmo vai contando de 15 em 15 até que $y > 0$ não seja mais verdadeira. Em outras palavras, ele somará o valor 15, 10 vezes seguida, ou seja, vai fazer $10 \times 15 = 150$. Eu posso continuar fazendo na mão, mas como eu já descobri o padrão, eu já descobri a resposta :)

Gabarito: E

3. (CESGRANRIO – 2014 – BASA - Analista de Sistemas) A saída do algoritmo apresentado abaixo para as entradas 100 e 20, respectivamente, é

```
inicio
inteiro X, Y
Ler X
Ler Y
Enquanto X  $\geq$  Y - 1 faz
X  $\leftarrow$  X - 1
Y  $\leftarrow$  Y + 2 Fim Enquanto
Escrever "saída =", Y - X
Fim
```

- a) -5
- b) -2
- c) 1
- d) 4
- e) 7

Comentários:

Vamos lá! O algoritmo declara duas variáveis do tipo inteiro X e Y. Manda ler X e Y do usuário (o enunciado diz que foram lidos $X = 100$ e $Y = 20$). E vamos para a estrutura de repetição **enquanto**:

Enquanto X \geq Y-1 faça, ou seja, enquanto $100 \geq 19$ faça.

$X \leftarrow X - 1$, ou seja, $X \leftarrow 99$;
 $Y \leftarrow Y + 2$, ou seja, $Y \leftarrow 22$;

--

Enquanto X \geq Y-1 faça, ou seja, enquanto $99 \geq 21$ faça.

$X \leftarrow X - 1$, ou seja, $X \leftarrow 98$;
 $Y \leftarrow Y + 2$, ou seja, $Y \leftarrow 24$;

--

Enquanto X \geq Y-1 faça, ou seja, enquanto $98 \geq 21$ faça.

$X \leftarrow X - 1$, ou seja, $X \leftarrow 97$;

$Y \leftarrow Y + 2$, ou seja, $Y \leftarrow 26$;

Já descobriram o padrão? X vai sendo decrementado de 1 em 1; e Y vai sendo incrementado de 2 em 2. Quando $X \geq Y-1$ não for mais verdadeiro, sai do enquanto e escreve a saída. Temos então as seguintes combinações:

X = 100, Y = 20;
X = 99, Y = 22;
X = 98, Y = 24;
X = 97, Y = 26;
X = 96, Y = 28;
X = 95, Y = 30;
X = 94, Y = 32;
X = 93, Y = 34;
X = 92, Y = 36;
X = 91, Y = 38;
X = 90, Y = 40;
X = 89, Y = 42;
X = 88, Y = 44;
X = 87, Y = 46;
X = 86, Y = 48;
X = 85, Y = 50;
X = 84, Y = 52;
X = 83, Y = 54;
X = 82, Y = 56;
X = 81, Y = 58;
X = 80, Y = 60;
X = 79, Y = 62;
X = 78, Y = 64;
X = 77, Y = 66;
X = 76, Y = 68;
X = 75, Y = 70;
X = 74, Y = 72;
X = 73, Y = 74;
X = 72, Y = 76;

Observem que nesse ponto, $X \geq Y - 1$ não é mais verdadeiro. *Por que?* Porque $X = 72$ e $Y = 76$, então $72 \geq 76-1$ é falso porque 72 é MENOR que 75. Então nós podemos sair da estrutura de repetição **enquanto** e escrever a saída, que ele pede que seja $Y-X$. $Y = 76$ e $X = 72$, logo $Y-X = 4$.

Gabarito: D

4. (CESGRANRIO – 2010 – PETROBRÁS – Técnico em Informático) Relacionado à programação de computadores, um algoritmo, seja qual for a sua complexidade e a linguagem de programação na qual será codificado, pode ser descrito por meio da:

- a) reografia.
- b) criptografia.
- c) linguagem de marcação.
- d) engenharia estruturada.
- e) pseudolinguagem.

Comentários:

*Pessoal... se eu não souber uma linguagem de programação, eu posso escrever um algoritmo utilizando um pseudocódigo ou **pseudolinguagem**! O que é isso? É uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples sem necessidade de conhecer a sintaxe de nenhuma linguagem de programação. **Trata-se de um pseudocódigo, logo não pode ser executado em um sistema real.***

Essa é tranquilinha! Trata-se da pseudolinguagem.

Gabarito: E

5. (FCC - 2010 - DPE-SP - Agente de Defensoria - Analista de Sistemas) É utilizada para avaliar uma determinada expressão e definir se um bloco de código deve ou não ser executado. Essa é a definição da estrutura condicional:

- a) For
- b) If...Then...Else
- c) While
- d) Do...While
- e) Next

Comentários:

Pessoal... falou em estrutura condicional, trata-se de decisão! Logo, é o If-Then-Else.

Gabarito: B

6. (FCC - 2010 – TRT/SE - Analista de Sistemas) Objeto que se constitui parcialmente ou é definido em termos de si próprio. Nesse contexto, um tipo especial de procedimento (algoritmo) será utilizado, algumas vezes, para a solução de alguns problemas. Esse procedimento é denominado:

- a) Recursividade.
- b) Rotatividade.
- c) Repetição.
- d) Interligação.
- e) Condicionalidade.

Comentários:

Por fim, vamos falar sobre recursividade! O que é isso, professor? Formalmente, é uma maneira de solucionar um problema por meio da instanciação de instâncias menores do mesmo problema. Grosso modo, a recursão é uma técnica em que uma função chama a si mesma. Eu gosto de pensar em recursividade por meio de imagens como as apresentadas abaixo.

Conforme vimos em aula, trata-se da recursividade.

Gabarito: A

7. (FCC - 2009 – TJ/SE - Analista de Sistemas) A recursividade na programação de computadores envolve a definição de uma função que:

- a) apresenta outra função como resultado.
- b) aponta para um objeto.
- c) aponta para uma variável.
- d) chama uma outra função.
- e) pode chamar a si mesma.

Comentários:

Por fim, vamos falar sobre recursividade! O que é isso, professor? Formalmente, é uma maneira de solucionar um problema por meio da instanciação de instâncias menores do mesmo problema. Grosso modo, a recursão é uma técnica em que

uma função chama a si mesma. Eu gosto de pensar em recursividade por meio de imagens como as apresentadas abaixo.

Conforme vimos em aula, ela pode chamar a si mesma.

Gabarito: E

8. (CESPE - 2011 - TJ-ES - Técnico de Informática - Específicos) Uma estrutura de repetição possibilita executar um bloco de comando, repetidas vezes, até que seja encontrada uma dada condição que conclua a repetição.

Comentários:

Essa é uma definição perfeita da Estrutura de Repetição.

Gabarito: Certo

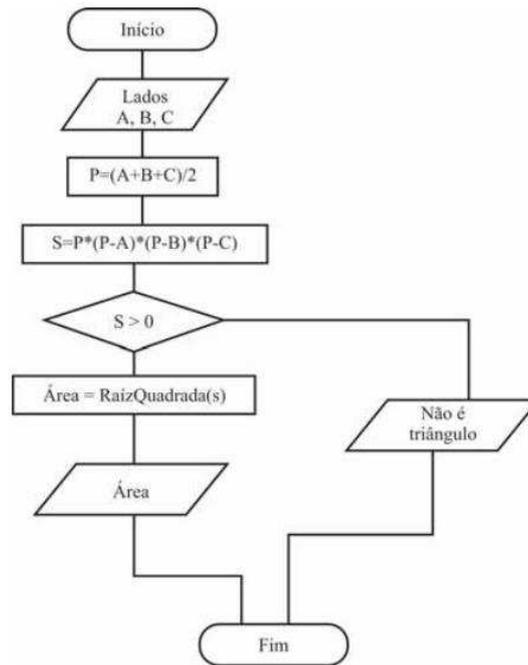
9. (CESPE - 2010 - MPU - Analista de Informática - Desenvolvimento de Sistemas) Se um trecho de algoritmo tiver de ser executado repetidamente e o número de repetições for indefinido, então é correto o uso, no início desse trecho, da estrutura de repetição Enquanto.

