# Introdução à Lógica de Programação

**Prof. Lucas Amparo Barbosa**

**Semestre letivo 2020.2**

## Mais dados, mais informação

- Se você tiver que informar o nome de um aluno, como isso poderia ser feito?

# Mais dados, mais informação

- Se você tiver que informar o nome de um aluno, como isso poderia ser feito?

```
Algoritmo "exemplo_aula"

Var

nome : caractere

Inicio

escreval("Digite o nome do aluno")
leia(nome)

escreval("Nome do Aluno: ", nome)

Fimalgoritmo
```

## Mais dados, mais informação

- E se você precisar informar
  o nome de 5 alunos?

# Mais dados, mais informação

- E se você precisar informar o nome de 5 alunos?

```
Algoritmo "exemplo_aula"

Var

nome1, nome2, nome3 : caractere
nome4, nome5 : caractere

Inicio

escreval("Digite o nome do aluno 1")
leia(nome1)

escreval("Digite o nome do aluno 2")
leia(nome2)

escreval("Digite o nome do aluno 3")
leia(nome3)

escreval("Digite o nome do aluno 4")
leia(nome4)

escreval("Digite o nome do aluno 5")
leia(nome5)

escreval("Nome do Aluno 1: ", nome1)
escreval("Nome do Aluno 2: ", nome2)
escreval("Nome do Aluno 3: ", nome3)
escreval("Nome do Aluno 4: ", nome4)
escreval("Nome do Aluno 5: ", nome5)

Fimalgoritmo
```

# Mais dados, mais informação

- E se você precisar informar o nome de 5 alunos?

**ATÉ AQUI, TUDO TRANQUILO NÉ?**

```
Algoritmo "exemplo_aula"

Var

nome1, nome2, nome3 : caractere
nome4, nome5 : caractere

Inicio

escreval("Digite o nome do aluno 1")
leia(nome1)

escreval("Digite o nome do aluno 2")
leia(nome2)

escreval("Digite o nome do aluno 3")
leia(nome3)

escreval("Digite o nome do aluno 4")
leia(nome4)

escreval("Digite o nome do aluno 5")
leia(nome5)

escreval("Nome do Aluno 1: ", nome1)
escreval("Nome do Aluno 2: ", nome2)
escreval("Nome do Aluno 3: ", nome3)
escreval("Nome do Aluno 4: ", nome4)
escreval("Nome do Aluno 5: ", nome5)

Fimalgoritmo
```

## Mais dados, mais informação

- E se você precisar informar o nome de 100 alunos?

# Mais dados, mais informação

- E se você precisar informar o nome de 100 alunos?

```
Algoritmo "exemplo_aula"

Var

nome1, nome2, nome3 : caractere
nome4, nome5, nome6 : caractere
[...]
nome98, nome99, nome100 : caractere

Inicio

escreval("Digite o nome do aluno 1")
leia(nome1)

escreval("Digite o nome do aluno 2")
leia(nome2)

[...]

escreval("Digite o nome do aluno 100")
leia(nome100)

escreval("Nome do Aluno 1: ", nome1)
escreval("Nome do Aluno 2: ", nome2)
[...]
escreval("Nome do Aluno 100: ", nome100)

Fimalgoritmo
```

## Mais dados, mais informação

- E se você precisar informar o nome de 100 alunos?
- CALMA! Dá para fazer um código melhor que isso…

