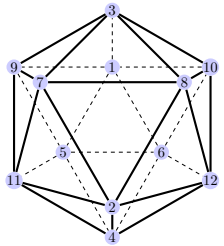
Metaheuristicas - Lab1 Estruturas de dados

Alexandre Checoli Choueiri

11/03/2023



Introdução

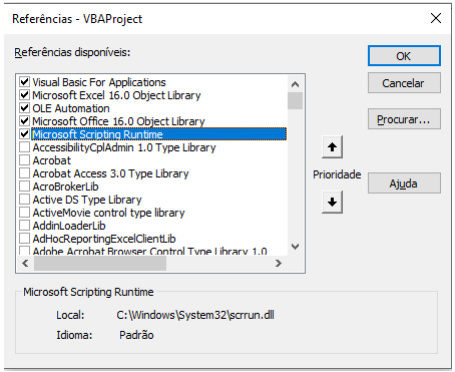
A importância

1. As estruturas que usamos para armazenar dados em um computador são chamadas estruturas de dados.
2. As diferentes estruturas possuem diferentes características em relação às operações básicas que executamos (INSERIR, REMOVER, BUSCAR).
3. Precisamos conhecer minimamente como estas estruturas se relacionam com a memória do computador, para conseguirmos um maior desempenho computacional.

Introdução

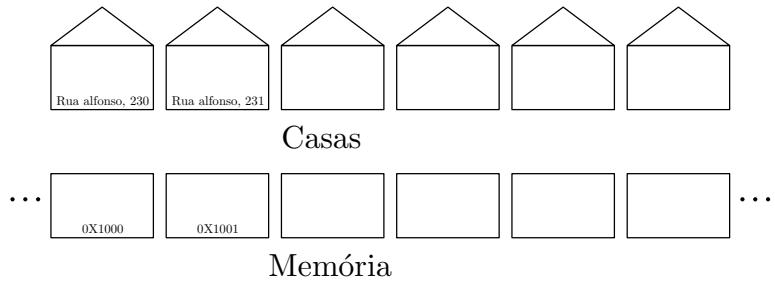
VBA

Tudo o que é mostrado nessa aula (na parte de VBA) está contido em detalhes em uma apostila de VBA, no seguinte LINK. Para usar as estruturas de dados desta aula, adicione o *Microsoft Scripting Runtime*: no VBA, abra ferramentas → referências, selecione a opção *Microsoft Scripting Runtime*, como mostrado na Figura abaixo.



Introdução

A memória do computador

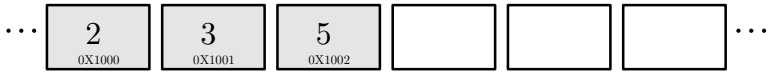


A disposição da memória em um computador pode ser pensada como um conjunto de casas, uma ao lado da outra. Toda casa possui um endereço, bem como cada "pedaço" da memória em um computador.

Estruturas de dados

Vetores

$$v = [2,3,5]$$



Memória

Se pensarmos em um vetor, cada elemento dele é salvo em um endereço de memória. A principal característica dos vetores é que os elementos são armazenados **de forma contígua**.

Estruturas de dados

Vetores

$$v = [2,3,5]$$

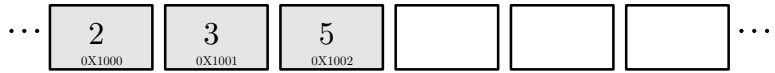
Memória

Ainda, o tamanho do vetor é constante. Após a definição de sua dimensão, só poderemos inserir mais elementos se o **redimensionarmos**.

Estruturas de dados

Vetores

$$v = 0X1000$$



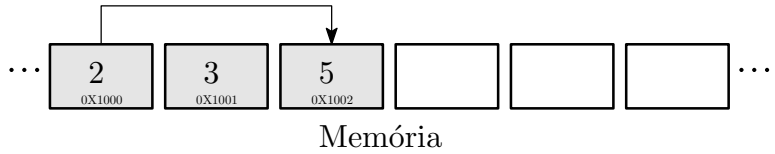
Memória

Na prática, para localizar um vetor, o computador só precisa saber o endereço na memória do seu primeiro elemento. Mas como ele faz para acessar (BUSCAR) um elemento?

