

# Elaborando Algoritmos

---

## 1. Introdução

Aprender projetar e implementar algoritmos aperfeiçoará sua habilidade em pensamentos lógicos; Isto lhe dará: a) uma base para avaliar os resultados e simulações realizados por computador nos quais você, como engenheiro, confiará; uma oportunidade antecipada para dedicar-se à análise e ao projeto que é o centro do ensino e prática da engenharia; e b) uma ferramenta para melhor entender e explorar a ciência e a matemática (Holloway, 2006)

Pode-se dizer que um algoritmo é a descrição de um conjunto de comandos que, quando ativados, resultam em uma sucessão finita de acontecimentos (Campos, 2007) - uma simples lista de instruções para realizar uma tarefa (Holloway, 2006). Alguns autores acreditam que uma receita de bolo é um algoritmo porque podem ser vistas como sequência de instruções que realizam uma tarefa, mas o fato é que esse pensamento é simplista demais e deixa de lado algumas propriedades muito específicas que um algoritmo deve ter, bem como as estruturas comuns para controlar a ordem na qual as instruções serão realizadas. Por exemplo, uma das propriedades de um algoritmo é que um algoritmo tem que ser executável por um mecanismo.

Um algoritmo é uma lista de instruções que, quando executadas transformam informações da entrada até a saída. As instruções são um conjunto finito de etapas que podem ser executadas, numa ordem precisa, por um mecanismo determinista. Quando essas ações são efetivamente executadas, a execução deve terminar após um tempo finito (Holloway, 2006).

Ao analisar essa definição devemos observar 3 aspectos:

- a) Expressão finita: Um algoritmo deve possuir início e fim, caso contrário a lista de instruções seriam infinitas e nunca teríamos a chance de executar o algoritmo. O que é um contrassenso ao próximo aspecto.
- b) Execução por mecanismo: Deve existir uma máquina que possa realizar – executar – as etapas do algoritmo. Isso garante que o algoritmo será executado de forma autônoma, sem percepção, habilidade, discernimento ou sensibilidade humana, ou seja, o algoritmo não será ambíguo porque o mecanismo é determinístico.
- c) Execução finita: Se apenas um número finito de tarefas podem ser efetivamente realizadas então o algoritmo tem um tempo de execução finito.

O computador, por ser programável, é um exemplo de mecanismo determinístico capaz de executar um algoritmo. Como já vimos, a CPU do computador é capaz apenas de compreender um conjunto específico de ações que denominamos linguagem de máquina, que é pouco significativa para humanos. Para nós, construir um algoritmo usando essa linguagem seria como construir um carro começando da bauxita e do látex.

Nós também já vimos que, infelizmente, a linguagem natural que falamos não é boa para a escrita de algoritmo por ser muito ambígua, então devemos expressar o algoritmo usando uma linguagem especial, redigível pelo homem mas que possa ser traduzida em código de máquina : a linguagem de programação.


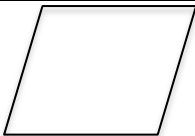
Uma linguagem de programação é uma linguagem bem definida, cujos construtores possuem uma forma precisa (sintaxe) e um significado (semântica) tais que eles podem ser traduzidos mecanicamente em linguagem de máquina. Um algoritmo expresso numa linguagem de programação ou numa linguagem de máquina é, frequentemente, chamado programa.


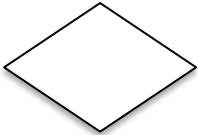

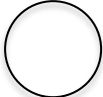
Existem algumas particularidades técnicas que precisam ser observadas ao se escrever um programa em uma linguagem de programação, logo é preferível rascunhar e organizar a lógica do algoritmo em uma notação algorítmica mais flexível antes de efetivamente implementar o mesmo. Com isso, é possível abstrair os detalhes da linguagem de programação e concentrar-se somente na matemática e no raciocínio lógico necessário a resolução do problema. Inicialmente, usaremos duas notações algorítmicas para mais adiante traduzi-las e codificá-las usando uma linguagem de programação. São elas: a) fluxograma; b) Português Estruturado (Portugol).