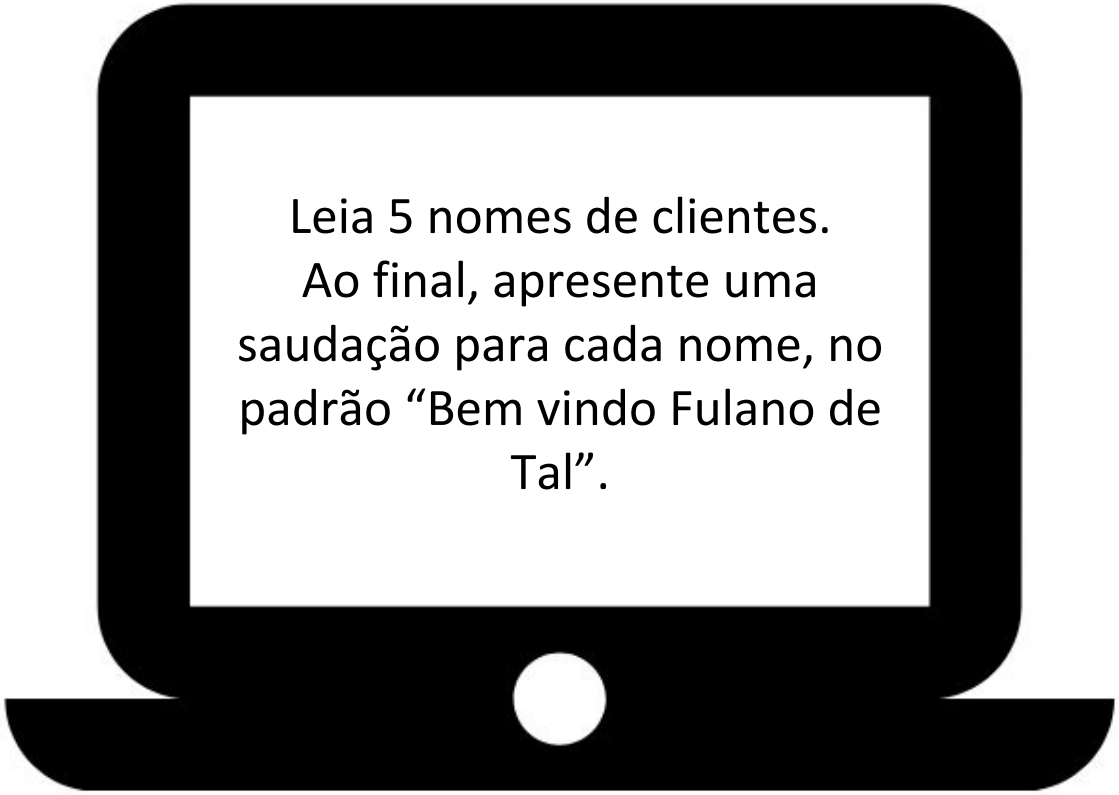
## Mais dados, mais informação

- E se você precisar informar o nome de 100 alunos?
- CALMA! Dá para fazer um código melhor que isso…
- Podemos utilizar **VETORES**.
  - É como se “empilhássemos” variáveis em uma única só.
  - Todas tem o mesmo tipo de dado básico

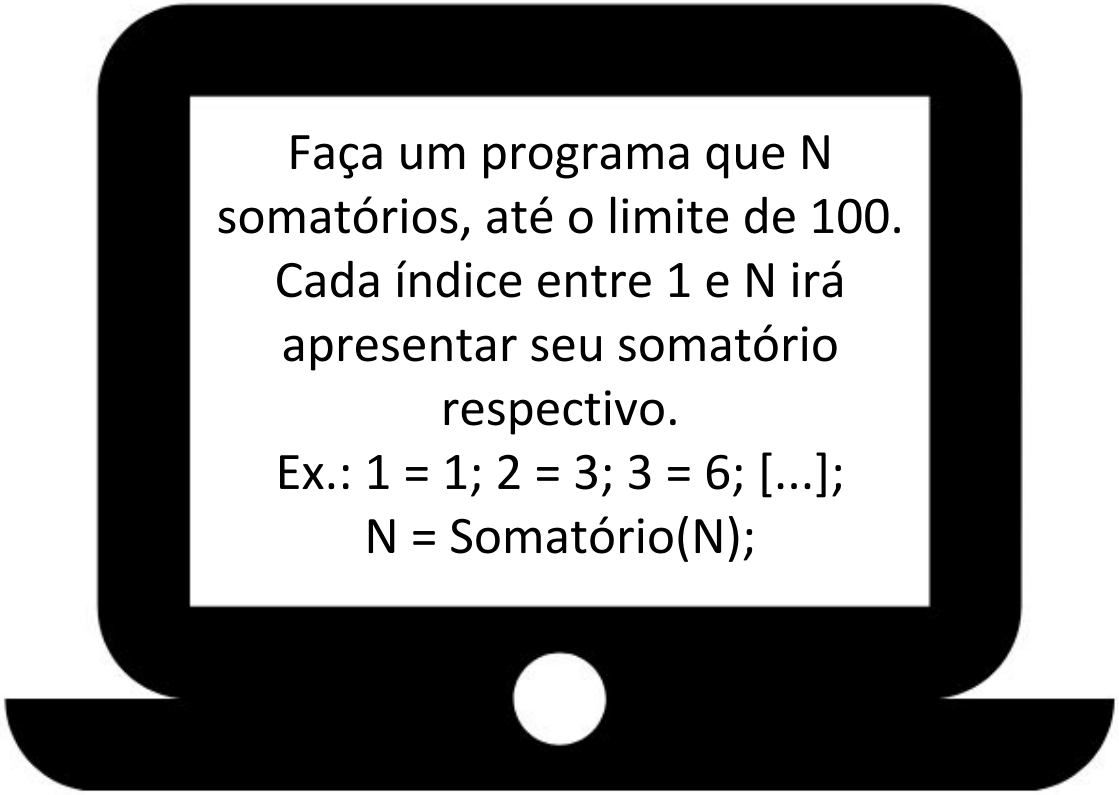
```
Algoritmo "exemplo_aula"

Var

nomes : vetor [1..100] de caractere
contador : inteiro

Inicio

para contador de 1 ate 100 faca
  escreval("Digite o nome do aluno n ", contador, ": ")
  leia(nomes[contador])
fimpara

para contador de 1 ate 100 faca
  escreval("Nome do Aluno ", contador, ": ", nomes[contador])
fimpara

Fimalgoritmo
```