Comentários:

Perfeito! *Lembram-se que nós temos três estruturas de repetição?* Pois é, o Enquanto-Faça serve para esse propósito!

Gabarito: Certo

10. (CESPE - 2013 - CNJ - Programador de computador) No fluxograma abaixo, se $A = 4$, $B = 4$ e $C = 8$, o resultado que será computado para Área é igual a 32.



Comentários:

Vamos lá:

$$P = (4+4+8)/2$$

Lembrem-se que parênteses sempre têm prioridade:

$$P = (16)/2 = 8$$

Seguimos para $S = P*(P-A)*(P-B)*(P-C)$:

$$S = 8*(8-4)*(8-4)*(8-8)$$

Simplificando:

$$S = 8*4*4*0 = 0$$

$S \leq 0$, logo: Não é Triângulo!

Gabarito: Errado

11. (CESPE - 2011 - TJ-ES - Analista Judiciário - Análise de Banco de Dados - Específicos) Em uma estrutura de repetição com variável de controle, ou estrutura PARA, a verificação da condição é realizada antes da execução do corpo da sentença, o que impede a reescrita desse tipo de estrutura por meio de estrutura de repetição pós-testada.

Comentários:

Nós temos três tipos de estruturas de repetição com variável de controle: While/Enquanto, For/Para e Do-While/Faça-Enquanto - as duas primeiras pré-testadas e a última pós-testada. De fato, a estrutura For/Para é pré-testada. No

entanto, é possível reescrevê-la como uma Estrutura de Repetição Pós-Testada (Do-While/Faça-Enquanto), de forma que elas sejam equivalentes.

Gabarito: Errado

12. (CESPE - 2010 – DETRAN/ES - Analista de Sistemas) O método de recursividade deve ser utilizado para avaliar uma expressão aritmética na qual um procedimento pode chamar a si mesmo, ou seja, a recursividade consiste em um método que, para que possa ser aplicado a uma estrutura, aplica a si mesmo para as subestruturas componentes.

Comentários:

*Por fim, vamos falar sobre recursividade! O que é isso, professor? Formalmente, é uma maneira de solucionar um problema por meio da instanciação de instâncias menores do mesmo problema. Grosso modo, **a recursão é uma técnica em que uma função chama a si mesma**. Eu gosto de pensar em recursividade por meio de imagens como as apresentadas abaixo.*

Conforme vimos em aula, a questão está correta. No entanto, sendo rigoroso, ele bem que podia ter usado um verbo diferente para evitar confusão ("O método de recursividade pode ser utilizado..." ou "O método de recursividade é utilizado...").

Gabarito: Certo

13. (CESPE - 2013 – CPRM - Analista de Sistemas) Na implementação de recursividade, uma das soluções para que se evite o fenômeno de terminação do programa – que possibilita a ocorrência de um looping infinito – é definir uma função ou condição de terminação das repetições.

Comentários:

*Nós dissemos que se trata de uma função que chama a si mesma e, daí, temos o nosso primeiro problema. Se ela chama a si mesma sempre, não entra em um loop infinito? **Sim, por isso é necessário um caso-base (solução trivial), i.e., um caso mais simples em que é utilizada uma condição de parada, que resolve o problema**. Toda recursividade é composta por um caso base e pelas chamadas recursivas.*

Conforme vimos em aula, está perfeito novamente.

Gabarito: Certo

14. (CESPE - 2014 – ANATEL - Analista de Sistemas) A recursividade é uma técnica que pode ser utilizada na implementação de sistemas de lógica complexa, com a finalidade de minimizar riscos de ocorrência de defeitos no software.

Comentários:

*Por fim, vamos falar sobre recursividade! O que é isso, professor? Formalmente, é uma maneira de solucionar um problema por meio da instanciação de instâncias menores do mesmo problema. Grosso modo, **a recursão é uma técnica em que uma função chama a si mesma**. Eu gosto de pensar em recursividade por meio de imagens como as apresentadas abaixo.*

Conforme vimos em aula, definitivamente essa não é uma finalidade da recursividade. Uma implementação simples não significa de maneira alguma mais eficiente. Em geral, um algoritmo iterativo (não-recursivo) é mais eficiente que um algoritmo recursivo (por conta da manutenção de estado, pilhas, etc). Portanto, não há essa correlação entre riscos de defeito e complexidade de implementação.

Gabarito: Errado

15. (CESPE - 2011 – TJ/ES - Analista de Sistemas) Tanto a recursividade direta quanto a indireta necessitam de uma condição de saída ou de encerramento.

Comentários:

*Por fim, gostaria de mencionar que existem dois tipos de recursividade: **direta e indireta**. De modo bem simples, a recursividade direta é aquela tradicional em que uma função chama a si mesma (Ex: Função A chama a Função A); a recursividade indireta é aquela alternativa em que uma função chama outra função que chama a primeira (Ex: Função A chama a Função B, que chama a Função A).*

Conforme vimos em aula, existe recursividade direta e indireta, mas ambas precisam de um caso-base, que é uma condição de saída, para não ficar em loop infinito.

Gabarito: Certo

16. (CONSULPLAN - 2012 - TSE - Programador de computador) Observe o trecho de pseudocódigo.

```
Atribuir 13 a X;  
Repetir  
    Atribuir X - 2 a X;  
    Imprimir (X);  
Até que X < -1;
```

A estrutura será executada até que X seja igual ao seguinte valor:

- a) - 1
- b) - 3

Comentários:

Vamos lá: $X = 13$

No laço, ele afirma: $X = X - 2$

$$X = 13 - 2 = 11$$

Segue esse loop novamente:

$$X = 11 - 2 = 9$$

$$X = 9 - 2 = 7$$

$$X = 7 - 2 = 5$$

$$X = 5 - 2 = 3$$

$$X = 3 - 2 = 1$$

$$X = 1 - 2 = -1$$

$$X = -1 - 2 = -3 < -1, \text{ logo saímos do loop!}$$

Portanto, a estrutura é executada até $X = -3$.

Gabarito: B

17. (CONSULPLAN - 2012 - TSE - Programador de computador) Observe o trecho de pseudocódigo, que mostra o emprego da estrutura de controle enquanto ... faça ...

```
atribuir 0 a n;  
enquanto n < 7 faça  
    início  
        imprimir (n);  
        atribuir n+1 a n;  
    fim;
```

A opção que utiliza a estrutura para ... faça ... correspondente, que gera o mesmo resultado, é:

- a) Para n de 0 até 6 faça imprimir(n);
- b) Para n de 0 até 7 faça imprimir(n);

Comentários:

Vejamos: $N = 0$ e irá até $N < 7$, logo N de 0 a 6.

Gabarito: A

18. (FEPESE - 2010 - SEFAZ-SC - Auditor Fiscal da Receita Estadual - Parte III - Tecnologia da Informação) Assinale a alternativa correta a respeito das variáveis e constantes, utilizadas em diversas linguagens de programação.

- a) O número de constantes deve ser menor ou igual ao número de variáveis em um programa.
- b) O número de constantes deve ser menor ou igual ao número de procedimentos em um programa.
- c) O número de constantes deve ser igual ao número de variáveis em um programa.
- d) O número de constantes independe da quantidade de variáveis em um programa.
- e) O número de constantes deve ser igual ao número de procedimentos em um programa.

Comentários:

Galera, não há essa relação! O número de constantes e variáveis são independentes.

Gabarito: D

19. (NUCEPE - 2015 – SEDUC/PI - Analista de Sistemas) O código abaixo é usado para calcular o fatorial de números. Assinale a alternativa CORRETA sobre esse código:

```
função fatorial(n)
{
  se (n <= 1)
    retorne 1;
  senão
    retorne n * fatorial(n-1);
}
```

- a) Este é um exemplo de procedimento.
- b) O comando retorne pode ser retirado do código e a função terá o mesmo efeito.
- c) Exemplo clássico de recursividade.
- d) Não é possível chamar a função fatorial dentro dela mesma.
- e) O resultado da função sempre retornará um valor elevado a ele mesmo (valor ^ valor).