## 2. Fluxograma (Flow-chart)

O fluxograma é um diagrama que define graficamente a ordem de execução de tarefas, sendo cada tipo de tarefa indicada por um objeto gráfico. A vantagem do uso de fluxogramas sobre outras notações algorítmicas é sua facilidade de leitura. Como desvantagem temos uma notação pouco concisa e pouco atualizável, ou seja, se alterações precisarem ser feitas provavelmente todo o *layout* do fluxograma deverá ser alterado de forma a acomodar as mudanças tomando-se o cuidado de não perder sua principal vantagem: a simplicidade da leitura.

Fluxogramas não foram desenvolvidos exclusivamente para formalizar algoritmos. Em realidade, qualquer área de atuação que precise modelar um fluxo dinâmico usará uma variação do fluxograma. Isso faz com que haja uma falta de padronização dos símbolos usados. Nesse documento, usaremos a seguinte notação gráfica:

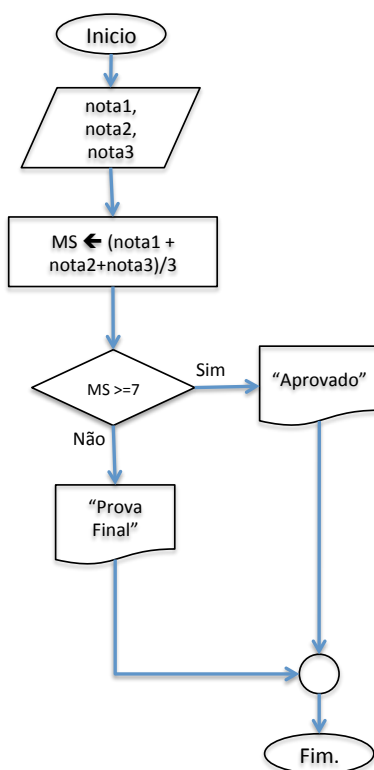
Símbolo	Significado
	Início/Fim
	Entrada de Dados, leitura (leia)

	Atribuição, Operação, Processamento
	Decisão/ Desvio Condicional
	Saída de Dados, Escrita (escrever)
	Conector

Existem algumas regras básicas que devem ser observadas na escrita de um algoritmo usando a notação de fluxogramas:

- Todo algoritmo deve possuir um INÍCIO e um FIM.
- O algoritmo deverá ser escrito utilizando-se dos blocos da simbologia básica (tabela acima)
- O fluxo de execução deve seguir um sentido único. O fluxo de leitura do algoritmo é de cima para baixo, e da esquerda para a direita. Qualquer alteração nesse fluxo deve ser indicada por seta unidirecional.
- Terminação (FIM) será o único bloco que não possuirá saída de fluxo.
- Blocos de decisão (SE) deverão possuir duas e somente duas saídas de fluxo não-simultâneas, sendo uma para VERDADEIRO e outra para FALSO. Nos outros blocos somente uma saída é permitida.
- Todas as linhas de fluxo devem descer por um lado do algoritmo, e retornar por outro, evitando que as mesmas se cruzem.
- Conectores somente devem ser utilizados em extrema necessidade.
- As variáveis utilizadas no programa devem ser nomeadas por uma única palavra, sem espaços, acentuação ou símbolos. O nome da variável deve começar necessariamente por uma letra, e pode ser seguida de letras e número.

O exemplo abaixo demonstra o uso do fluxograma para escrever um algoritmo que calcula se um aluno precisará de prova final ou se está dispensando da mesma.



Observe que a primeira tarefa a ser executada após o início do algoritmo é uma entrada de dados, nesse caso para as variáveis: nota1, nota2, nota3. Os valores dessas variáveis são irrelevantes no momento bastando apenas a certeza que esses valores serão informados. Essa é a função do símbolo de entrada de dados: garantir que as variáveis nota1, nota2 e nota3 tenham valores nominais ao deixar o símbolo de entrada de dados.

Um vez que os valores das variáveis nota1, nota2 e nota3 forem informados, então pode-se seguir para o próximo passo observando o sentido de leitura do fluxograma (de cima para baixo, da esquerda para a direita). O próximo passo é um processamento da média semestral (MS). Note que as variáveis nota1, nota2 e nota 3 são usadas nesse processamento e por isso seus valores precisaram ser previamente informados.