Nome do vetor
vetor[i]
Índice do vetor
(em que posição estamos??)

# Prática 1: Apresentar nomes



# Prática 2: Somatórios



# Mais dados, mais informação

- E se ao invés do nome de 100 alunos, precisássemos armazenar as médias de cada unidade?
  - Um vetor para cada média?

# Mais dados, mais informação

- E se ao invés do nome de 100 alunos, precisássemos armazenar as médias de cada unidade?
  - Um vetor para cada média?

```
Algoritmo "exemplo_aula"

Var

unidade1 : vetor [1..100] de real
unidade2 : vetor [1..100] de real
unidade3 : vetor [1..100] de real
contador : inteiro
media : real

Inicio

para contador de 1 ate 100 faca
  escreval("Digite a 1a media do aluno n ", contador, ": ")
  leia(unidade1[contador])
fimpara

para contador de 1 ate 100 faca
  escreval("Digite a 2a media do aluno n ", contador, ": ")
  leia(unidade2[contador])
fimpara

para contador de 1 ate 100 faca
  escreval("Digite a 3a media do aluno n ", contador, ": ")
  leia(unidade3[contador])
fimpara

Fimalgoritmo
```

## Mais dados, mais informação

- Agora imagine que você trabalha com pesquisas de preços de mercado… Seu chefe te passou a missão de levantar os valores de 100 produtos em 50 mercados da cidade.

## Mais dados, mais informação

- Agora imagine que você trabalha com pesquisas de preços de mercado… Seu chefe te passou a missão de levantar os valores de 100 produtos em 50 mercados da cidade.

- Como ficaria esse código?
  - Um vetor para cada mercado?
  - Um vetor para cada produto?

## Mais dados, mais informação

- Agora imagine que você trabalha com pesquisas de preços de mercado… Seu chefe te passou a missão de levantar os valores de 100 produtos em 50 mercados da cidade.

- Como ficaria esse código?
  - Um vetor para cada mercado?
  - Um vetor para cada produto?

- Nenhuma dessas estratégias é boa.
  - Vetores podem ter **mais de uma dimensão**
  - Assim, recebem o nome de **Matrizes**

```
Algoritmo "exemplo_aula"

Var

precos : vetor [1..100,1..50] de real
lin, col : inteiro

Inicio

para lin de 1 ate 100 faca
  para col de 1 ate 50 faca
    escreval("Preço produto ", lin, " no mercado ", col)
    leia(precos[lin][col])
  fimpara
fimpara

Fimalgoritmo
```

# Prática 3: Calcular a média

A turma possui 10 alunos.
Leia as médias das 4 unidades e calcule a média final de cada aluno.
A média é ponderada, com peso 3 para unidades pares e 2 para unidades ímpares.

## Mais dados, mais informação

- E se eu quiser vincular variáveis de tipos diferentes?
  - Podemos “amarrar” pelo índice dos vetores
    - nome[n] é o nome do aluno que tem as notas salvas em notas[n]
    - preco[i][j] é o valor do produto de produto[i] no mercado[j]
  - Funcional, mas pouco elegante
  - Pode não representar os dados da forma mais legível

## Mais dados, mais informação

- E se eu quiser vincular variáveis de tipos diferentes?
  - Podemos “amarrar” pelo índice dos vetores
    - nome[n] é o nome do aluno que tem as notas salvas em notas[n]
    - preco[i][j] é o valor do produto de produto[i] no mercado[j]
  - Funcional, mas pouco elegante
  - Pode não representar os dados da forma mais legível