Comentários:

Em muitos casos, uma solução iterativa gasta menos memória e torna-se mais eficiente em termos de performance do que usar recursão. Galera, vocês se lembram que estudaram fatorial na escola? Pois é, fatorial é uma operação matemática em que um número natural (representado por $n!$) é o produto de todos os inteiros positivos menores ou iguais a n .

Conforme vimos em aula, trata-se de um exemplo clássico de recursividade.

Gabarito: C

20. (IADES - 2011 – PG/DF - Analista Jurídico - Analista de Sistemas) Os algoritmos são compostos por estruturas de controle de três tipos: sequencial, condicional e de repetição. Assinale a alternativa que apresenta apenas um tipo de estrutura de controle:

a) ...

```
escreva ("Digite seu nome: ")
leia (nome)
escreva ("Digite sua idade: ")
leia (idade)
limpe a tela
escreva ("Seu nome é:", nome)
escreva ("Sua idade é:", idade)
se (nome = "João") entao
    se (idade > 18) entao
        escreva (nome, " é maior de 18 anos!")
    fim se
fim se
...
```

b) ...

```
escreva ("Pressione qualquer tecla para começar...")
leia (tecla)
mensagem ← "Não devo acordar tarde..."
numero ← 0
enquanto (numero < 100)
    escreva (mensagem)
    numero ← (numero + 1)
fim enquanto
escreva ("Pressione qualquer tecla para
terminar...")
leia (tecla)
escreva ("Tecla digitada: ")
escreva (tecla)
...
```

c) ...

```
leia (nome)
escreva ("nome digitado: ")
escreva (nome)
se (nome = "Wally") entao
    escreva ("Encontrado o Wally!")
senao
    cont ← 5
    enquanto (cont > 0)
```

```
    escreva ("Não é Wally"...")
    cont ← (cont - 1)
  fim enquanto
fim se
...
```

```
d) ...
var
  nome: literal
  num: inteiro
inicio
  escreva ("Digite seu nome: ")
  leia (nome)
  num ← 0
  se (nome = "José") entao
    num ← (num + 1)
  fim se
  escreva ("Quantidade de João encontrados:
")
  escreva (num)
...
```

```
e) ...
var
  nome: literal
  idade: inteiro
inicio
  escreva ("Digite seu nome: ")
  leia (nome)
  escreva ("Digite sua idade: ")
  leia (idade)
  limpe a tela
  escreva ("Seu nome é:")
  escreva (nome)
  escreva ("Sua idade é:")
  escreva (idade)
fim algoritmo
...
```

Comentários:

(a) possui instruções sequenciais e também uma estrutura de decisão (uma delas aninhada), ou seja, temos dois tipos; (b) possui instruções sequenciais e também uma estrutura de repetição; (c) possui instruções sequenciais, uma estrutura de decisão e uma de repetição; (d) possui instruções sequenciais e também uma estrutura de decisão; (e) não possui estruturas de decisão (seleção), nem de repetição, somente sequenciais.

Gabarito: E

21. (IADES - 2011 – TRE-PA - Programador de Computador)

```
VAR
N1, N2 : INTEIRO;
N1 ← 2;
N2 ← 30;
INICIO
ENQUANTO N1<N2 FAÇA
    N2 ← N2 + N1;
    N1 ← N1 * 3;
FIM ENQUANTO;
N1 ← N2 + 11;
FIM
```

Dado o algoritmo escrito em pseudocódigo, quais os valores de N1 e N2, respectivamente, ao final da execução?

- a) 162 e 110.
- b) 110 e 121.
- c) 110 e 162.
- d) 121 e 110.
- e) 173 e 110.

Comentários:

A questão traz um exemplo de estrutura de repetição pré-testada e quer saber se vocês entenderam bem a aula teórica! ☺ *Vamos executar passo a passo para realizar os testes e mostrar como funciona cada iteração (repetição) do bloco da estrutura de repetição?*

1ª iteração:

N1 é igual a 2 (atribuição da 3ª linha)

N2 é igual a 30 (atribuição da 4ª linha)

Teste: $N1 < N2$?? Sim, pois 2 é menor que 30!! Então entramos no loop para executar as instruções do bloco da estrutura

N2 recebe $30 + 2$

N1 recebe $2 * 3$

2ª iteração:

N1 é igual a 6

N2 é igual a 32

Teste: $N1 < N2$?? Sim, pois 6 é menor que 32!! Então entramos no loop para executar as instruções do bloco da estrutura

N2 recebe $32 + 6$

N1 recebe $6 * 3$

3ª iteração:

N1 é igual a 18

N2 é igual a 38

Teste: $N1 < N2$?? Sim, pois 18 é menor que 38!! Então entramos no loop para executar as instruções do bloco da estrutura

N2 recebe $38 + 18$

N1 recebe $18 * 3$

4ª iteração:

N1 é igual a 54

N2 é igual a 56

Teste: $N1 < N2$?? Sim, pois 54 é menor que 56!! Então entramos no loop para executar as instruções do bloco da estrutura

N2 recebe $56 + 54$

N1 recebe $54 * 3$

5ª iteração:

N1 é igual a 162

N2 é igual a 110

Teste: $N1 < N2$? Não (finalmente :D), visto que 162 não é menor que 110!! Desse modo, saímos do loop e voltamos para a instrução imediatamente posterior, ou

seja: N1 recebe 110 + 11. Fim do Código. Ao final, N1 tem o valor de 121 e N2 de 110.

Gabarito: D

22. (CESPE – 2017 – TRE/BA – Analista Judiciário – Analista de Sistemas) Assinale a opção que apresenta a saída resultante da execução do algoritmo antecedente.

```
var a = 0, b = 1, f = 1;
  for (var i = 2; i <= 6; i++) {
    f = a + b;
    a = b;
    b = f;
    document.write(b);
  }
```

- a) 123456
- b) 12121
- c) 12345
- d) 0112
- e) 12358

Comentários:

Antes do loop, nossas variáveis valem:

a = 0; b = 1; f = 1;

Após entrar no loop, não sai enquanto $i \leq 6$. Então, temos que:

Quando $i = 2$, temos:

Primeira instrução: **f = a + b; f = 0 + b; f = 0 + 1; f = 1;**

Segunda instrução: **a = b; a = 1;**

Terceira instrução: **b = f; b = 1;**

Quarta instrução: **Saída = 1;**

Logo, temos que: **i = 2; f = 1; a = 1; b = 1;**

Quando $i = 3$, temos:

Primeira instrução: **f = a + b; f = 1 + b; f = 1 + 1; f = 2;**

Segunda instrução: **a = b; a = 1;**

Terceira instrução: **b = f; b = 2;**

Quarta instrução: **Saída = 12;**

Logo, temos que: **i = 3; f = 2; a = 1; b = 2;**

Quando $i = 4$, temos:

Primeira instrução: **f = a + b; f = 1 + b; f = 1 + 2; f = 3;**

Segunda instrução: a = b; a = 2;
Terceira instrução: b = f; b = 3;
Quarta instrução: Saída = 123;
Logo, temos que: i = 4; f = 3; a = 2; b = 3;

Quando i = 5, temos:
Primeira instrução: f = a + b; f = 2 + b; f = 2 + 3; f = 5;
Segunda instrução: a = b; a = 3;
Terceira instrução: b = f; b = 5;
Quarta instrução: Saída = 1235;
Logo, temos que: i = 5; f = 5; a = 3; b = 5;

Quando i = 6, temos:
Primeira instrução: f = a + b; f = 3 + b; f = 2 + 5; f = 8;
Segunda instrução: a = b; a = 3;
Terceira instrução: b = f; b = 8;
Quarta instrução: Saída = 12358;
Logo, temos que: i = 6; f = 8; a = 3; b = 8;

A partir daí, não entra mais no loop porque a variável de controle será maior que seis. Beleza? Então, a saída será: 12358.