O próximo passo é o desvio condicional. Nele há um questionamento se o valor resultante do processamento do passo anterior (MS) é superior ou igual a 7. Dependendo da resposta, o fluxo é desviado para tarefas distintas. No nosso exemplo, se MS é maior ou igual a 7 então o próximo passo é o símbolo de saída de dados que deve dizer que o aluno está "Aprovado" - note que as aspas são importantes para deixar claro que a palavra *aprovado* não é uma variável, assim como nota1, nota2, nota3 ou MS são. Por outro lado, se MS é menor que 7 então o fluxo segue para um outro bloco de saída de dados que exibe a mensagem "Prova Final". Ambos os blocos de saída de dados tem o status de estados terminais pois estão ligados ao fim do fluxograma.

Apesar do fluxograma ser uma importante ferramenta que nos permite visualizar o algoritmo de modo quase intuitivo, ela não será a notação preferencial que usaremos em nossa disciplina; isto porque a codificação de um fluxograma em uma linguagem de programação não é tão símia como a da notação que veremos a seguir.

### 3. Pseudocódigos (PORTUGOL)

As linguagens de programação estruturadas ganharam força no início da década de 70 buscando uma melhoria na qualidade do processo e desenvolvimento de software. Exemplos consagrados dessas linguagens são o ALGOL e o PASCAL, esse última, muito utilizada no ensino de programação estruturada até hoje devido sua facilidade. Ocorre que essas linguagens, assim como quase todas as outras, possuem sintaxe em inglês e isso pode desmotivar alunos menos familiarizados com esse idioma. Para facilitar o entendimento da lógica e estruturas de programação uma tradução dessas linguagens é normalmente utilizada no ensino teórico da programação: o PORTUGOL.

Atualmente existem ambientes de programação com fins exclusivamente didáticos que permitem a codificação de programa usando PORTUGOL como linguagem. Um desses ambientes é o software VisuALG disponibilizado gratuitamente pela Apoio Informática (<http://www.apoioinformatica.inf.br/o-visualg>).

Note que o PORTUGOL encontrado na literatura não é padronizado seu objetivo é exclusivamente facilitar o entendimento de estruturas de programação e não implementar um programa diretamente no computador como faremos usando uma linguagem de programação. No entanto, para dar ao PORTUGOL o rigor sintático e semântico mínimos necessário para funcionar como uma linguagem de programação, os desenvolvedores do VisuALG precisaram ser mais rígidos na forma de escrita do mesmo. Tenha em mente que nem sempre é o caso nos livros.

### 4. Estrutura de um Programa VisuALG

No VisuALG, um algoritmo deve possuir a seguinte estrutura:

```
algoritmo "semnome"  
// Isso é um comentário  
var  
  
inicio  
// Seção de Comandos  
fimalgoritmo
```

Cada símbolo e palavra tem um propósito específico e devemos ser muito rigorosos na escrita destes comandos, caso contrário, o computador não os reconhecerá. São eles:

**Algoritmo:** É o nome do programa e deve ser escrito entre aspas. Apesar de não ser obrigatório, é recomendável que esse nome siga as regras de nomenclatura das variáveis, porém começando com letra maiúscula. Ainda, é recomendado que o nome do arquivo tenha o mesmo nome.

**Comentários:** Toda linha que é precedida de “//” não possui valor lógico, ou seja, não é imprescindível ao funcionamento do programa. Comentários são usados para deixar claro o funcionamento do algoritmo em certos trechos lógicos pouco triviais ou simplesmente pra agregar informações relevantes ao processo de desenvolvimento. O uso de comentários é estimulado mas não obrigatório.

**Var:** Marca o início da sessão de declaração de variáveis. Todas as variáveis e constantes devem ser declaradas nessa sessão antes de iniciar a lógica do algoritmo propriamente dita.

**Início:** Marca o início da lógica do algoritmo. Similar ao início do fluxograma, visto que nos fluxograma não precisamos declarar variáveis.

**Fimalgoritmo:** Marca o fim do algoritmo. Nada deve ser escrito após o fim do algoritmo. É análogo ao fim do fluxograma.