- Para melhorar isso, utilizamos os **registros**
  - Assim podemos ter vários tipos de dados “amarrados” em uma única variável

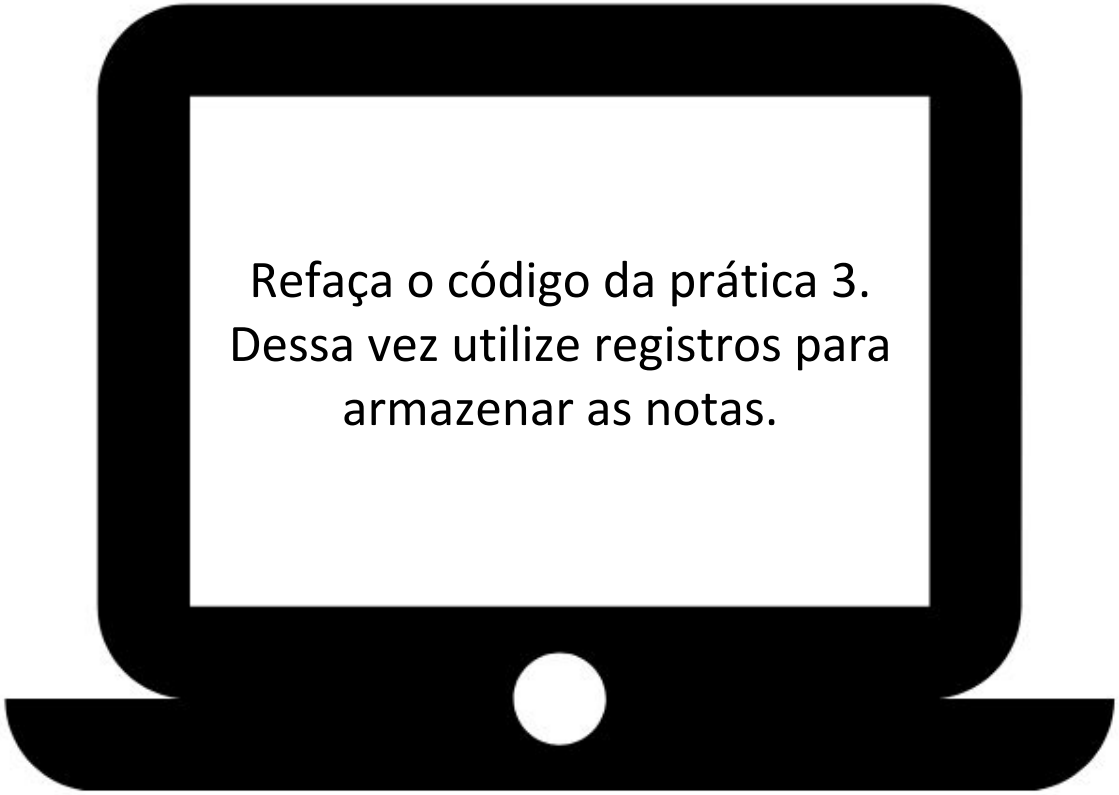
## Mais dados, mais informação

- Como montar um programa que simule uma agenda de contatos?
  - Temos 10 contatos a ser salvo
  - Cada contato tem:
    - Nome
    - Celular
    - e-Mail
  - Como armazenar tudo?

```
Algoritmo "exemplo_aula"

Var

nomes : vetor [1..10] de caractere
celular : vetor [1..10] de caractere
email : vetor [1..10] de caractere
contador : inteiro

Inicio

para contador de 1 ate 10 faca
  escreval("Cadastro No ", contador)
  escreval("Insira o Nome: ")
  leia(nomes[contador])
  escreval("Insira o Celular:")
  leia(celular[contador])
  escreval("Insira o email:")
  leia(email[contador])
fimpara

para contador de 1 ate 10 faca
  escreval("Nome:       ", nomes[contador])
  escreval("Celular:    ", celular[contador])
  escreval("e-Mail:     ", email[contador])
  escreval("")
fimpara

Fimalgoritmo
```

```
Algoritmo "exemplo_aula"
Tipo cadastro = registro
  nome   : caractere
  celular: caractere
  mail   : caractere
fimregistro

Var

agenda : vetor [1..10] de cadastro
contador : inteiro

Inicio

para contador de 1 ate 10 faca
  escreval("Cadastro No ", contador)
  escreval("Insira o Nome: ")
  leia(agenda[contador].nome)
  escreval("Insira o Celular:")
  leia(agenda[contador].celular)
  escreval("Insira o email:")
  leia(agenda[contador].mail)
fimpara

para contador de 1 ate 10 faca
  escreval("Nome:       ", agenda[contador].nome)
  escreval("Celular:    ", agenda[contador].celular)
  escreval("e-Mail:     ", agenda[contador].mail)
  escreval("")
fimpara

Fimalgoritmo
```