Gabarito: E

23. (CONSULPLAN - 2012 - TSE – Técnico – Programação de Sistemas) Analise o pseudocódigo, que ilustra o uso de uma função recursiva.

```
programa PPRGG;  
variáveis  
  VERDE, AZUL : numérica;  
função FF(AUX:numérica):numérica;  
início  
  atribuir VERDE+1 a VERDE;  
  se AUX <= 2  
  então atribuir 5 a FF  
  senão atribuir AUX*FF(AUX-1) a FF;  
fim; { fim da função FF }  
início  
  atribuir 0 a VERDE;  
  atribuir FF(4) a AZUL;  
  escrever(VERDE,AZUL);  
fim.
```

O valor de retorno de FF e a quantidade de vezes que a função será executada serão, respectivamente,

- a) 5 e 1.
- b) 15 e 2.
- c) 60 e 3.
- d) 300 e 4.

Comentários:

Galera, como vimos, as sub-rotinas que chamam a si mesmas são chamadas de recursivas. Na questão temos a função FF, que na última linha faz uma chamada a ela mesma. Vamos fazer o passo a passo para entender como funciona a função, lembrando que a recursividade requer um critério de parada a partir de algum teste de um valor da variável usada como controle.

Segundo o início do código temos:

VERDE tem o valor 0

AZUL tem o valor retornado por FF(4)

1ª Chamada da função FF

AUX tem o valor 4 (passagem de parâmetro na chamada do programa principal)

VERDE RECEBE VERDE + 1, ou seja, VERDE RECEBE 1

AUX <= 2? Ou ainda, 4 <= 2? Não! Bloco senão é executado

FF retorna AUX * FF(AUX-1), ou seja, FF retorna 4 * FF(3)

2ª Chamada da função FF

AUX tem o valor 3 (passagem de parâmetro na chamada recursiva)

VERDE RECEBE VERDE + 1, ou seja, VERDE RECEBE 2

/ Opa, opa, professor, porque VERDE continua com o valor que recebeu na 1ª chamada de FF? Ora, meu caro padawan, porque as variáveis VERDE e AZUL foram declaradas fora da função num escopo mais geral, ou seja, as alterações dentro da função valem fora dela e para todas as suas chamadas recursivas. Agora de volta ao código :) */*

AUX <= 2? Ou ainda, 3 <= 2? Não! Bloco senão é executado

FF retorna AUX * FF(AUX-1), ou seja, FF retorna 3 * FF(2)

3ª Chamada da função FF

AUX tem o valor 2 (passagem de parâmetro na chamada recursiva)

VERDE RECEBE VERDE + 1, ou seja, VERDE RECEBE 3
AUX <= 2? Ou ainda, 3 <= 3? Sim! Bloco então é executado
FF retorna 5

/ Agora é hora de voltar nas chamadas recursivas, do último para o primeiro
Isso lembra alguma coisa?? LIFO?? Pilhas! São utilizadas nas chamadas
recursivas para voltar a execução na hierarquia das chamadas */*

Voltando à 2ª Chamada de FF

FF retorna 3 * 5, ou seja, FF retorna 15

Voltando à 1ª chamada de FF

FF retorna 4 * FF(3), ou seja, FF retorna 4 * 15, FF retorna 60. Em outras palavras, o valor de FF(4) é 60 e a função FF é chamada 3 vezes.

Gabarito: C

24. (CESPE – 2017 – TRE/BA - Analista de Sistemas)

```
var i = 0;
while (i < 5) {
  i++;
  if (i == 3) {
    continue;
  }
  document.write(i);
}
```

Assinale a opção que apresenta a saída resultante da execução do algoritmo antecedente.

- a) 12345
- b) 1245
- c) 3
- d) 0124
- e) 1234

Comentários:

Antes do loop, temos que: $i = 0$;

Após entrar no loop, não sai enquanto $i < 5$. Então, temos que:

Quando $i = 0$, temos:

Primeira instrução: $i++$ é o mesmo que $i = i + 1$, logo $i = 0 + 1 = 1$;
Segunda instrução: $i == 3$? Não, $i == 1$. Então, não entra no if;
Terceira instrução: Saída = 1;

Quando $i = 1$, temos:

Primeira instrução: $i++$ é o mesmo que $i = i + 1$, logo $i = 1 + 1 = 2$;
Segunda instrução: $i == 3$? Não, $i == 2$. Então, não entra no if;
Terceira instrução: Saída = 12;

Quando $i = 2$, temos:

Primeira instrução: $i++$ é o mesmo que $i = i + 1$, logo $i = 2 + 1 = 3$;
Segunda instrução: $i == 3$? Sim, o continue faz iterar o loop do início;

Quando $i = 3$, temos:

Primeira instrução: $i++$ é o mesmo que $i = i + 1$, logo $i = 3 + 1 = 4$;
Segunda instrução: $i == 3$? Não, $i == 4$. Então, não entra no if;
Terceira instrução: Saída = 124;

Quando $i = 4$, temos:

Primeira instrução: $i++$ é o mesmo que $i = i + 1$, logo $i = 4 + 1 = 5$;
Segunda instrução: $i == 3$? Não, $i == 5$. Então, não entra no if;
Terceira instrução: Saída = 1245;

Quando $i = 5$, temos: Não entra mais no loop.

A partir daí, não entra mais no loop porque a variável de controle não será menor que cinco. *Capiche?* Então, a saída será: 1245.

Gabarito: B

25. (VUNESP – 2014 – SP/URBANISMO – Analista Administrativo) Analise o algoritmo a seguir, apresentado na forma de uma pseudolinguagem (Português Estruturado). Esse algoritmo deverá ser utilizado para responder às questões.

```
Início
Inteiro x1, x2, x3, i;
Leia x1, x2, x3;
Para i de 1 até x1 faça
[
  x2 ← x1 + x3;
  x3 ← x1 - x2;
]
Imprima (x2 + x3);
Fim
```

Considere que os valores lidos para x_1 , x_2 e x_3 tenham sido, respectivamente, 5, 4 e 3. É correto afirmar que o valor impresso ao final da execução do algoritmo é igual a:

- a) -3
- b) 0
- c) 5
- d) 8
- e) 11

Comentários:

Sabendo que x_1 recebe 5, x_2 recebe 4 e x_3 recebe 3. Executa-se o loop do algoritmo da forma a seguir.

Quando $i = 1$, temos que:

$$x_2 \leftarrow x_1 + x_3 = 5 + 3 = 8 \quad (x_1, x_2, x_3 = 5, 8, 3)$$

$$x_3 \leftarrow x_1 - x_2 = 5 - 8 = -3 \quad (x_1, x_2, x_3 = 5, 8, -3)$$

Quando $i = 2$, temos que:

$$x_2 \leftarrow x_1 + x_3 = 5 - 3 = 2 \quad (x_1, x_2, x_3 = 5, 2, -3)$$

$$x_3 \leftarrow x_1 - x_2 = 5 - 2 = 3 \quad (x_1, x_2, x_3 = 5, 2, 3)$$

Quando $i = 3$, temos que:

$$x_2 \leftarrow x_1 + x_3 = 5 + 3 = 8 \quad (x_1, x_2, x_3 = 5, 8, 3)$$

$$x_3 \leftarrow x_1 - x_2 = 5 - 8 = -3 \quad (x_1, x_2, x_3 = 5, 8, -3)$$

Quando $i = 4$, temos que:

$$x_2 \leftarrow x_1 + x_3 = 5 - 3 = 2 \quad (x_1, x_2, x_3 = 5, 2, -3)$$

$$x_3 \leftarrow x_1 - x_2 = 5 - 2 = 3 \quad (x_1, x_2, x_3 = 5, 2, 3)$$

Quando $i = 5$, temos que:

$$x_2 \leftarrow x_1 + x_3 = 5 + 3 = 8 \quad (x_1, x_2, x_3 = 5, 8, 3)$$

$$x_3 \leftarrow x_1 - x_2 = 5 - 8 = -3 \quad (x_1, x_2, x_3 = 5, 8, -3)$$

$$\text{Logo, } x_2 + x_3 = 8 - 3 = 5.$$

Gabarito: C

26. (CESPE – 2017 – TRT 7ª Região – CE – Técnico Judiciário – Tecnologia da Informação)

```
10 A ← 5;  
11 B ← A * -2;  
12 C ← A - 1;  
13 D ← A - 2;  
14 H ← (((4*A) div D) - B) - pot(A, 2) mod C;
```

Considerando a execução do trecho de algoritmo precedente, assinale a opção que apresenta o valor atribuído a H na linha 14.