## 5. Declaração de Variáveis

Como visto anteriormente, uma variável corresponde a uma posição de memória do computador que armazena um determinado valor que pode ser usado em uma computação futura. As variáveis são representadas por identificadores, que são cadeias de caracteres alfanuméricos, e devem possuir um tipo, que no VisuALG são:

**Inteiro:** Variável numérica que armazena valores inteiros, ou seja, sem casa decimal.

**Real:** Números com casas decimais;

**Lógico:** Variável que armazena o resultado de uma expressão lógica, ou seja, verdadeiro ou falso. Escreve-se sem acentuação.

**Caracter:** Armazena texto. Lembre-se que para caracterizar valores textuais precisamos usar as aspas. Ex: “Isso é um texto”.

Observe o exemplo abaixo onde declaramos e inicializamos as variáveis:

```
algoritmo "ExemploDeclaracaoVariaveis"
```

```
var
```

```
    salarioEmpregado : real
```

```
    idadeEmpregado : inteiro
```

```
    nomeEmpregado : caracter
```

```
    possuiDependentes : logico
```

**início**

nomeEmpregado <- "Jose da Silva"

idadeEmpregado <- 40

salarioEmpregado <- 1000.00

possuiDependentes <- verdadeiro

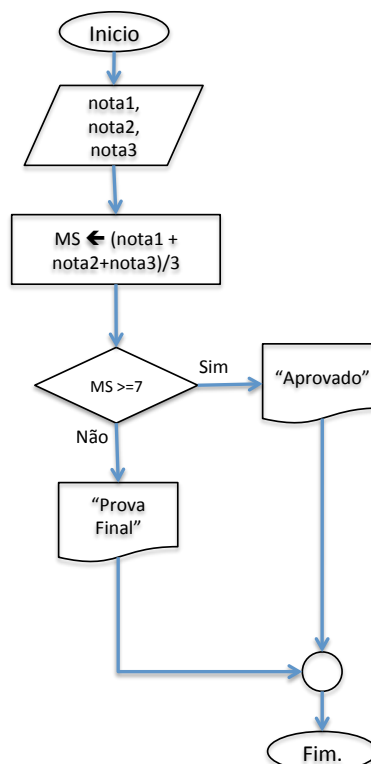
**fimalgoritmo**

Ao serem declaradas no VisuALG as variáveis são automaticamente inicializadas, isso é, lhes são atribuídas um valor padrão. Para variáveis numéricas esse valor é o *zero*, para variáveis lógicas o valor padrão é *falso*, e variáveis do tipo caracter tem como valor padrão a ausência de caracteres "".

Um vez declaradas, essas variáveis estão aptas a receberem valores. Esses valores podem ser estipulados diretamente via programação através do operador de atribuição (Ex. Idade <- 10) ou podem ser informados pelo usuário, similarmente ao símbolo de entrada de dados usado na notação de fluxograma.

## 6. Comando de Entrada

Para explicar o comando de entrada vamos voltar ao exemplo de fluxograma apresentado.



Note que nota1, nota2 e nota3 são variáveis que em um determinado momento (logo após o início) devem ter seus valores informados. Isso acontece porque estão inseridas em um bloco de Entrada de Dados. Mas que valores são esses? A resposta mais apropriada é que não importa – é irrelevante! Para algoritmo

basta dizer que, em um determinado momento esses valores devem ser conhecidos, possivelmente para efetuar um processamento (cálculo da média semestral); quaisquer que sejam os valores, o algoritmo deve funcionar e ser consistente. Lembre-se: algoritmo não é uma conta! Nós só precisamos montar a expressão computacional de forma que o computador consiga processá-la.

Quando escrevemos um algoritmo em PORTUGOL, devemos substituir o símbolo de entrada pelo comando *leia*( <nome da variável>). Observe o trecho de código abaixo em PORTUGOL que calcula a média semestral de um aluno dadas 3 notas quaisquer informadas pelo usuário.

