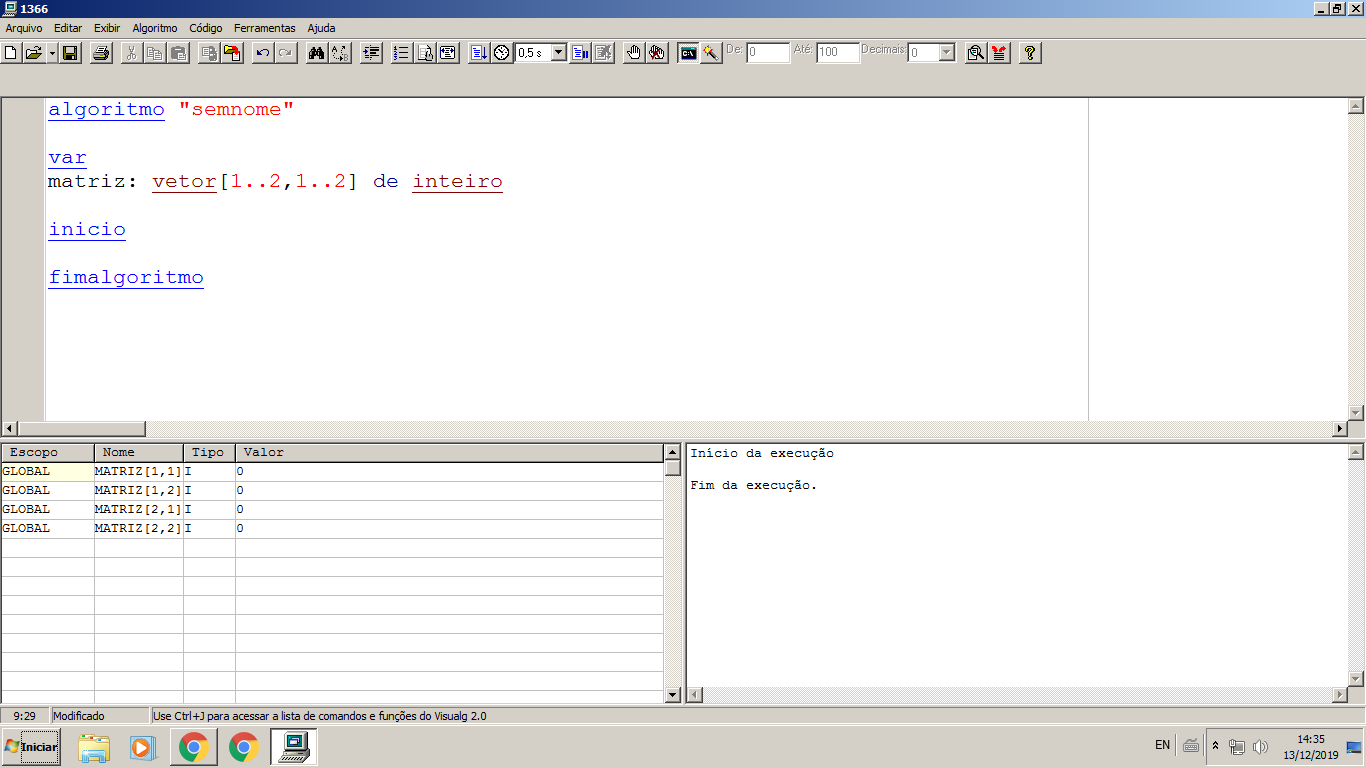
**Introdução**

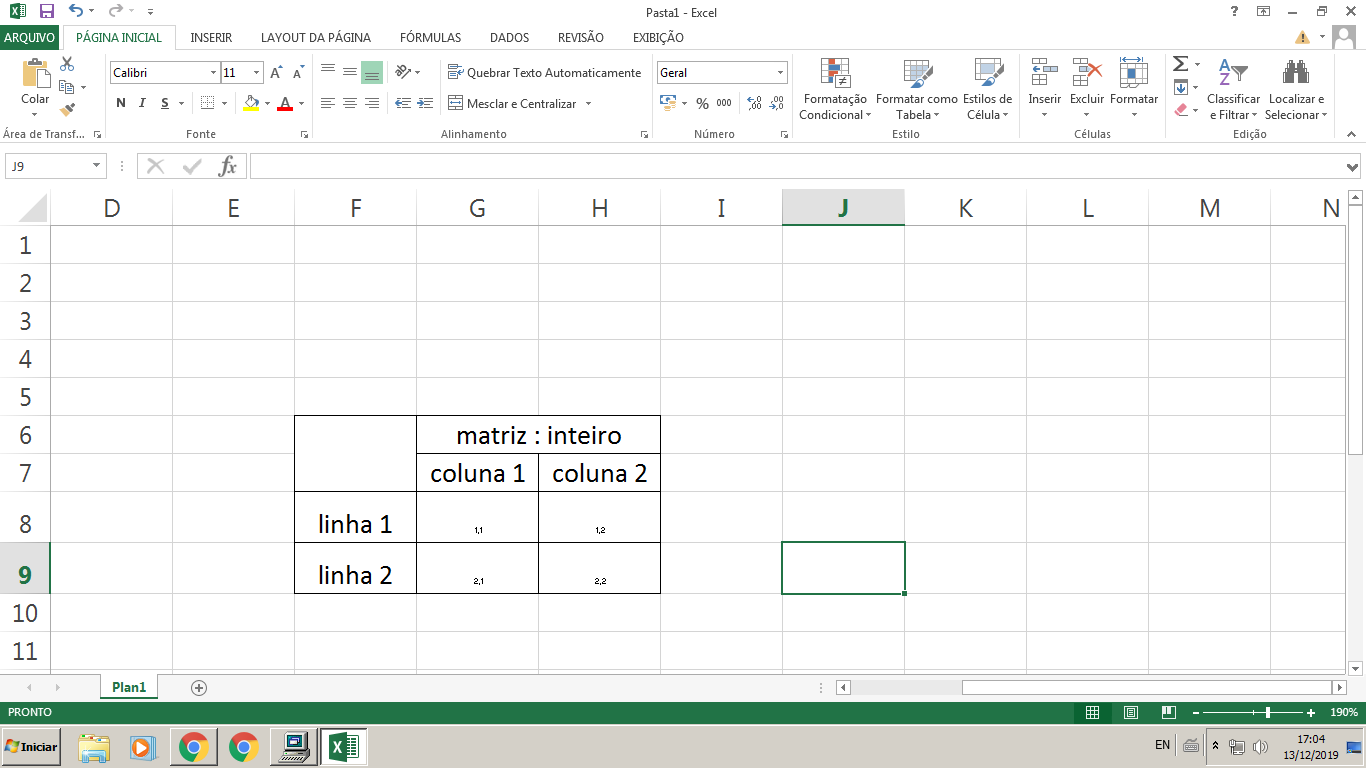
Este documento abordará uma estrutura semelhante ao vetor, que você conheceu na lista anterior, no entanto, trata-se de um **vetor bidimensional** ou **matriz** como vamos chamar. A matriz é uma estrutura composta não somente por **colunas**, mas também por **linhas**.