Estruturas de dados

Vetores

$$v[2] = 0X1000 + 2 = 0X1002$$

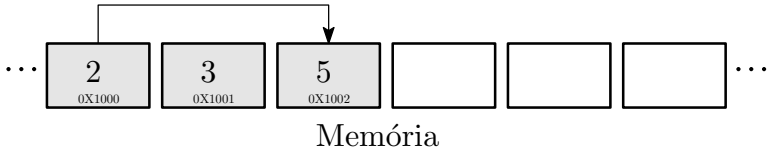


Como os elementos em um vetor estão contíguos, basta selecionarmos o primeiro endereço de memória + o elemento que queremos buscar (isso é feito pelo SO).

Estruturas de dados

Vetores

$$v[2] = 0X1000 + 2 = 0X1002$$

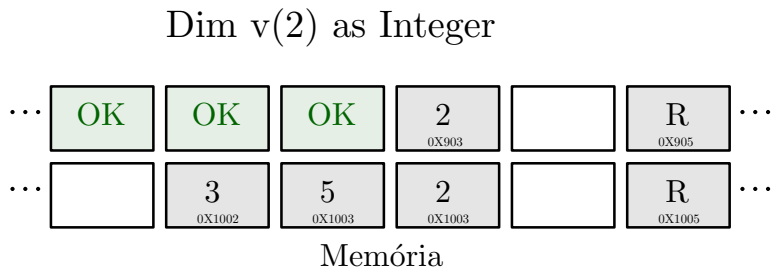


Dizemos que a complexidade para a operação busca em um vetor é O(1). De forma bem simplificada e matemáticamente errônea, isso quer dizer que é necessário **uma operação** para se realizar a busca em um vetor de *n* elementos.

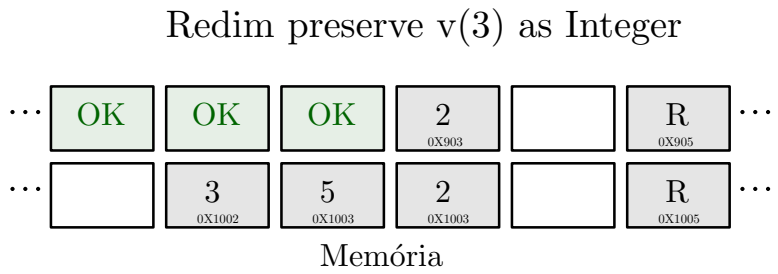
Dim v(2) as Integer

Memória

O que você pode dizer sobre a situação acima: queremos dimensionar um vetor com 3 elementos com a memória como mostrado acima. O que vai acontecer?



Da forma como está, não é possível! O sistema operacional deve buscar um espaço na memória com **3 elementos livres contíguos**.



E agora, se quiséssemos adicionar um elemento a mais no vetor (usando o Redim preserve), o que o sistema operacional deverá fazer?

Estruturas de dados

Vetores

Isso mesmo:

1. Remover todos os elementos de onde estão.
2. Buscar um novo espaço na memória capaz de comportá-los.
3. Realocar todos os elementos novamente.

Ou seja, sempre que possível...

Estruturas de dados

Vetores

Isso mesmo:

1. Remover todos os elementos de onde estão.
2. Buscar um novo espaço na memória capaz de comportá-los.
3. Realocar todos os elementos novamente.

Ou seja, sempre que possível...

NÃO USE REDIM PRESERVE!!!!

EXERCÍCIO: Implemente as seguintes tarefas em VBA:

1. Leia os elementos da primeira coluna da planilha em um vetor de inteiros. O primeiro elemento (Célula A1) indica quantos elementos abaixo dele serão lidos, e os outros (A2, A3, ...) indicam os números a serem lidos. **Armazene os números em um vetor**.
2. Passe toda a rotina de leitura dos elementos para uma função, que simplesmente retorna o vetor lido.
3. Use a função para ler um vetor de números e imprimir na janela de verificação o menor elemento.

Estruturas de dados

Collections (listas)

Mas professor...em alguns casos **não sabemos de antemão o número de elementos que vamos precisar armazenar**, como armazená-los sem o amaldiçoado Redim preserve?