- a) 16
- b) -9
- c) 15
- d) -5

Comentários:

Para achar o valor atribuído de H, precisamos descobrir os valores de A, B, C e D. Começando da ordem das linhas da 10 até a 13, temos que:

$$A \leftarrow 5;$$

$$B \leftarrow A \times (-2) = 5 \times (-2) = -10;$$

$$C \leftarrow A - 1 = 5 - 1 = 4;$$

$$D \leftarrow A - 2 = 5 - 2 = 3;$$

Sabendo desses valores, pode-se calcular o valor atribuído a H:

$$H \leftarrow (((4 * A) \text{ div } D) - B) - \text{pot}(A, 2) \text{ mod } C, \text{ substituindo os valores;}$$

$$H \leftarrow (((4 * 5) \text{ div } 3) - (-10)) - \text{pot}(5, 2) \text{ mod } 4;$$

$$H \leftarrow ((20 \text{ div } 3) - (-10)) - \text{pot}(5, 2) \text{ mod } 4, \text{ resolvendo a multiplicação de 4 por 5, igual 20;}$$

$$H \leftarrow (6 - (-10)) - \text{pot}(5, 2) \text{ mod } 4, \text{ div é o Quociente da divisão, } 20 \text{ div } 3 = 6;$$

$$H \leftarrow (6 + 10) - 25 \text{ mod } 4, \text{ pot é a potenciação, 5 elevado ao 2, que é igual a 25;}$$

$$H \leftarrow 16 - 1, \text{ mod é o resto da divisão, } 25 \text{ mod } 4 = 1;$$

Portanto, $H \leftarrow 15$.

Gabarito: C

27. (QUADRIX – 2017 – SEDF/DF – Professor – Informática) É correto afirmar que o uso de algoritmos eficientes está relacionado ao emprego de estruturas de dados adequadas.

Comentários:

É difícil avaliar isso – trata-se de uma questão abstrata. No entanto, nós podemos fazer uma ilação e sugerir que a utilização de estruturas de dados adequadas garante uma maior eficiência no uso de algoritmos.

Gabarito: Certo

28. (IF/CE – 2017 – IF/CE – Técnico de Tecnologia da Informação) Observe a seguinte lógica de programação.

```
Inicio algoritmo
var
  N: inteiro
Inicio
  para N de 1 ate 9 faca
    se N mod 2 = 1 entao
      escreva(N)
    fimse
  fimpara
fim algoritmo
```

Este algoritmo escreve a saída:

- a) 3, 5, 7, 9
- b) 1, 3, 5, 7, 9
- c) 2, 4, 6, 8
- d) 1, 2, 4, 6, 8
- e) 1, 3, 5, 7, 8

Comentários:

Para cada número de 1 a 9, se o resto da divisão desse número por 2 for igual a 1, escreva esse número. Logo:

- 1 Mod 2: Resto 1 = 1. Logo, escreve 1;
- 2 Mod 2: Resto 0 \neq 1. Logo, não escreve 2;
- 3 Mod 2: Resto 1 = 1. Logo, escreve 3;

- 4 Mod 2: Resto $0 \neq 1$. Logo, não escreve 4;
- 5 Mod 2: Resto $1 = 1$. Logo, escreve 5;
- 6 Mod 2: Resto $0 \neq 1$. Logo, não escreve 6;
- 7 Mod 2: Resto $1 = 1$. Logo, escreve 7;
- 8 Mod 2: Resto $0 \neq 0$. Logo, não escreve 8;
- 9 Mod 2: Resto $1 = 1$. Logo, escreve 9;

Gabarito: B

29. (IF/PE – 2017 – IF/PE – Técnico de Laboratório - Informática para Internet) No que diz respeito a algoritmos, analise as proposições a seguir:

- I. Algoritmo é uma sequência de procedimentos que são executados sequencialmente com o objetivo de resolver um problema específico.
- II. O comando CASE não deve ser utilizado caso já exista no programa um comando IF.
- III. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa.
- IV. Diferentes algoritmos não podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em mais ou menos tempo, espaço ou esforço do que outros.
- V. Serve como modelo para programas, pois sua linguagem é intermediária à linguagem humana e às linguagens de programação, funcionando como uma boa ferramenta na validação da lógica de tarefas a serem automatizadas.

Estão CORRETAS as proposições:

- a) I, IV e V.
- b) II, III e IV.
- c) I, III e V.
- d) II, IV e V.
- e) I, II e III.

Comentários:

(I) Certo. Definição de algoritmo, i.e., sequência de procedimento executados sequencialmente; (II) Errado. Pode-se utilizar o comando CASE caso já exista um IF;

(III) Certo. Algoritmo representa uma sequência de passos realizados, não necessariamente sendo um programa de computador; (IV) Errado. Pode-se ter diferentes algoritmos realizando uma mesma tarefa, com diferentes instruções e diferentes tempos de execução; (V) Certo. Realiza as atividades de maneira lógica e sequencial, não somente utilizada para programas de computador.

Gabarito: C

30. (CESPE– 2017 – SEDF – DF – Professor de Educação Básica - Informática)
Considere o algoritmo a seguir:

```
Inteiro x=1, y=4, z=5;  
Enquanto (x<y) faça  
    z=z+y%x;  
    y=y-1;  
    x=x+1;  
Fim Enquanto  
Imprima (z);
```

A operação % representa o resto da divisão entre dois inteiros. Assinale a alternativa que indica o valor que será impresso:

- a) 5.
- b) 6.
- c) 7.
- d) 8.
- e) 9.

Comentários:

O algoritmo executa um loop enquanto o x for maior que y, nessa execução é adicionado ao valor de Z o resto da divisão entre x e y.

$$x = 1, y = 4, z = 5$$

$$x < y \rightarrow 1 < 4$$

$$z = z + y \% x \rightarrow 5 = 5 + 4 \% 1 = 5 + 0 = 5$$

$$y = y - 1 = 4 - 1 = 3$$

$$x = x + 1 = 1 + 1 = 2$$

$$x < y \rightarrow 2 < 3$$

$$z = z + y \% x \rightarrow 5 = 5 + 3 \% 2 = 5 + 1 = 6$$

$$y = y - 1 = 3 - 1 = 2$$

$$x = x + 1 = 2 + 1 = 3$$

$x < y \rightarrow 3 < 2$? Não! Sai do loop e escreve $z = 6$.