**Declaração**

No VisuAlg, declaramos uma matriz indicando o número de **linhas** e **colunas** que desejamos que ela contenha:

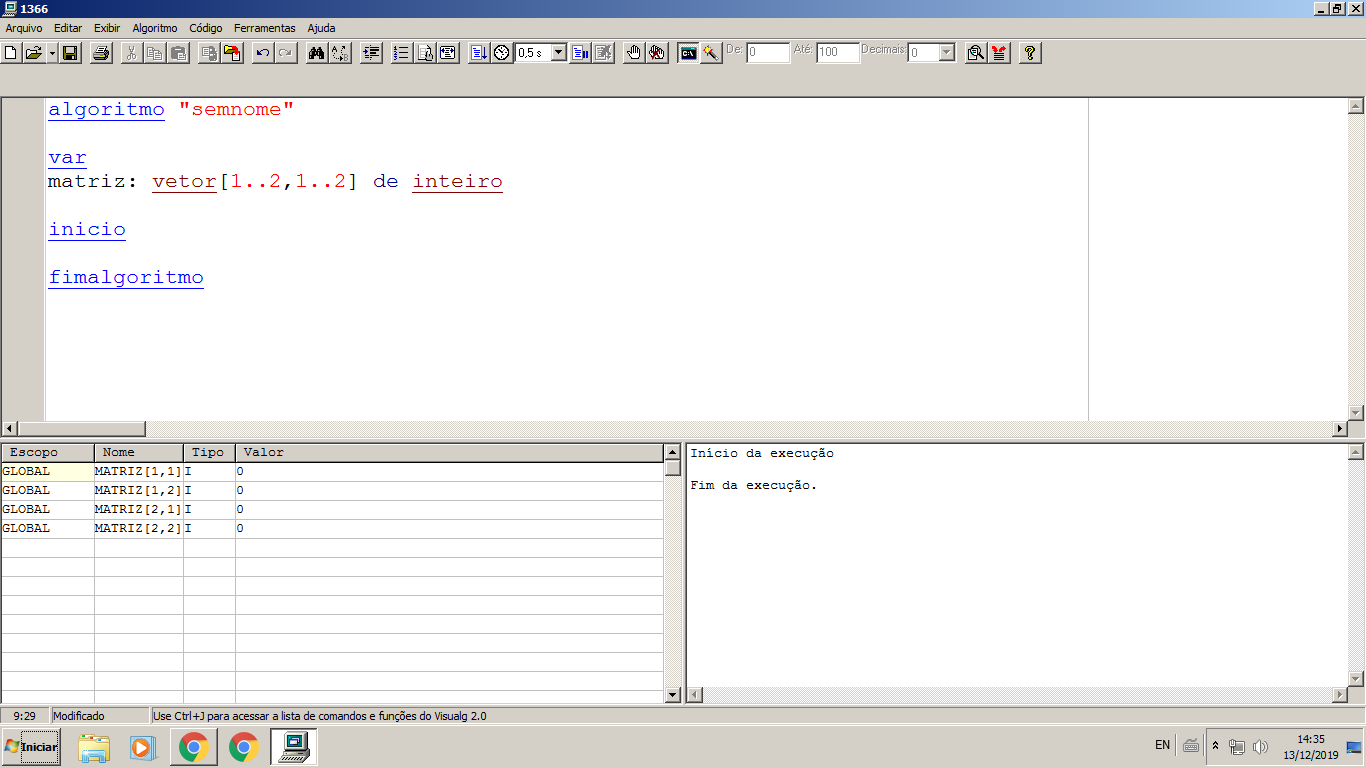


Perceba que, da mesma forma como você fez para os vetores unidimensionais (e para qualquer variável declarada no VisuAlg) é necessário que seja informado o tipo de dado a ser armazenado na matriz. No exemplo, a matriz é composta por 2 linhas e 2 colunas. Representando graficamente a matriz declarada, seria algo como:

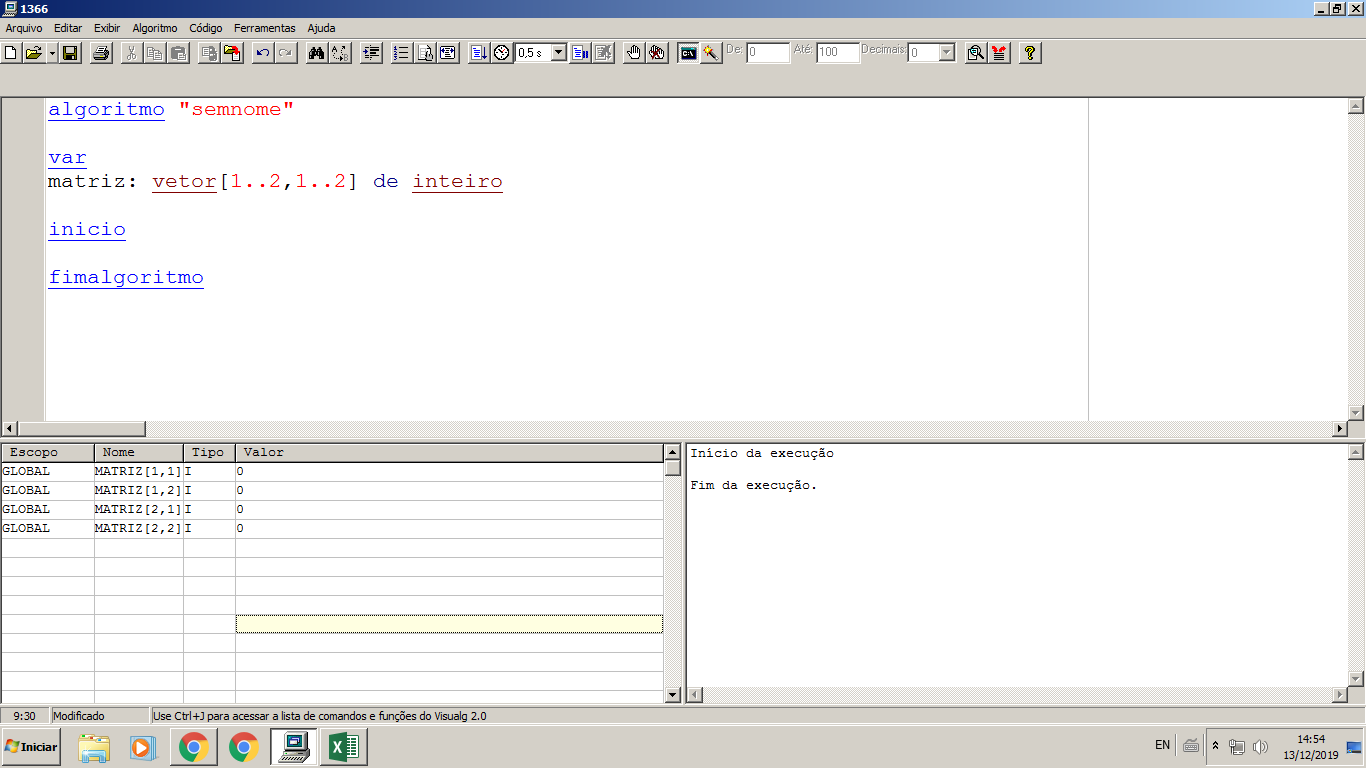


Outro ponto interessante é que a nossa matriz recebeu o nome de **matriz**, pois essa não é uma palavra reservada no VisuAlg. A matriz para o VisuAlg é um vetor, porém, um **vetor bidimensional,** como já mencionamos. A grande diferença é que ao declarar um vetor bidimensional, é necessário que sejam informadas quantas LINHAS e COLUNAS compõem a matriz. Da mesma forma como nos vetores, as chaves de uma matriz podem ser apenas numéricas e são representadas sempre por um par de números, ex.: matriz**[1,1]**, sendo um valor para cada dimensão (primeiro a linha, depois a coluna).

Bom, continuando com o exemplo dado, onde declaramos uma matriz **2x2**, iniciando as **linhas** e **colunas** com 1...