**algoritmo** "CalculoMediaSemestral"

**var**

nota1, nota2, nota3 : real

mediaSemestral: real

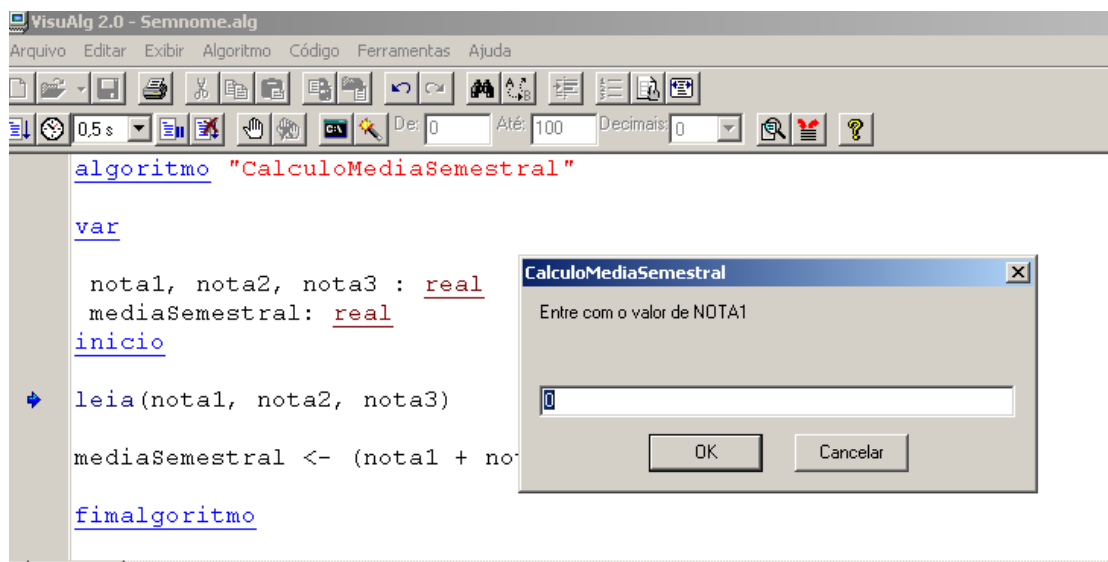
**inicio**

leia(nota1, nota2, nota3)

mediaSemestral <- (nota1 + nota2 + nota3)/3

**fimalgoritmo**

Ao executar o programa (algoritmo escrito em VisuALG) o usuário será indagado sobre que os valores que dar a nota1, nota2 e nota3 (ilustrado na figura abaixo)



Um vez informados os valores, o algoritmo prossegue e a mediaSemestral é calculada usando a fórmula  $(nota1 + nota2 + nota3)/3$ .

No visuALG o programa pode ser executado em modo DOS (padrão) ou não. O funcionamento do algoritmo é mesmo nos dois casos porém a forma de entrada dos dados e a saída difere. Nesse documento os exemplos serão em modo no NÃO – DOS a menos que seja informado o contrário.



Se olharmos para porção da janela inferior esquerda do VisuALG teremos a visão dos valores passados a essas variáveis durante a execução do programa. Nesse caso, os valores atribuídos a nota1, nota2 e nota3 foram respectivamente 10, 8, e 6. Após efetuar o processamento da média semestral a variável mediaSemestral foi usada para armazenar a resposta, nesse caso, mediaSemestral = 8.

Escopo	Nome	Tipo	Valor
GLOBAL	NOTA1	R	10
GLOBAL	NOTA2	R	8
GLOBAL	NOTA3	R	6
GLOBAL	MEDIASEMESTRAL	R	8

A janela acima referida é útil ao programador para acompanhar o que está acontecendo durante a execução do programa, porém essa forma de visualização dos dados não é adequada ao usuário final que utilizará o programa sem ter acesso ao código do mesmo. Se o objetivo for mostrar o valor de um processamento então precisa-se usar o comando de saída de dados.

## 7. Comando de Saída de Dados

No fluxograma, toda vez que precisamos exibir o resultado de um processamento, usamos o bloco conhecido como Saída de Dados. O mesmo acontece em PORTUGOL através do uso do comando *escreva* (<informação>). No exemplo anterior nós calculamos o valor da variável mediaSemestral depois que os valores da 3 notas foram informados, porém esse resultado não é exibido na tela automaticamente após o cálculo. O leitor pode ver isso com estranheza uma vez que, a priori, somente executamos um cálculo para saber o resultado do mesmo. Acontece, no entanto, que o computador sabe o resultado! Quem não sabe é você - a menos, é claro, que você mande que o resultado seja exibido no monitor. Já imaginou se o computador precisa-se mostrar o resultado de todos os bilhões de cálculos de faz por segundo?