Estruturas de dados

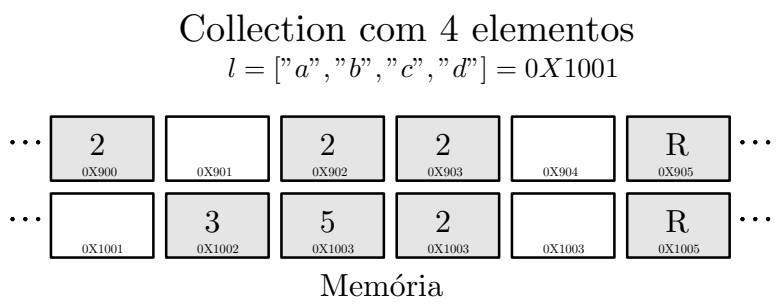
Collections (listas)

Mas professor...em alguns casos **não sabemos de antemão o número de elementos que vamos precisar armazenar**, como armazená-los sem o amaldiçoado Redim preserve?

> Nesses casos podemos usar a estrutura de dados Collections, ou listas (linked-lists), para simplificar a nossa vida.

Estruturas de dados

Collections (listas)



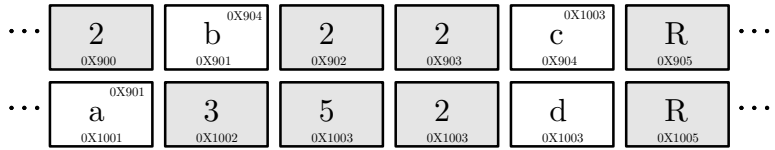
Diferentemente dos vetores, as listas não salvam os seus elementos em espaços contíguos na memória. Suponha uma lista de strings como mostrado acima (ainda é salvo o primeiro endereço da lista).

Estruturas de dados

Collections (listas)

Collection com 4 elementos

$l = [\text{"}a\text{"}, \text{"}b\text{"}, \text{"}c\text{"}, \text{"}d\text{"}] = 0X1001$



Memória

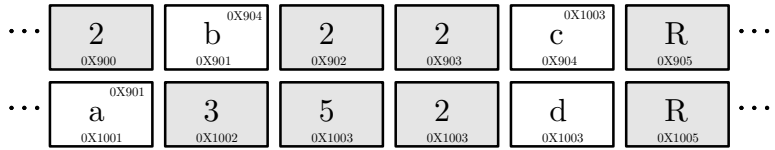
O sistema operacional pode usar qualquer espaço de memória para alocar seus elementos, como mostrado.

Estruturas de dados

Collections (listas)

Collection com 4 elementos

$l = [\text{"}a\text{"}, \text{"}b\text{"}, \text{"}c\text{"}, \text{"}d\text{"}] = 0X1001$

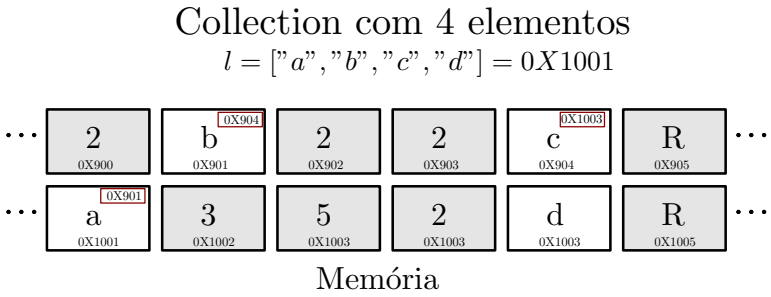


Memória

Mas como funcionará BUSCA por um elemento em uma lista? Como eles não são contíguos, não é possível simplesmente somar uma constante ao endereço de memória inicial.

Estruturas de dados

Collections (listas)



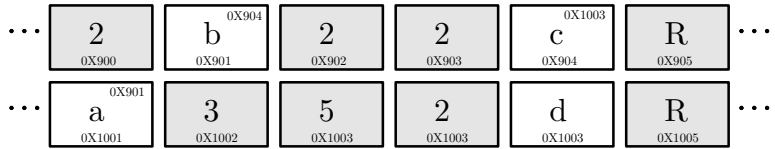
Note que além de armazenar as strings, cada elemento da lista também armazena um outro endereço: o endereço do próximo elemento da lista!