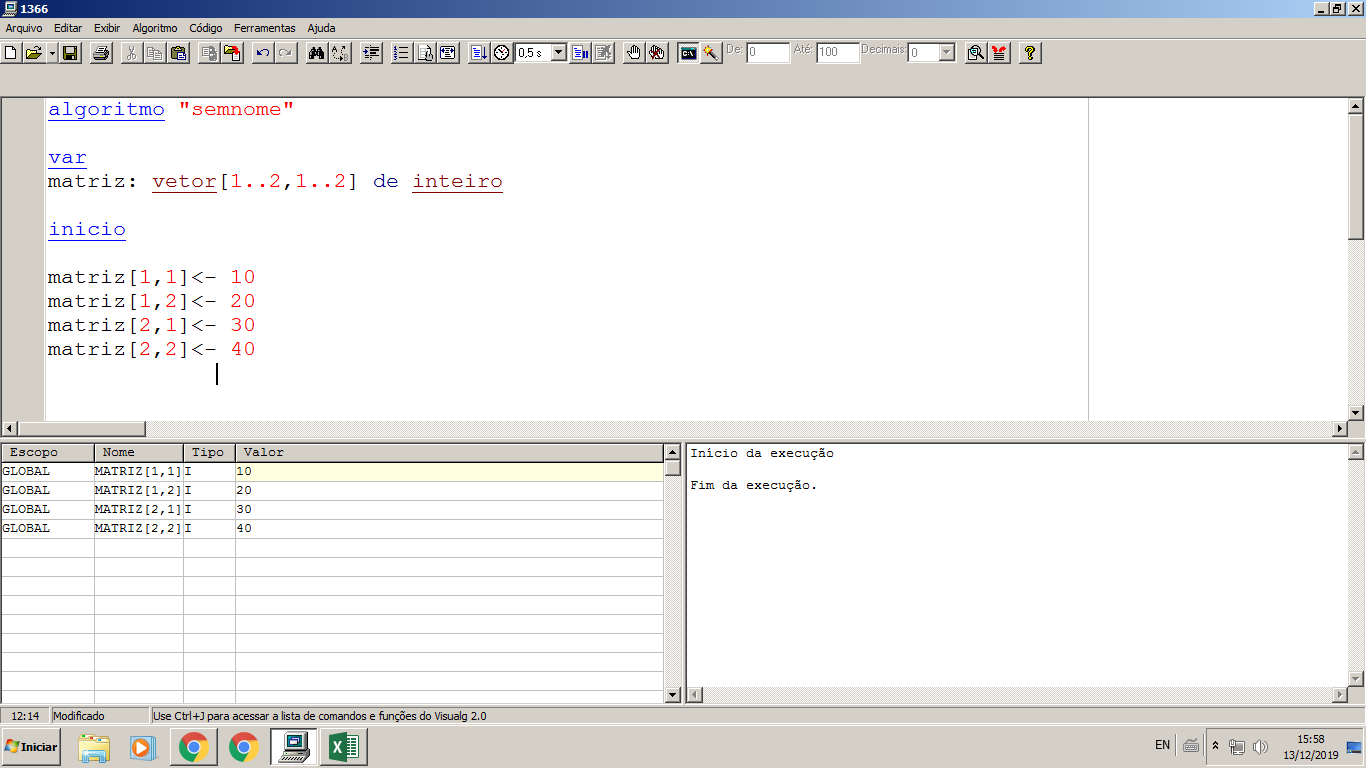


...veja a imagem abaixo, que representa as posições que o programa disponibilizaria na memória:

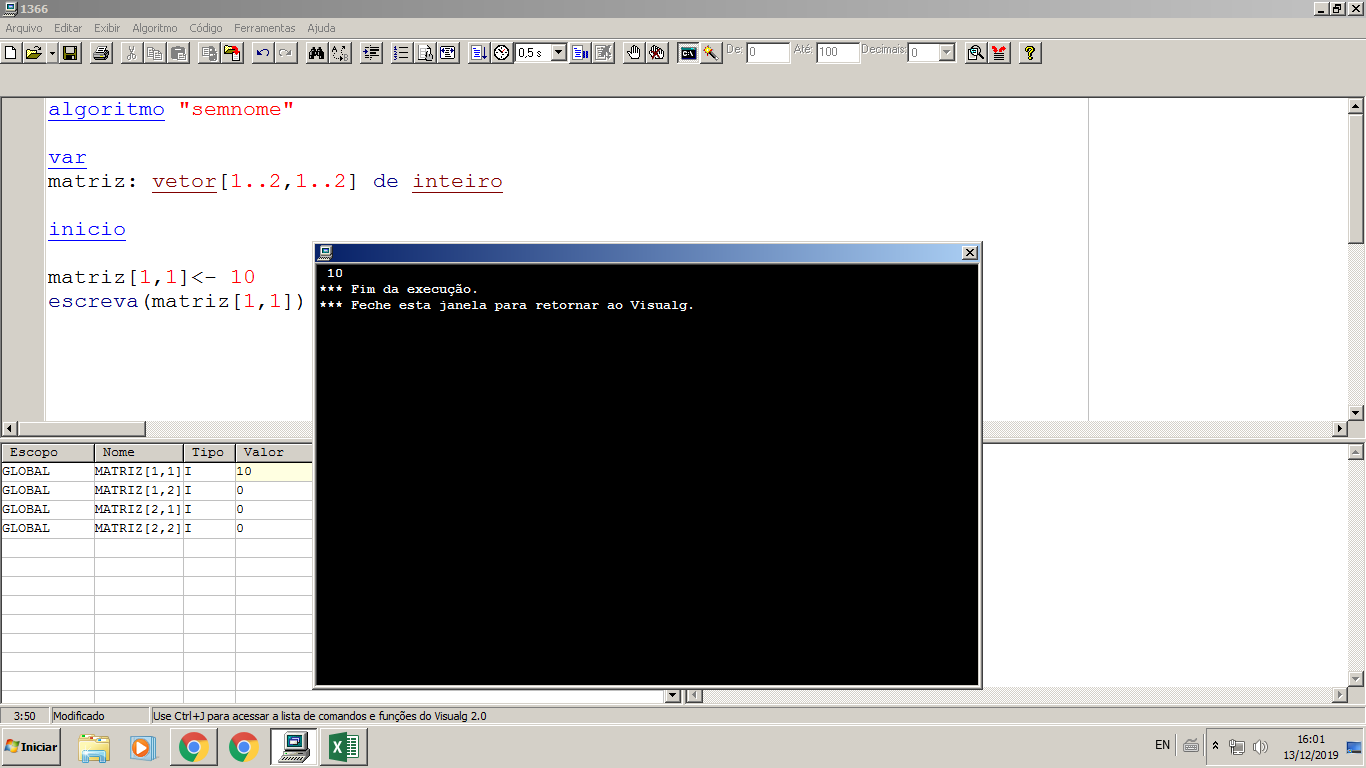


Explicando a imagem acima, **matriz** é o *nome* da variável e os valores em colchetes são as **chaves**, de **1** a **2**, pois foram esses os valores informados dentro dos colchetes na declaração. A primeira posição da matriz é composta pela chave **[1,1]**, a segunda, pela chave **[1,2]**, a terceira pela chave **[2,1]** e a quarta pela chave **[2,2]**.

Repare que com essa matriz poderíamos salvar na memória 4 números inteiros. Para fazer isso, precisamos indicar a chave na qual desejamos guardar a informação. Para preencher essa matriz *estaticamente*, poderíamos fazer assim:

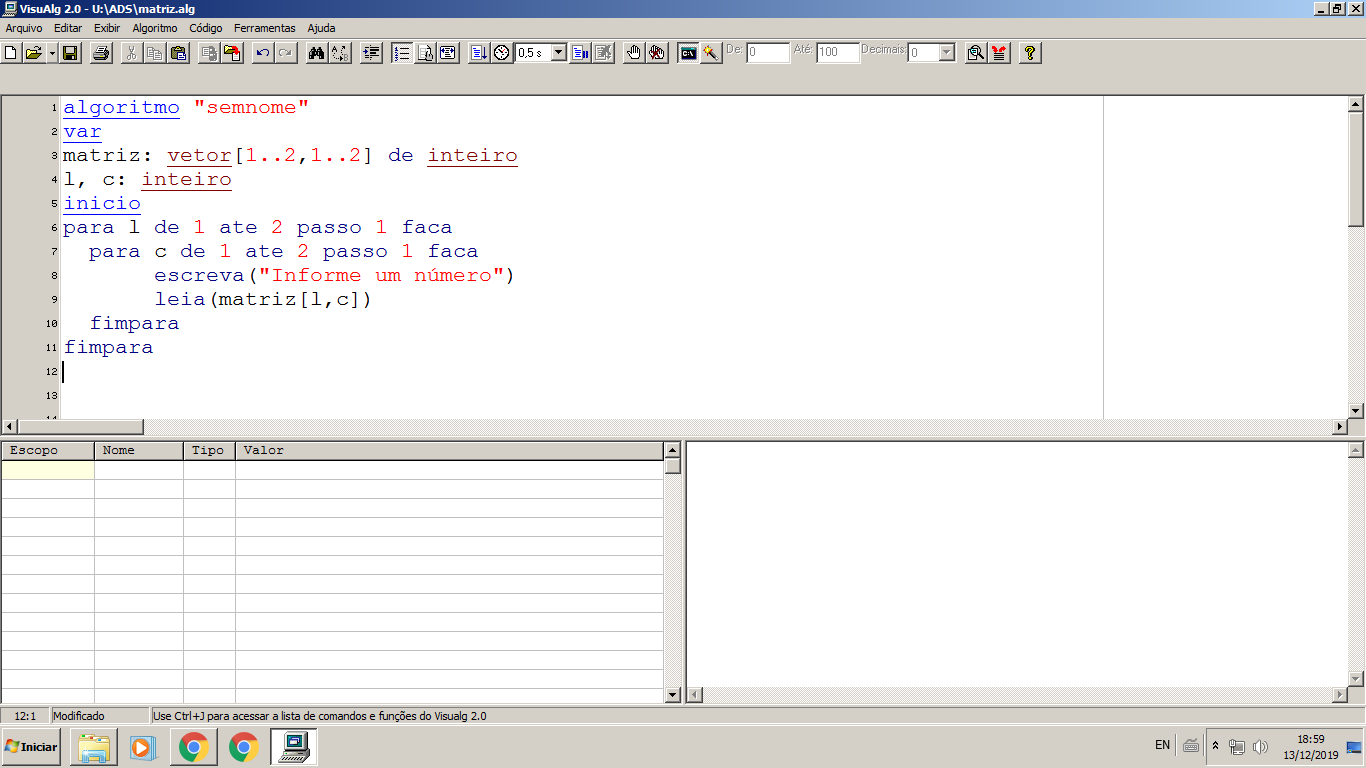


Para mostrar qualquer um desses valores, também é preciso informar a chave que contém a informação que queremos. Veja abaixo:



No nosso exemplo, temos uma matriz relativamente pequena, de somente 4 posições. Mas, assim como quando você trabalhou com vetores, os tamanhos dessas estruturas podem ser maiores, a ponto de não ser viável manipular os valores das posições escrevendo-as estaticamente. Sendo assim, podemos e devemos utilizar estruturas de repetição para percorrer uma matriz, seja para armazenar valores nela ou mostrá-la.