# Prática 4: Média 2 - O inimigo agora é outro



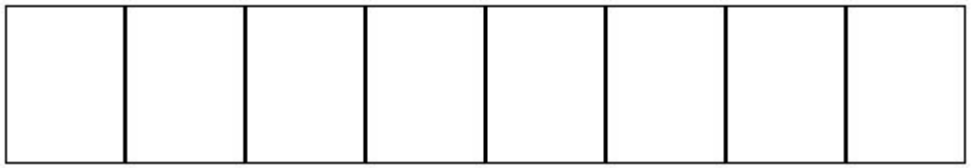
# Complexidade de Algoritmos

- É possível comparar dois códigos?
  - Medir eficiência antes de mesmo de executá-lo?

- Sim, é possível. E para entender como isso pode ser feito, iremos analisar o Problema da Busca.

# Complexidade de Algoritmos - Problema da Busca

- Objetivo: Encontrar o número X numa lista de tamanho N
- 1ª Possibilidade: A lista está **desordenada**
  - Precisaremos passar posição por posição



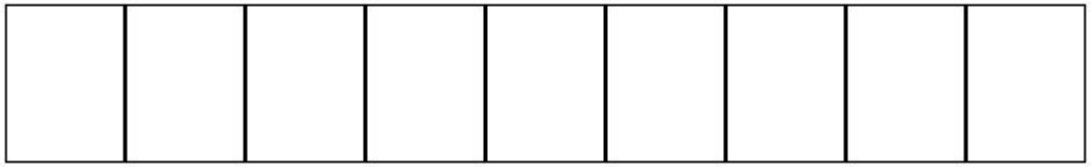
  - Melhor Caso: X logo na primeira casa


  - Pior Caso: X na última casa

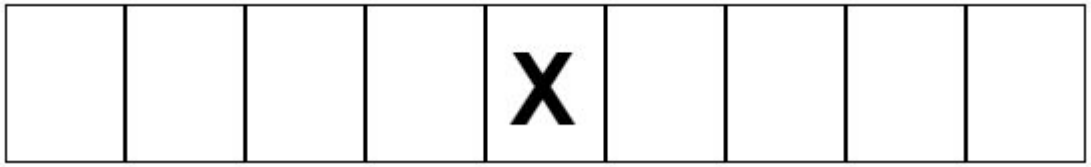
# Complexidade de Algoritmos - Problema da Busca

- Objetivo: Encontrar o número X numa lista de tamanho N

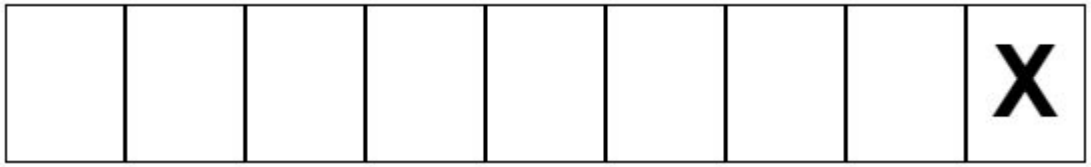
- 2ª Possibilidade: A lista está **ordenada**
  - Se eu sei como a lista está ordenada, posso descartar parte dos valores dependendo da relação entre X e o valor atual.



  - Melhor Caso: Logo na primeira casa




  - Pior Caso: Na última casa



# Complexidade de Algoritmos - Problema da Busca

- Objetivo: Encontrar o número X numa lista de tamanho N

- Para analisar a complexidade, devemos pensar sempre no **pior caso**
  - Para ambas as possibilidades, o pior caso está em encontrar X somente na última tentativa
    - Qual a diferença então?

# Complexidade de Algoritmos - Problema da Busca

- Objetivo: Encontrar o número X numa lista de tamanho N

- Para analisar a complexidade, devemos pensar sempre no **pior caso**
  - Para ambas as possibilidades, o pior caso está em encontrar X somente na última tentativa
    - Qual a diferença então?