Estruturas de dados

Collections (listas)

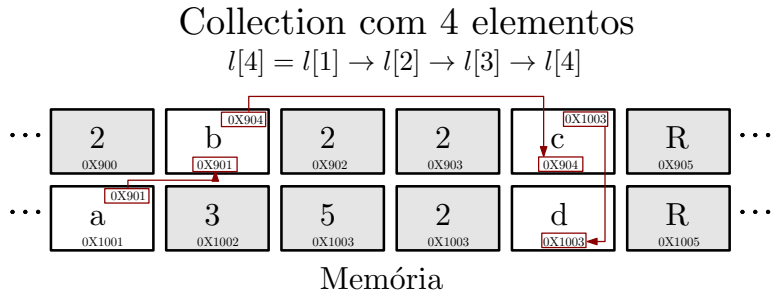
Collection com 4 elementos

$l[4] = l[1] \rightarrow l[2] \rightarrow l[3] \rightarrow l[4]$



Memória

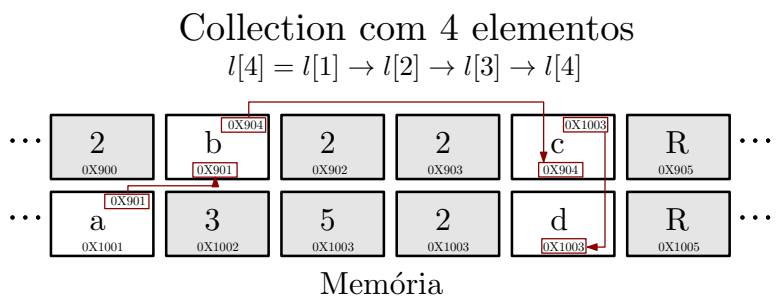
Assim, se precisarmos acessar o 4º elemento da lista (l[4] - sim, as listas começam em 1...) é necessário percorrer todos os elementos.



Pois não sabemos o endereço do elemento *l*[4], quem sabe é o elemento *l*[3]. E quem sabe o endereço de *l*[3] é *l*[2], e assim sucessivamente.

Estruturas de dados

Collections (listas)



Dizemos que a complexidade para a operação busca em uma lista é O(n). De forma bem simplificada e matemáticamente errônea, isso quer dizer que são necessárias **n operações** para se realizar a busca em uma lista de *n* elementos (no pior cenário).

Estruturas de dados

Collections (listas) - exemplos

Cria uma lista e adiciona 10 inteiros nela (de 0 a 9).

```
Sub collection()
   Dim list As Collection    'Declarando a lista
   Set list = New Collection 'Instanciando
   ' Dim list as New Collection (pode ser assim tambem, direto)

   For i = 0 To 9 'adicionando 10 elementos na lista
      list.Add i
   Next i

End Sub
```

Estruturas de dados

Collections (listas) - exemplos

Itera pelos elementos de uma lista (note que o índice começa em 1).

```
Sub collection()
   Dim list As Collection
   Set list = New Collection

   For i = 0 To 9
      list.Add i
   Next i

   ' Iterando pelos elementos da lista
   For i = 1 To list.Count 'Note que o indice comeca em 1
      Debug.Print list.Item(i)
   Next i
End Sub
```

Estruturas de dados

Collections (listas) - exemplos

Remove o último elemento da lista (indice list.Count).

```
Sub teste()
   Dim list As Collection
   Set list = New Collection

   For i = 0 To 9
      list.Add i
   Next i
   list.Remove (list.Count) 'Removendo o ultimo elemento da lista

   For i = 1 To list.Count
      Debug.Print list.Item(i)
   Next i
End Sub
```

Estruturas de dados

Collections (listas) - Aquecimento VBA

EXERCÍCIO: Implemente as seguintes tarefas em VBA:

1. Leia os elementos da primeira coluna da planilha em uma lista. Não existe uma indicação de quantos elementos a coluna têm. A indicação de fim dos elementos é a primeira célula vazia encontrada.
2. Encontre o maior elemento da lista e o imprima na tela.
3. Salve todos os elementos da lista em um vetor de forma invertida (o primeiro elemento da lista é o último do vetor).

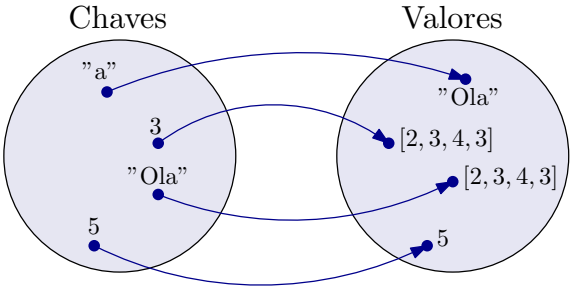
Estruturas de dados

Quando usar vetores e listas?