Gabarito: B

31. (FCC– 2017 – TRT 24ª Região – MS – Técnico Judiciário - Tecnologia da Informação) Considere o algoritmo em pseudocódigo abaixo.

```
var v1, v2, v3: inteiro
início
  leia (v1, v2, v3)
  exiba(v1)
  enquanto v3>1 faça
    v1 ← v1 * v2
    v3 ← v3 - 1
    exiba(v1)
  fim_enquanto
fim
```

Se forem lidos para as variáveis v_1 , v_2 e v_3 , respectivamente, os valores 3, 3 e 4, o último valor exibido será:

- a) 729
- b) 243
- c) 27
- d) 81
- e) 128

Comentários:

O algoritmo executa um loop que multiplica os valores das variáveis v_1 e v_2 , e coloca esse valor na variável v_1 , e faz esse passo até que a variável v_3 seja menor que 1.

$$v_1 = 3, v_2 = 3, v_3 = 4$$

$$v_3 > 1 \rightarrow 4 > 1$$

$$v_1 \leftarrow v_1 * v_2 = 3 * 3 = 9$$

$$v_3 \leftarrow v_3 - 1 = 4 - 1 = 3$$

$$v_3 > 1 \rightarrow 3 > 1$$

$$v_1 \leftarrow v_1 * v_2 = 9 * 3 = 27$$

$$v_3 \leftarrow v_3 - 1 = 3 - 1 = 2$$

$$v_3 > 1 \rightarrow 2 > 1$$

$$v_1 \leftarrow v_1 * v_2 = 27 * 3 = 81$$

$$v_3 \leftarrow v_3 - 1 = 2 - 1 = 1$$

Exibe 81.

Gabarito: D

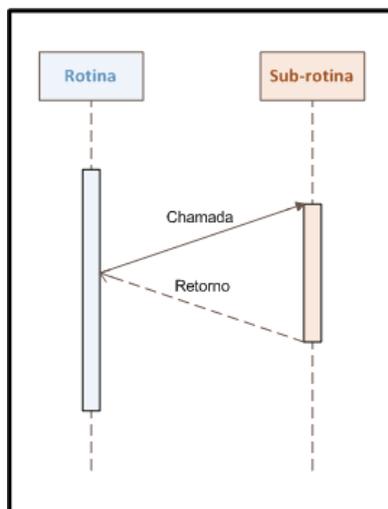
ACERTEI	ERREI



Pessoal, é importante falarmos de alguns conceitos fundamentais! *O que é uma rotina?* Uma rotina nada mais é que um conjunto de instruções – um algoritmo. **Uma sub-rotina (ou subprograma) é um pedaço menor desse conjunto de instruções que, em geral, será utilizado repetidamente em diferentes locais do sistema e que resolve um problema mais específico, parte do problema maior.**

Em vez de repetir um mesmo conjunto de instruções diversas vezes no código, esse conjunto pode ser agrupado em uma sub-rotina que é chamada em diferentes locais. **As sub-rotinas podem vir por meio de uma função ou de um procedimento.** A diferença fundamental é que, no primeiro caso, retorna-se um valor e no segundo caso, não. *Entenderam?*

Galera, essa é a ideia geral! No entanto, algumas linguagens de programação têm funções e procedimentos em que nenhum retorna valor ou ambos retornam valor, ou seja, muitas vezes, sequer há uma distinção. **No contexto da programação orientada a objetos, estas sub-rotinas são encapsuladas nos próprios objetos, passando a designar-se métodos.**



A criação de sub-rotinas foi histórica e permitiu fazer coisas fantásticas. Não fossem elas, estaríamos fazendo *goto* até hoje. **Uma sub-rotina é um trecho de código marcado com um ponto de entrada e um ponto de saída.** *Entenderam?*

Quando uma sub-rotina é chamada, **ocorre um desvio do programa para o ponto de entrada, e quando a sub-rotina atinge seu ponto de saída, o programa desaloca a sub-rotina,** e volta para o mesmo ponto de onde tinha saído. Vejam a imagem ao lado!

Algumas das vantagens na utilização de sub-rotinas durante a programação são: redução de código duplicado; possibilidade de reutilizar o mesmo código sem grandes alterações em outros programas; decomposição de problemas grandes em pequenas partes; melhorar a interpretação visual; esconder ou regular uma parte de um programa; reduzir o acoplamento; entre outros.

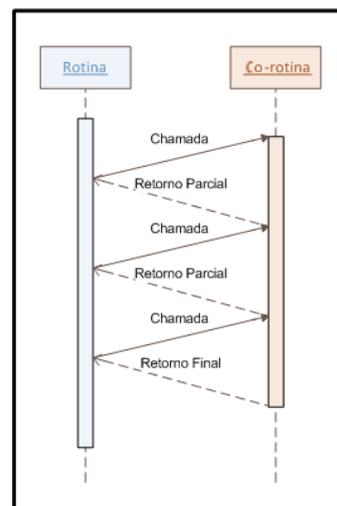
Quando uma sub-rotina termina, ela é desalocada das estruturas internas do programa, i.e., seu tempo de vida termina! É praticamente como se ela nunca tivesse existido. **Podemos chamá-la de novo, isso é claro, mas será uma nova encarnação da sub-rotina: o programa novamente começará executando desde seu ponto de entrada.** Uma co-rotina não funciona assim!

As co-rotinas são parecidas com as sub-rotinas, diferindo apenas na forma de transferência de controle, realizada na chamada e no retorno. **Elas possuem um ponto de entrada e podemos dizer que são mais genéricas que as sub-rotinas.** Na co-rotina, o trecho de código trabalha conjuntamente com o código chamador até que sua tarefa seja terminada.

O controle do programa é passado da co-rotina para o chamador e de volta para a co-rotina até que a tarefa da co-rotina termine. Numa co-rotina, existe o ponto de entrada (que é por onde se inicia o processamento), um ponto de saída (pela qual o tempo de vida da co-rotina termina), e um ponto de retorno parcial. Esse retorno parcial não termina com a vida da co-rotina.

Pelo contrário, apenas passa o controle do programa para a rotina chamadora, e marca um ponto de reentrada. **Quando a rotina chamadora devolver o controle para a co-rotina, o processamento começa a partir do último ponto de retorno parcial.**

Na prática, uma co-rotina permite retornar valores diversas vezes antes de terminar o seu tempo de vida. Enquanto o tempo de vida das sub-rotinas é ditado pela pilha de execução, o tempo de vida das co-rotinas é ditado por seu uso e necessidade.



Por fim, é bom lembrar que existem funções pré-definidas. O que é isso, professor? Galera, é um conjunto de funções já prontinhas para serem utilizadas de cada linguagem de programação. Ex: na Linguagem C, temos: `ceil(x)`, que arredonda um número real x para cima; `sqrt(x)` que calcula a raiz quadrada de x ; `strcat(x,y)`, que concatena duas strings; `strlen(x)`, que retorna a quantidade de caracteres, etc.

Infelizmente, eu não encontrei questões sobre esse tema.



Se alguém encontrar, eu posso gentilmente comentá-las.

LISTA DE EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)

LÓGICA DE PROGRAMAÇÃO

1. (CESGRANRIO – 2014 – EPE – Tecnologia da Informação) Analise o algoritmo abaixo, onde $a\%b$ representa o resto da divisão de a por b .

```
inicio
  inteiro x,y,i,r
  ler x
  ler y
  para i de 1 até x
    se (x%i=0) e (y%i=0) então
      r <- i
    fim se
  próximo
  escrever r
fim
```

Qual será a resposta, caso as entradas sejam 128, para x , e 56, para y ?

- a) 2
 - b) 8
 - c) 56
 - d) 64
 - e) 128
2. (CESGRANRIO – 2014 – PETROBRÁS - Analista de Sistemas) Analise o algoritmo abaixo em português estruturado.

```
algoritmo segredo;
variáveis
  x,y,z : inteiro;
fim-variáveis
inicio
  x:=15;
  y:=10;
  z:=0;
  enquanto y>0 faça
    z:=z+x;
    y:=y-1;
  fim-enquanto
  imprima(z);
fim
```

Que número seria impresso caso esse programa executasse?