Para o nosso exemplo, precisaríamos de **duas estruturas de repetição** aninhadas, afinal, **temos que percorrer linhas e colunas** no caso de uma matriz. A estrutura de repetição adequada para essa situação é o PARA-FACA, pois temos controle sobre a quantidade de repetições necessária. Veja o exemplo a seguir:

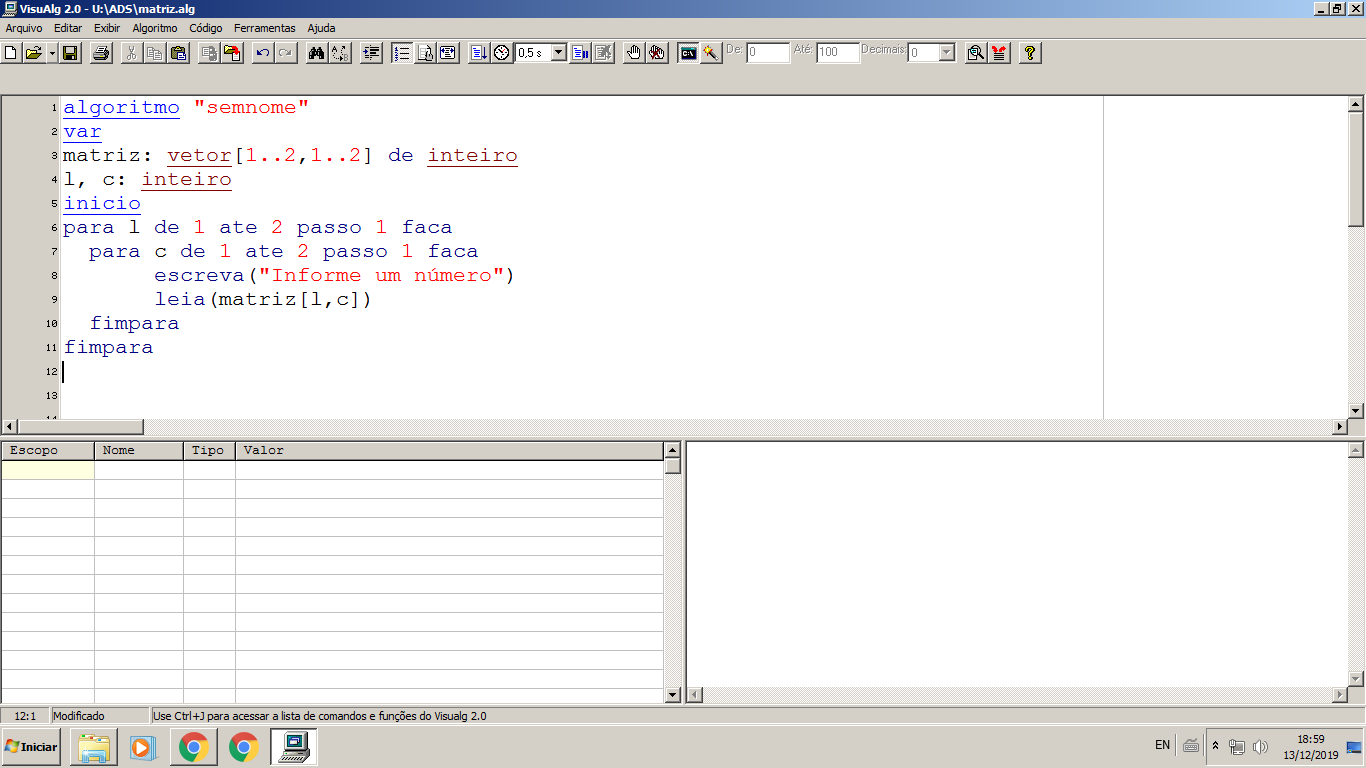


Observe que da mesma forma como no vetor você precisava designar a variável contadora para percorrer as **colunas/posições,** para a matriz, você irá designar, no nosso exemplo, o **“L”** para percorrer as **linhas** e o **“C”** para percorrer as **colunas**. Obviamente, essas variáveis precisam ser declaradas na seção **var** e devem ser do tipo **inteiro**.

Observe que na repetição, indicamos que a variável “L” deve começar em 1 e ir até 2, bem como a variável “C”, pois temos exatamente 02 linhas e 02 colunas. A matriz nesse caso é quadrada (mesma quantidade de linhas e colunas) mas não precisa e não irá ser sempre assim, podemos ter matrizes de diferentes ordens (ou seja, diferentes quantias de linhas e colunas).

Os nomes das variáveis são bastante óbvios e você deve estar se perguntando *“mas até então, não era uma má prática declarar variáveis com nomes ruins?”*... sim! Continua sendo, mas nesse caso, essas variáveis são somente para percorrer a matriz, e por lógica pode ser utilizado **“L”** e **“C”**. Se você fizer algumas pesquisas sobre matriz, verá que é largamente utilizado **“I”** para percorrer as linhas e **“J”** para percorrer as colunas, isso por conta de convenções da matemática. Enfim, em resumo, para esse caso específico não seria uma má prática usar variáveis com somente uma letra como nome, até para que fique mais “limpa” a linha de código (*mas não pegue gosto por isso… rs!*).