Devemos estudar a situação do algoritmo (a tarefa que precisamos fazer) e avaliar qual a melhor estrutura de dados a ser usada. De forma resumida, podemos pensar da seguinte maneira:

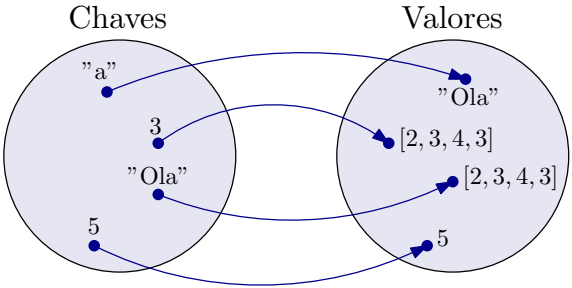
Situação	Vetores	Listas
Muita iteração nos elementos, buscas, etc...	Rápidos pela memória contígua	Lentas pois não conhecem os endereços
Toda hora uma nova inserção/remoção. Não sabemos o número de elementos totais.	Lentos, pois a cada nova alteração todos os elementos são removidos e reinseridos (pior cenário)	Razoáveis, não precisam adicionar elementos contíguos

Dicionários - exemplos



Os dicionários fazem o mapeamento de chaves a valores. A cada inserção, sempre alocamos os dois elementos juntos (chave e valor).

Dicionários



As chaves de um dicionário devem ser únicas. Os elementos de um dicionário são coletados via chave.

Estruturas de dados

Dicionários - exemplos

Criar dicionário, adicionar elementos e iterar pelas chaves.

```
Sub dicionario()

   Dic dic As New Dictionary ' Declarando o dicionario
   ' Adicionando chaves, valores (se inserir em um valor que ja existe vai
      substituir)
   dic.Add "b", 10       ' chave = "b", valor = 10
   dic.Add "c", "Cairo" ' chave = "c", valor = "Cairo"
   Debug.Print dic("b") ' Imprimindo o elemento da chave "b"

   ' Iterando pelos elementos (imprimindo as chaves)
   Dim k As Variant
   For Each k In dic.Keys()
      'dic(k) para usar o valor da chave
      Debug.Print k
   Next k

End Sub
```

Estruturas de dados

Dicionários - exemplos

Verifica se uma chave existe no dicionário.

```
Sub dicionario()

   Dic dic As New Dictionary ' Declarando o dicionario
   ' Adicionando chaves, valores (se inserir em um valor que ja existe vai
       substituir)
   dic.Add "b", 10       ' chave = "b", valor = 10
   dic.Add "c", "Cairo" ' chave = "c", valor = "Cairo"
   Debug.Print dic("b") ' Imprimindo o elemento da chave "b"

   If dic.Exists("c") Then ' Verificando se a chave "c" existe
      Debug.Print "Chave c existe"
   Else
      Debug.Print "Chave c nao existe"
   End If

End Sub
```

Estruturas de dados

Dicionários - exemplos

Remove elementos do dicionário pelas chaves.

```
Sub dicionario()

   Dic dic As New Dictionary ' Declarando o dicionario
   ' Adicionando chaves, valores (se inserir em um valor que ja existe vai
       substituir)
   dic.Add "b", 10       ' chave = "b", valor = 10
   dic.Add "c", "Cairo"  ' chave = "c", valor = "Cairo"
   Debug.Print dic("b") ' Imprimindo o elemento da chave "b"

   dic.Remove "vec" 'rmovendo um elemento pela chave
   If dic.Exists("c") Then ' Verificando se a chave "c" existe
      Debug.Print "Chave c existe"
   Else
      Debug.Print "Chave c nao existe"
   End If

End Sub
```

Estruturas de dados

Dicionários - exemplos

Cria um vetor de inteiros e adiciona o vetor no dicionário, iterando por seus elementos usando o próprio dicionário.

```
Sub dicionario()
   Dim v(2) As Integer
   Dim i As Integer
   ' Declarando o dicionario
   Dic dic As New Dictionary

   v(0) = 2
   v(1) = 3
   ' Adicionando chaves, valores (se inserir em um valor que ja existe vai
       substituir)
   dic.Add "vec", v        ' vetor
   ' Iterando no elemento vetor da chave 'vec'
   For i = 0 To UBound(dic("vec"))
      Debug.Print dic("vec")(i)
   Next i
End Sub
```