- a) 0
- b) 10
- c) 15
- d) 100
- e) 150

3. (CESGRANRIO – 2014 – BASA - Analista de Sistemas) A saída do algoritmo apresentado abaixo para as entradas 100 e 20, respectivamente, é

```
inicio
inteiro X, Y
Ler X
Ler Y
Enquanto X  $\geq$  Y - 1 faz
X < - X - 1
Y < - Y + 2 Fim Enquanto
Escrever "saída =" , Y - X
Fim
```

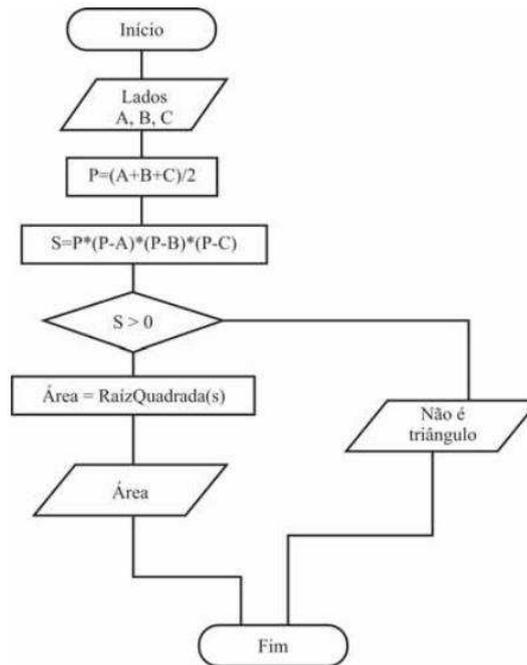
- a) -5
- b) -2
- c) 1
- d) 4
- e) 7

4. (CESGRANRIO – 2010 – PETROBRÁS – Técnico em Informático) Relacionado à programação de computadores, um algoritmo, seja qual for a sua complexidade e a linguagem de programação na qual será codificado, pode ser descrito por meio da:

- a) reografia.
- b) criptografia.
- c) linguagem de marcação.
- d) engenharia estruturada.
- e) pseudolinguagem.

5. (FCC - 2010 - DPE-SP - Agente de Defensoria - Analista de Sistemas) É utilizada para avaliar uma determinada expressão e definir se um bloco de código deve ou não ser executado. Essa é a definição da estrutura condicional:

- a) For
 - b) If...Then...Else
 - c) While
 - d) Do...While
 - e) Next
6. (FCC - 2010 – TRT/SE - Analista de Sistemas) Objeto que se constitui parcialmente ou é definido em termos de si próprio. Nesse contexto, um tipo especial de procedimento (algoritmo) será utilizado, algumas vezes, para a solução de alguns problemas. Esse procedimento é denominado:
- a) Recursividade.
 - b) Rotatividade.
 - c) Repetição.
 - d) Interligação.
 - e) Condicionalidade.
7. (FCC - 2009 – TJ/SE - Analista de Sistemas) A recursividade na programação de computadores envolve a definição de uma função que:
- a) apresenta outra função como resultado.
 - b) aponta para um objeto.
 - c) aponta para uma variável.
 - d) chama uma outra função.
 - e) pode chamar a si mesma.
8. (CESPE - 2011 - TJ-ES - Técnico de Informática - Específicos) Uma estrutura de repetição possibilita executar um bloco de comando, repetidas vezes, até que seja encontrada uma dada condição que conclua a repetição.
9. (CESPE - 2010 - MPU - Analista de Informática - Desenvolvimento de Sistemas) Se um trecho de algoritmo tiver de ser executado repetidamente e o número de repetições for indefinido, então é correto o uso, no início desse trecho, da estrutura de repetição Enquanto.
10. (CESPE - 2013 - CNJ - Programador de computador) No fluxograma abaixo, se $A = 4$, $B = 4$ e $C = 8$, o resultado que será computado para Área é igual a 32.



11. (CESPE - 2011 - TJ-ES - Analista Judiciário - Análise de Banco de Dados - Específicos) Em uma estrutura de repetição com variável de controle, ou estrutura PARA, a verificação da condição é realizada antes da execução do corpo da sentença, o que impede a reescrita desse tipo de estrutura por meio de estrutura de repetição pós-testada.
12. (CESPE - 2010 – DETRAN/ES - Analista de Sistemas) O método de recursividade deve ser utilizado para avaliar uma expressão aritmética na qual um procedimento pode chamar a si mesmo, ou seja, a recursividade consiste em um método que, para que possa ser aplicado a uma estrutura, aplica a si mesmo para as subestruturas componentes.
13. (CESPE - 2013 – CPRM - Analista de Sistemas) Na implementação de recursividade, uma das soluções para que se evite o fenômeno de terminação do programa – que possibilita a ocorrência de um looping infinito – é definir uma função ou condição de terminação das repetições.
14. (CESPE - 2014 – ANATEL - Analista de Sistemas) A recursividade é uma técnica que pode ser utilizada na implementação de sistemas de lógica complexa, com a finalidade de minimizar riscos de ocorrência de defeitos no software.
15. (CESPE - 2011 – TJ/ES - Analista de Sistemas) Tanto a recursividade direta quanto a indireta necessitam de uma condição de saída ou de encerramento.

16. (CONSULPLAN - 2012 - TSE - Programador de computador) Observe o trecho de pseudocódigo.

```
Atribuir 13 a X;  
Repetir  
    Atribuir X - 2 a X;  
    Imprimir (X);  
Até que X < -1;
```

A estrutura será executada até que X seja igual ao seguinte valor:

- a) - 1
- b) - 3

17. (CONSULPLAN - 2012 - TSE - Programador de computador) Observe o trecho de pseudocódigo, que mostra o emprego da estrutura de controle enquanto ... faça ...

```
atribuir 0 a n;  
enquanto n < 7 faça  
    início  
        imprimir (n);  
        atribuir n+1 a n;  
    fim;
```

A opção que utiliza a estrutura para ... faça ... correspondente, que gera o mesmo resultado, é:

- a) Para n de 0 até 6 faça imprimir(n);
- b) Para n de 0 até 7 faça imprimir(n);

18. (FEPESE - 2010 - SEFAZ-SC - Auditor Fiscal da Receita Estadual - Parte III - Tecnologia da Informação) Assinale a alternativa correta a respeito das variáveis e constantes, utilizadas em diversas linguagens de programação.

- a) O número de constantes deve ser menor ou igual ao número de variáveis em um programa.
- b) O número de constantes deve ser menor ou igual ao número de procedimentos em um programa.

- c) O número de constantes deve ser igual ao número de variáveis em um programa.
- d) O número de constantes independe da quantidade de variáveis em um programa.
- e) O número de constantes deve ser igual ao número de procedimentos em um programa.

19. (NUCEPE - 2015 – SEDUC/PI - Analista de Sistemas) O código abaixo é usado para calcular o fatorial de números. Assinale a alternativa CORRETA sobre esse código:

```
função fatorial(n)
{
  se (n <= 1)
    retorne 1;
  senão
    retorne n * fatorial(n-1);
}
```

- a) Este é um exemplo de procedimento.
- b) O comando retorne pode ser retirado do código e a função terá o mesmo efeito.
- c) Exemplo clássico de recursividade.
- d) Não é possível chamar a função fatorial dentro dela mesma.
- e) O resultado da função sempre retornará um valor elevado a ele mesmo (valor ^ valor).

20. (IADES - 2011 – PG/DF - Analista Jurídico - Analista de Sistemas) Os algoritmos são compostos por estruturas de controle de três tipos: sequencial, condicional e de repetição. Assinale a alternativa que apresenta apenas um tipo de estrutura de controle:

- a) ...
escreva ("Digite seu nome: ")
leia (nome)

```
escreva ("Digite sua idade: ")
leia (idade)
limpe a tela
escreva ("Seu nome é:", nome)
escreva ("Sua idade é:", idade)
se (nome = "João") entao
    se (idade > 18) entao
        escreva (nome, " é maior de 18 anos!")
    fim se
fim se
...
```

b) ...

```
escreva ("Pressione qualquer tecla para começar...")
leia (tecla)
mensagem ← "Não devo acordar tarde..."
numero ← 0
enquanto (numero < 100)
    escreva (mensagem)
    numero ← (numero + 1)
fim enquanto
escreva ("Pressione qualquer tecla para
terminar...")
leia (tecla)
escreva ("Tecla digitada: ")
escreva (tecla)
...
```

c) ...

```
leia (nome)
escreva ("nome digitado: ")
escreva (nome)
se (nome = "Wally") entao
    escreva ("Encontrado o Wally!")
senao
    cont ← 5
    enquanto (cont > 0)
        escreva ("Não é Wally"...")
        cont ← (cont - 1)
    fim enquanto
```

fim se

...

d) ...

var

nome: literal

num: inteiro

inicio

escreva ("Digite seu nome: ")

leia (nome)

num ← 0

se (nome = "José") entao

num ← (num + 1)

fim se

escreva ("Quantidade de João encontrados:

")

escreva (num)

...

e) ...

var

nome: literal

idade: inteiro

inicio

escreva ("Digite seu nome: ")

leia (nome)

escreva ("Digite sua idade: ")

leia (idade)

limpe a tela

escreva ("Seu nome é:")

escreva (nome)

escreva ("Sua idade é:")

escreva (idade)

fim algoritmo

...