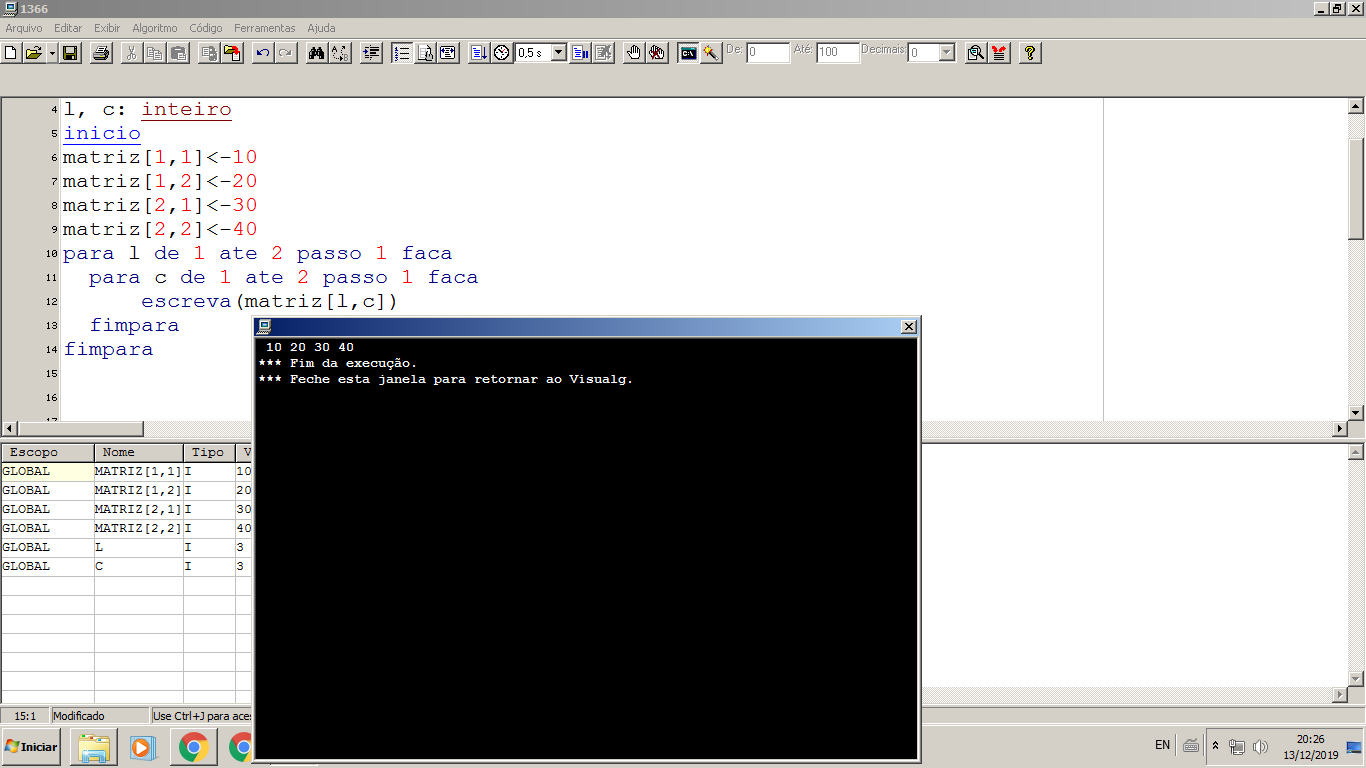
Você deve se atentar de que todas as colunas da primeira linha da matriz são percorridas antes de se passar para a segunda linha e assim sucessivamente. Com um rápido teste de mesa você conseguiria identificar isso. No nosso exemplo, como temos 02 estruturas de repetição aninhadas, onde por sua vez, cada laço de repetição acontece 02 vezes, ao final da linha 11, 04 repetições acontecerão, certo? Reveja abaixo...