Estruturas de dados

Collections (listas) - Aquecimento VBA

EXERCÍCIO: Implemente as seguintes tarefas em VBA:

1 Considere um conjunto de dados salvo na planilha, como mostrado na Figura abaixo:

	A	B
1	Máquina	Tempo produzindo (min)
2	M1	30
3	M1	20
4	M1	10
5	M1	10
6	M2	20
7	M2	20
8	M2	10
9	M2	50
10	M3	10
11	M3	55
12	M3	60
13	M5	20
14	M5	30
15	M9	45

Os dados são de uma linha de produção que trabalhou por uma semana. As colunas indicam quais máquinas foram utilizadas para processar uma determinada OP, e os tempos de processamento. Note que a mesma máquina pode ser usada diversas vezes. Crie um código que mostre o tempo total que cada máquina ficou processando nesta semana. OBS: O método deve ser genérico, ou seja, ler qualquer número de dados que estejam na planilha.

2 Considerando a função de leitura de um vetor implementada nos últimos exercícios. Calcule e imprima a frequência de ocorrência de cada número no vetor.

Estruturas de dados

Type

```
Dim nome as String
Dim idade as Integer
Dim RG as String
```

Suponha que vamos armazenar os dados de clientes. Os dados são mostrados acima: nomes, idade e RG.

Estruturas de dados

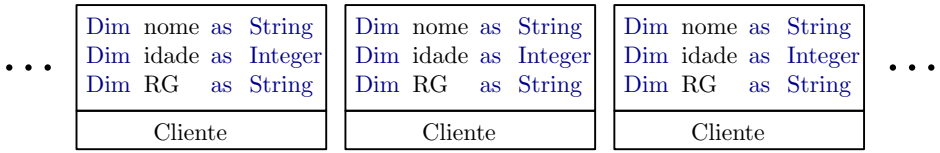
Type

Dim nome as String Dim idade as Integer Dim RG as String
Cliente

Podemos pensar nas Types como uma variável que nós mesmos criamos. Podemos criar uma variável do tipo **Cliente**, que tem dentro de si as 3 informações dos clientes (nome, idade e RG).

Estruturas de dados

Type



Ainda, podemos criar um vetor de clientes, cada elemento do vetor terá as 3 variáveis do cliente (nome, idade e RG).

Estruturas de dados

Type - exemplos

Criando uma Type com as variáveis nome(string) compras(Integer) e RG(string). OBS: Isso deve ser declarado acima de todas as subs/functions.

```
Option Explicit
Type cliente
   nome As String
   compras As Integer
   RG As String
End Type
```

Estruturas de dados

Type - exemplos

Usando a Type criada e adicionando valores às variáveis.

```
Sub adiciona_clientes()

   Dim c1 As cliente
   c1.nome = "Alexandre"
   c1.compras = 10
   c1.RG = "47585658214"

End Sub
```

Estruturas de dados

Type - exemplos

Adiciona um vetor de clientes e imprime seus nomes e RGs.

```
Sub adiciona_clientes()
    Dim v_clientes(1) As cliente 'vetor de clientes

   ' Informacoes cliente 1
   v_clientes(0).nome = "Michael"
   v_clientes(0).compras = 10
   v_clientes(0).RG = "47585658214"

   ' Informacoes cliente 2
   v_clientes(1).nome = "Dwight"
   v_clientes(1).compras = 10
   v_clientes(1).RG = "47585658215"

   Dim i As Integer
   For i = 0 To UBound(v_clientes)
      Debug.Print "Nome: " & v_clientes(i).nome & " RG : " & v_clientes(i).RG
   Next i

End Sub
```

Estruturas de dados

Type - Aquecimento VBA

EXERCÍCIO: Implemente as seguintes tarefas em VBA:

1 Considere um conjunto de dados salvo na planilha, como mostrado na Figura abaixo:

	A	B	C	D	E
1	Funcionario	Tempo de empresa (meses)	Salario	Bonificação (%)	Novo salario
2	Dmitri	2	1000	??	??
3	Ivan	1	1500	??	??
4	Alexiei	4	2000	??	??
5	Raskolnikov	8	3000	??	??
6	Iliucha	5	2500	??	??