**algoritmo** "CalculoMediaSemestral"

**var**

nota1, nota2, nota3 : real

mediaSemestral: real

**inicio**

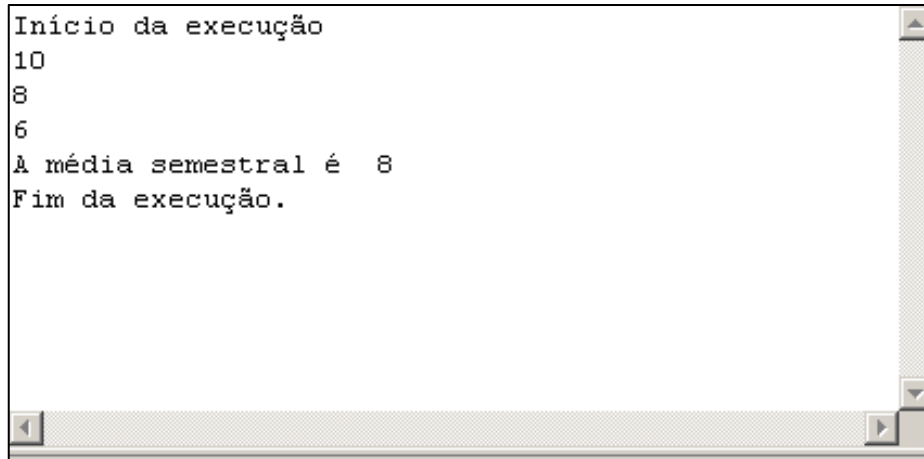
leia(nota1, nota2, nota3)

mediaSemestral <- (nota1 + nota2 + nota3)/3

escreva ("A média semestral é ", mediaSemestral)

### finalgoritmo

No exemplo acima o valor computado da média semestral é exibido na tela procedido do texto " A média semestral é ". Observe como isso é exibido pelo VisuALG na porção da janela inferior direita (modo não-DOS).



```

Início da execução
10
8
6
A média semestral é 8
Fim da execução.
  
```

O comando *escreva(<informação>)* mostra a informação na tela na posição onde o cursor está estacionado. Ele não move o cursor para próxima linha como o comando *leia(<variável>)* faz. Se desejar escrever uma informação na tela e saltar para a próxima linha então o comando *escreval(<info>)* deve ser usado.

## 8. Etapas para Elaboração do Algoritmo

Elaborar um algoritmo é uma tarefa que, essencialmente, demanda atenção aos detalhes. Por exemplo, se você é capaz de efetuar um cálculo manualmente então certamente você é capaz de dizer como procedeu passo a passo para realiza-lo. A única diferença é que você deve explicar em um nível de detalhamento e em uma linguagem capaz de ser compreendida/traduzível para linguagem de máquina.

Normalmente não existe UMA maneira correta de elaborar um algoritmo. Existe sim um conjunto de possibilidades que estão corretas – isso se observarmos simplesmente do ponto de vista lógico e funcional e desprezarmos aspectos de eficiência e performance.

Edmonds (2010) lista um série de 14 passos para se elaborar um algoritmo iterativo. Proponho simplificarmos essa lista para os cinco passos descritos abaixo. Essa simplificação deve ser vista com um guia geral para iniciarmos o pensamento algoritmo e não a única maneira de se elaborar um algoritmo. Os etapas são:

1. Identificar quais são as saídas/resultados pretendidos? O que o problema trata e qual resposta solucionaria o problema?