21. (IADES - 2011 – TRE-PA - Programador de Computador)

```
VAR
N1, N2 : INTEIRO;
N1 ← 2;
```

```
N2 ← 30;  
INICIO  
ENQUANTO N1 < N2 FAÇA  
    N2 ← N2 + N1;  
    N1 ← N1 * 3;  
FIM ENQUANTO;  
N1 ← N2 + 11;  
FIM
```

Dado o algoritmo escrito em pseudocódigo, quais os valores de N1 e N2, respectivamente, ao final da execução?

- a) 162 e 110.
- b) 110 e 121.
- c) 110 e 162.
- d) 121 e 110.
- e) 173 e 110.

22. (CESPE – 2017 – TRE/BA – Analista Judiciário – Analista de Sistemas) Assinale a opção que apresenta a saída resultante da execução do algoritmo antecedente.

```
var a = 0, b = 1, f = 1;  
for (var i = 2; i <= 6; i++) {  
    f = a + b;  
    a = b;  
    b = f;  
    document.write(b);  
}
```

- a) 123456
- b) 12121
- c) 12345
- d) 0112
- e) 12358

23. (CONSULPLAN - 2012 - TSE – Técnico – Programação de Sistemas) Analise o pseudocódigo, que ilustra o uso de uma função recursiva.

```
programa PPRGG;  
variáveis  
  VERDE, AZUL : numérica;  
função FF(AUX:numérica):numérica;  
início  
  atribuir VERDE+1 a VERDE;  
  se AUX <= 2  
  então atribuir 5 a FF  
  senão atribuir AUX*FF(AUX-1) a FF;  
fim; { fim da função FF }  
início  
  atribuir 0 a VERDE;  
  atribuir FF(4) a AZUL;  
  escrever(VERDE,AZUL);  
fim.
```

O valor de retorno de FF e a quantidade de vezes que a função será executada serão, respectivamente,

- a) 5 e 1.
- b) 15 e 2.
- c) 60 e 3.
- d) 300 e 4.

24. (CESPE – 2017 – TRE/BA - Analista de Sistemas)

```
var i = 0;  
while (i < 5) {  
  i++;  
  if (i == 3) {  
    continue;  
  }  
  document.write(i);  
}
```

Assinale a opção que apresenta a saída resultante da execução do algoritmo antecedente.

- a) 12345
- b) 1245
- c) 3
- d) 0124
- e) 1234

25. (VUNESP – 2014 – SP/URBANISMO – Analista Administrativo) Analise o algoritmo a seguir, apresentado na forma de uma pseudolinguagem (Português Estruturado). Esse algoritmo deverá ser utilizado para responder às questões.

```
Início
  Inteiro x1, x2, x3, i;
  Leia x1, x2, x3;
  Para i de 1 até x1 faça
  [
    x2 ← x1 + x3;
    x3 ← x1 - x2;
  ]
  Imprima (x2 + x3);
Fim
```

Considere que os valores lidos para x1, x2 e x3 tenham sido, respectivamente, 5, 4 e 3. É correto afirmar que o valor impresso ao final da execução do algoritmo é igual a:

- a) -3
- b) 0
- c) 5
- d) 8
- e) 11

26. (CESPE – 2017 – TRT 7ª Região – CE – Técnico Judiciário – Tecnologia da Informação)

```
10 A ← 5;
11 B ← A * -2;
12 C ← A - 1;
13 D ← A - 2;
14 H ← ((4*A) div D) - B) - pot(A, 2) mod C;
```

Considerando a execução do trecho de algoritmo precedente, assinale a opção que apresenta o valor atribuído a H na linha 14.

- a) 16
- b) -9
- c) 15
- d) -5

27. (QUADRIX – 2017 – SEDF/DF – Professor – Informática) É correto afirmar que o uso de algoritmos eficientes está relacionado ao emprego de estruturas de dados adequadas.

28. (IF/CE – 2017 – IF/CE – Técnico de Tecnologia da Informação) Observe a seguinte lógica de programação.

```
Inicio algoritmo
var
  N: inteiro
Inicio
  para N de 1 ate 9 faca
    se N mod 2 = 1 entao
      escreva(N)
    fimse
  fimpara
fim algoritmo
```

Este algoritmo escreve a saída:

- a) 3, 5, 7, 9
- b) 1, 3, 5, 7, 9
- c) 2, 4, 6, 8
- d) 1, 2, 4, 6, 8
- e) 1, 3, 5, 7, 8

29. (IF/PE – 2017 – IF/PE – Técnico de Laboratório - Informática para Internet) No que diz respeito a algoritmos, analise as proposições a seguir:

- I. Algoritmo é uma sequência de procedimentos que são executados sequencialmente com o objetivo de resolver um problema específico.
- II. O comando CASE não deve ser utilizado caso já exista no programa um comando IF.
- III. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa.
- IV. Diferentes algoritmos não podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em mais ou menos tempo, espaço ou esforço do que outros.
- V. Serve como modelo para programas, pois sua linguagem é intermediária à linguagem humana e às linguagens de programação, funcionando como uma boa ferramenta na validação da lógica de tarefas a serem automatizadas.

Estão CORRETAS as proposições:

- a) I, IV e V.
- b) II, III e IV.
- c) I, III e V.
- d) II, IV e V.
- e) I, II e III.

30. (CESPE– 2017 – SEDF – DF – Professor de Educação Básica - Informática) Considere o algoritmo a seguir:

```
Inteiro x=1, y=4, z=5;  
Enquanto (x<y) faça  
    z=z+y%x;  
    y=y-1;  
    x=x+1;  
Fim Enquanto  
Imprima(z);
```

A operação % representa o resto da divisão entre dois inteiros. Assinale a alternativa que indica o valor que será impresso:

- a) 5.
- b) 6.
- c) 7.
- d) 8.
- e) 9.

31. (FCC– 2017 – TRT 24ª Região – MS – Técnico Judiciário - Tecnologia da Informação) Considere o algoritmo em pseudocódigo abaixo.

```
var v1, v2, v3: inteiro  
início  
    leia (v1, v2, v3)  
    exiba(v1)  
    enquanto v3>1 faça  
        v1 ← v1 * v2  
        v3 ← v3 - 1  
        exiba(v1)  
    fim_enquanto  
fim
```

Se forem lidos para as variáveis v1, v2 e v3, respectivamente, os valores 3, 3 e 4, o último valor exibido será:

- a) 729
- b) 243
- c) 27
- d) 81
- e) 128

GABARITO DOS EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
LÓGICA DE PROGRAMAÇÃO

1	2	3	4	5	6	7	8	9	10
B	E	D	E	B	A	E	Certo	Certo	Errado
11	12	13	14	15	16	17	18	19	20
Errado	Certo	Certo	Errado	Certo	B	A	D	C	E
21	22	23	24	25	26	27	28	29	30
D	E	C	B	C	C	Certo	B	C	B
31	32	33	34	35	36	37	38	39	40
D									

ESSA LEI TODO MUNDO CONHECE: PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.



1 Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.



2 Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).



3 Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).



4 Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).



5 Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).



6 Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).



7 Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.



8 O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.



Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.