Os dados são de funcionários de uma empresa, com o tempo de empresa (**T**) e salário (**S**).

Estruturas de dados

Type - Aquecimento VBA

Os funcionários receberão uma taxa de bonificação (**B**%), dada pela formula:

$$B = T_i C \tag{1}$$

Em que

$$\begin{cases} T_i : \text{Tempo de empresa do funcionário } i \\ C : \text{Coeficiente de bonificação} \end{cases}$$

E o coeficiente de bonificação (**C**) é dado por:

$$C = \frac{\sum S_i}{max\{S_i\}} \frac{1}{100} \tag{2}$$

Em que:

$$\begin{cases} \sum S_i : \text{Soma de todos os salários} \\ max\{S_i\} : \text{Maior salário} \end{cases}$$

De forma que o aumento efetivo do funcionário é a taxa de bonificação (B) multiplicada pelo seu salário atual. Implemente um código que calcule as bonificações de todo funcionário, usando Types.

Input de texto

Input de texto - exemplos

Lê um arquivo de texto e imprime seu conteúdo na janela de verificação.

```
Sub le_arquivo_de_texto()
   Dim leitor As New FileSystemObject
   Dim arquivo As Object
   Dim caminho As String

   caminho = "C:\Users\Usuario\Documents\Alexandre\Nomes.txt"
   Dim line As String
   Set arquivo = leitor.OpenTextFile(caminho, ForReading) 'abre o arquivo

   While Not arquivo.AtEndOfStream ' le linha a linha do arquivo
      line = arquivo.ReadLine      ' toda linha e armazenada em uma string '
          linha'
      Debug.Print line             ' imprimindo a linha lida
   Wend
   arquivo.Close 'fecha o arquivo
End Sub
```

Input de texto

Input de texto - exemplos

Lê um arquivo de texto, atribuindo os elementos de cada linha (separados por espaço) em um vetor v_split.

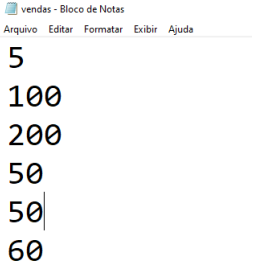
```
Sub le_arquivo_de_texto_vetor()
   Dim leitor As New FileSystemObject
   Dim arquivo As Object
   Dim caminho As String
   Dim v_split() As String
   caminho = "C:\Users\Usuario\Documents\Alexandre\Nomes.txt"
   Dim line As String
   Set arquivo = leitor.OpenTextFile(caminho, ForReading) 'abre o arquivo
   While Not arquivo.AtEndOfStream ' le linha a linha do arquivo
      line = arquivo.ReadLine      ' toda linha e armazenada em uma string '
          linha'
      v_split = Split(line, " ")   ' o vetor v_split com os elementos separados
           da linha
   Wend
   arquivo.Close 'fecha o arquivo
End Sub
```

Input de texto

Input de texto - Aquecimento VBA

EXERCÍCIO: Implemente as seguintes tarefas em VBA:

1 Considere um arquivo de texto com as indicações de vendas de um produto, como na Figura abaixo:



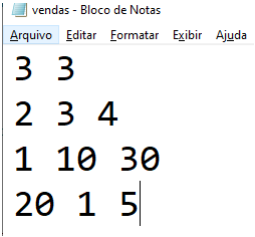
Em que o primeiro elemento indica a quantidade de números a serem lidos. Aloque os elementos em um vetor.

Input de texto

Input de texto - Aquecimento VBA

2 Considere agora um mesmo tipo de arquivo, porém só com as demandas, sem indicar o número de valores a serem lidos. Armazene os valores na estrutura de dados mais adequada.

3 Considere o arquivo abaixo, com os dados de uma matriz de inteiros:



Sendo que os dois primeiros elementos indicam o número de linhas e colunas da matriz. Leia os dados e aloque em uma matriz de inteiros.

Input de texto

Input de texto - Aquecimento VBA

4 Novamente, considere os dados de uma matriz como no exercício anterior, porém sem a indicação do número de linhas e colunas. Leia o arquivo e salve em uma matriz de inteiros.