2. Quais informações são necessárias para se chegar a saída desejada? Existe algum valor que não foi previamente informado (entrada de dados)? Essas entradas se tornarão variáveis, logo é necessário declara-las. Qual é o tipo de dados que essas variáveis armazenarão (inteiro, real, lógico, caracter)?
3. As entradas precisam ser processadas para se tornarem as saídas. Identifique como ocorrem esses processamentos? Isso nem sempre é uma tarefa trivial – aí deve entrar sua lógica. Logo, deve-se tentar:
  - a. Achar as formulas dos cálculos que precisam ser feitos;
  - b. Identificar se os valores oriundos das formulas precisam ser armazenados e, caso positivo, declarar variáveis para esse fim;
  - c. Identificar a sequencia em que esses processamentos (cálculos) precisam ocorrer e se há situações onde eles não devem ser aplicados (desvios condicionais); Ainda, verifique se os mesmos devem se repetir e sobre quais circunstancias.
  - d. Resolva os cálculos manualmente, ou seja, resolva o problema para ganhar entendimento do mesmo; Faça-o para vários cenários (valores de entradas distintos)
4. Defina a sequencia em que as entradas de dados, processamentos, e saídas devem ocorrer. Note que o algoritmo segue um fluxo contínuo, ou seja, não podemos usar o valor resultante de um cálculo se o mesmo ainda não foi efetuado. Isso pode parecer trivial, mas é sempre um ponto de confusão entre iniciantes .
5. Teste! Faça o teste de mesa/depure o algoritmo.

Exemplificando: Suponha que relógio interno de uma máquina automatizada opera em segundos decorridos no dia. Precisamos fazer um algoritmo que converta os segundos para o formato de *hora:minutos:segundos*. Ex: 11:37:29. Como fazer isso? Vamos responder as perguntas de nosso roteiro:

1. Saída solicitada: Horas, minutos, segundos.
2. Entradas: Quantidade de segundos do dia (valor não informado). Declaremos segundos como variável do tipo inteiro.
3. Processamento: Já sabemos que em uma hora, tem-se 60 minutos e em cada minuto, 60 segundos. Logo:
  - $\text{Minutos} \leftarrow \text{segundos}/60$ ;
  - $\text{Horas} \leftarrow \text{minutos}/60$ ;

Esses cálculos no entanto não produzem valores necessariamente inteiros. Observe o exemplo: as 24 horas de um dia pode ser transformada em segundos através da multiplicação de 24 horas \* 60 minutos = 1440 minutos \* 60 segundos = 86400 segundos. Seguindo raciocínio inverso, quantos minutos tem-se em 41849 segundos? A

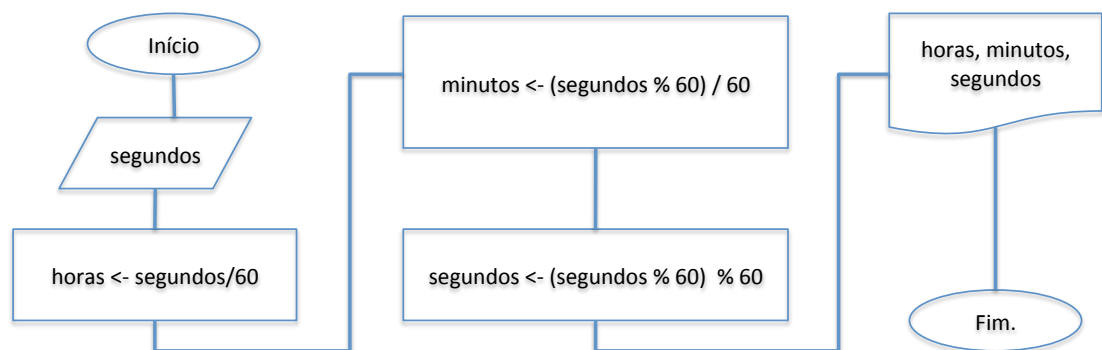
resposta é dada por  $41849/60 = 697.48$ , ou seja, 697 minutos + 0.48 minutos. Seguindo a lógica,  $0.48 * 60 = 29$  segundos. Ao dividirmos 697 por 60 achamos a quantidade de horas que é de 11.61, ou seja, 11 hora e  $0.61 * 60 = 37$  minutos.