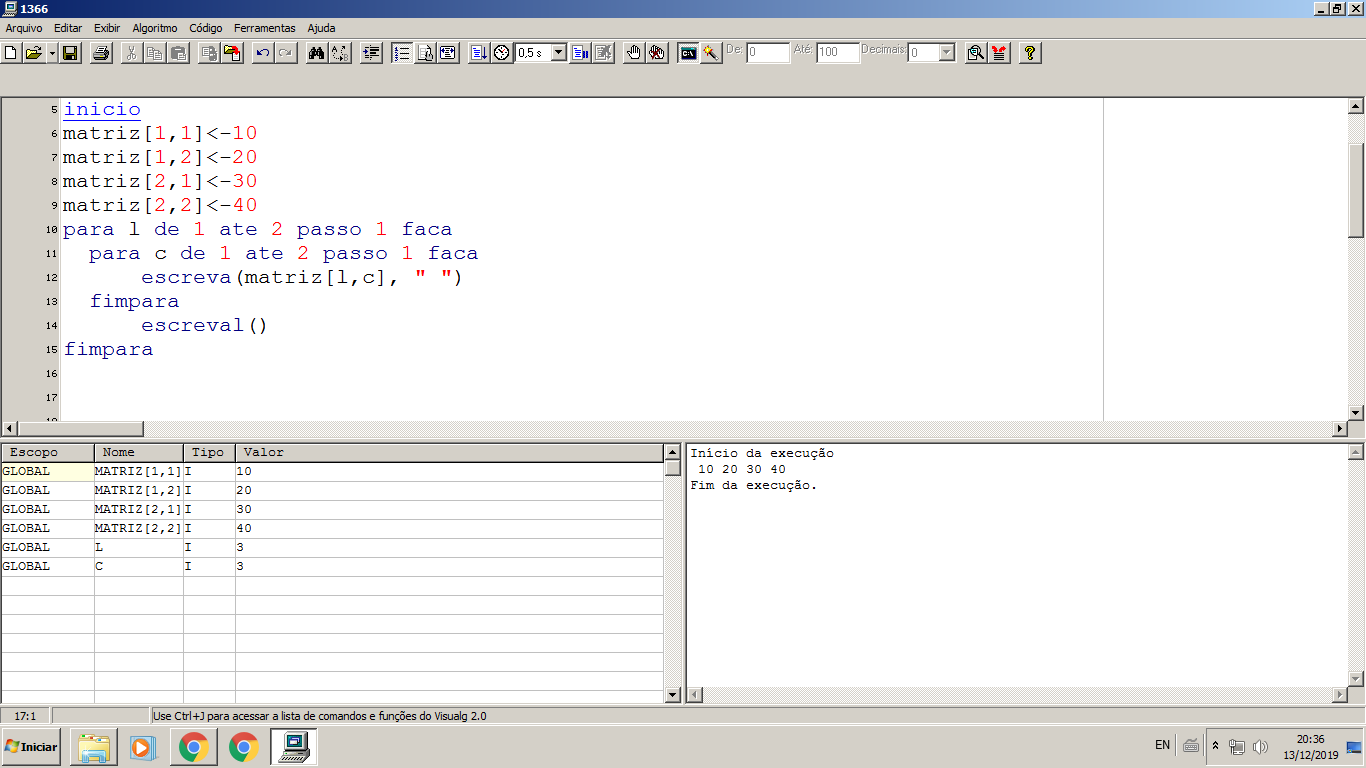
Detalhando um pouco mais, cada vez que o programa passar pela linha 09, a cada repetição, os valores de “L” e “C” serão:

| **Repetição** | **Valor de L** | **Valor de C** | **Posição acessada** |
| --- | --- | --- | --- |
| 1ª | 1 | 1 | matriz[1,1] |
| 2ª | 1 | 2 | matriz[1,2] |
| 3ª | 2 | 1 | matriz[2,1] |
| 4ª | 2 | 2 | matriz[2,2] |

Ou seja, como já mencionado, todas as colunas da primeira linha são acessadas antes da segunda linha. Se desejássemos mostrar a matriz 2x2 que criamos, considerando os valores 10, 20, 30 e 40 colocados estaticamente, bastaria:



Observe que, como utilizamos o **escreva()**, os valores foram exibidos lado a lado. Não é porque estamos trabalhando com uma matriz que o comando escreva() irá se comportar de forma diferente. Sendo assim, para que a matriz seja exibida no formato de uma “tabela”, pode ser feito:



Observe que na linha 12 mantivemos o escreva(), porém, após o valor da matriz, adicionamos um espaço e na linha 14 inserimos um escreval() para pular linha assim que a segunda repetição encerrar, o que significa que uma mudança de linha acontecerá assim que todas as colunas daquela linha forem percorridas.

**Quando utilizar uma matriz?**

A vantagem de se usar uma matriz é semelhante à vantagem de se utilizar um vetor: quando você precisa armazenar vários valores do mesmo tipo e, de forma organizada. No entanto, no caso da matriz, você deve considerar que existe uma dimensão a mais e avaliar se é importante para a resolução do seu problema considerar essa dimensão. Em resumo, se for necessária a utilização de vários vetores de mesmo tamanho para resolver um problema, pode ser que uma matriz lhe seja útil, considerando que todos os vetores armazenariam o mesmo tipo de dado. Na prática, para as soluções “comerciais” que você irá desenvolver aqui no curso, o conceito e a utilização de vetores unidimensionais já nos atendem.

**Alerta de *spoiler*:** você poderá usar matrizes para modelar o tabuleiro do desafio final, seja para o jogo da velha ou o batalha naval.

**Prática**

Declare uma matriz de tamanho 2x3 que deverá receber nomes de pessoas que devem ser digitados pelo usuário. Utilize estruturas de repetição aninhadas para percorrer as posições da matriz, como fizemos aqui. Atente-se à quantidade de linhas e colunas necessárias e tome cuidado para não definir uma estrutura de repetição que ultrapasse as posições existentes! Ao final, mostre a matriz utilizando a dica do coração que fornecemos, para exibi-la em formato de matriz (tabela).

**P.S.:** Aproveite e, ao fazer este exemplo conforme solicitamos, repare em outro detalhe: se os tamanhos dos nomes forem diferentes, a matriz vai ter uma exibição ruim… Não se pode ter tudo ~~na vida~~ no VisuAlg, né, gente...