Uma outra linha de raciocínio é calcular o resto da divisão (%) entre os inteiros 41849 e 3600 = 2249. Isso representa os segundos excedentes para que a conta seja “redonda”. Sendo assim,  $(41849 - 2249) / 3600 = 11$  horas exatamente! Aplicando o mesmo raciocínio aos minutos, temos o  $2249 \% 60 = 29$  segundos, então  $(2249 - 29)/60 = 37$  minutos exatos. Logo,  $41849 = 11:37:29$

#### 4. Definindo a sequencia:

- a. Ler a quantidade de total segundos;
- b. Dividir a quantidade total de segundos por 60 para achar os minutos, e depois novamente por 60 para achar as horas. Alternativamente, divide-se por 3600. Essa divisão deve ser entre números inteiros para resultar em um número inteiro, ou seja,  $horas \leftarrow segundos/3600$ .
- c. A quantidade de minutos é dada pelo resto da divisão dos segundos por 3600, porém esse valor resultante estará em segundos, ou seja, precisa ser dividido novamente por 60. Logo,  $minutos \leftarrow (segundos \% 3600)/60$ .
- d. A quantidade de segundos é dada pelo resto da divisão dos minutos (em segundos) por 60, ou seja,  $segundos \leftarrow (segundos \% 3600) \% 60$
- e. Exibir horas, minutos, segundos.

Em forma de fluxograma, podemos escrever sequencia como:



Ao traduzir isso para o Portugol executável do VisuALG precisamos tomar alguns cuidados técnicos como, por exemplo, indicar que o resultado de um cálculo deve ser inteiro (função int). Observe como fica:

**algoritmo** "TransformaSegundosHoras"

**var**

segundos : inteiro

horas : inteiro

minutos: inteiro

**inicio**

leia (segundos)

horas <- int(segundos / 3600)

minutos <- int((segundos % 3600) / 60)

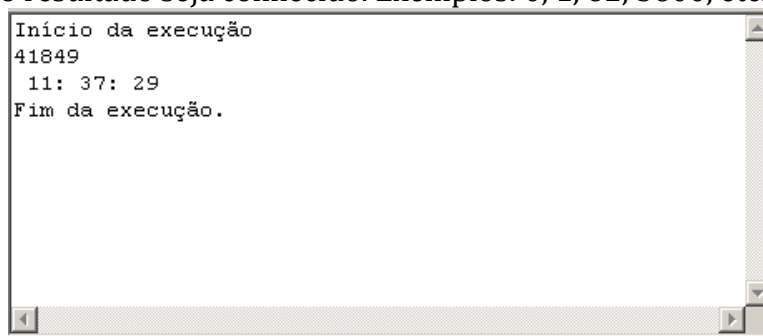
segundos <- int((segundos % 3600) % 60)

escreva (horas, ":", minutos, ":", segundos)

**fimalgoritmo**

5. Agora temos que testar o programa ou fazer o teste de mesa no caso de o algoritmo não ter sido escrito no VisuALG. Para verificar se o algoritmo funciona, deveríamos testar todos os cenários possíveis mas isso é contraproducente considerando que temos 86400 cenários possíveis (quantidade de segundos em 24 horas) Logo, escolheremos uma base de teste e verificaremos sobre ela.

Vamos começar com o valor usado no exemplo do passo anterior: 41849. Conforme podemos ver na figura abaixo, o valor calculado foi o mesmo que era esperado. Agora temos que testar com outros valores aleatórios cujo o resultado seja conhecido. Exemplos: 0, 1, 62, 3600, etc.



Uma boa base de testes deve possuir valores que testem os limites do algoritmo (os extremos da faixa de valores). Por exemplo, se entrarmos com valores maiores que 86400 ou menores que 0, será que o algoritmo vai funcionar? Em princípio esses valores não deveriam ser entrados mas erros ocorrem e seu programa tem que estar preparado para suportá-los. Vamos voltar a esse assunto mais a frente.

## 9. Referencias

CAMPOS, F. **Algoritmos Numéricos** - 2ª Ed. Rio de Janeiro: LTC, 2007.

EDMONDS, Jeff. **Como pensar em Algoritmos**. Rio de Janeiro: LTC, 2010.

HOLLOWAY, James Paul. **Introdução à Programação para Engenharia: Resolvendo Problemas com Algoritmos**. Rio de Janeiro: LTC, 2006