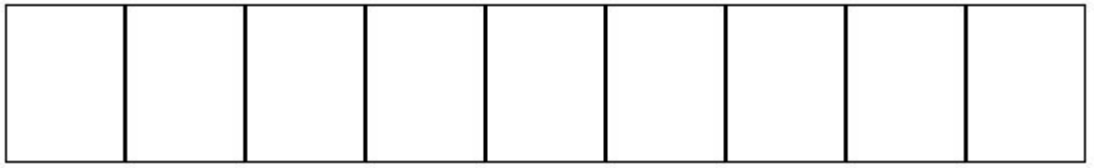
**QUANTAS ETAPAS DEMORA PARA CHEGAR NA ÚLTIMA ETAPA!!!!**

# Complexidade de Algoritmos - Problema da Busca

- Na primeira possibilidade, precisamos passar por **todos os elementos**

- Em função de N, teremos que passar por todos os N elementos
  - Utilizamos a notação O(n) para descrever um código com esse comportamento

- Na segunda possibilidade, não passamos por todos os elementos



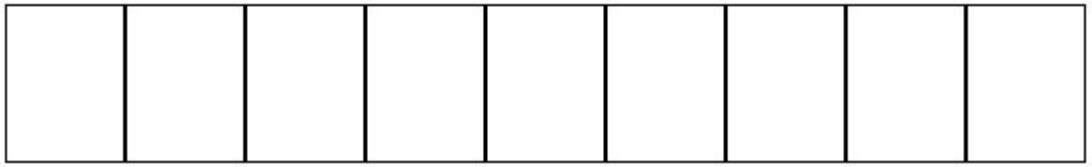
- Novamente em função de N, quantos elementos iríamos passar?
  - Com 9 elementos, passamos por 4;
  - Com 15 elementos, passaríamos por 4 também;
  - E se tivéssemos 50?

# Complexidade de Algoritmos - Problema da Busca

- Na primeira possibilidade, precisamos passar por **todos os elementos**

- Em função de N, teremos que passar por todos os N elementos
  - Utilizamos a notação O(n) para descrever esse comportamento

- Na segunda possibilidade, não passamos por todos os elementos



- Novamente em função de N, quantos elementos iríamos passar?
  - Com 9 elementos, passamos por 4;
  - Com 15 elementos, passaríamos por 4 também;
  - E se tivéssemos 50?
  - Esse crescimento se assemelha a uma função logarítmica, logo O(log n)

# Complexidade de Algoritmos - Problema da Busca

- Quem é mais eficiente?
  - Utilizamos o conhecimento de crescimento de funções para escolher
  - A função O(n) “cresce” muito mais rápido que O(log n)

| N | O(N) | O(log N) |
|---|---|---|
| 10 | 10 | 3.32 |
| 100 | 100 | 6.64 |
| 1000 | 1000 | 9.96 |

  - Assim, a busca que acontece quando a lista estiver ordenada é significativamente mais eficiente do que a busca na lista não ordenada
    - A busca na lista ordenada recebe o nome de **busca binária**
    - A busca na lista não ordenada se chama **busca sequencial**
- Qual a melhor opção para ser utilizado no seu código? **Depende!**

# Complexidade de Algoritmos

- Vamos rever o problema da Prática 2

```
Algoritmo "exemplo_aula"

Var

n, cont1, cont2, soma : inteiro

Inicio

escreval("Infome o limite do somatório (Max. 100):")
leia(n)

para cont1 de 1 ate n faca
  escreva("Somatório de ", cont1, " = ")
  soma := 0
  para cont2 de 1 ate cont1 faca
    soma := soma + cont2
  fimpara
  escreval(soma)
fimpara

Fimalgoritmo
```

# Complexidade de Algoritmos

- Vamos rever o problema da Prática 2

```
Algoritmo "exemplo_aula"

Var

n, cont1, cont2 : inteiro
soma : vetor [1..100] de inteiro

Inicio

escreval("Infome o limite do somatório (Max. 100):")
leia(n)

para cont1 de 1 ate n faca
  escreva("Somatório de ", cont1, " = ")
  soma[cont1] := 0
  para cont2 de 1 ate cont1 faca
    soma[cont1] := soma[cont1] + cont2
  fimpara
  escreval(soma[cont1])
fimpara

Fimalgoritmo
```

# Complexidade de Algoritmos

- Podemos reescrever ele, de forma mais eficiente, utilizando informações já armazenadas para evitar refazer contas que já foram feitas.

```
Algoritmo "exemplo_aula"

Var

n, cont1, cont2 : inteiro
soma : vetor [1..100] de inteiro

Inicio

escreval("Infome o limite do somatório (Max. 100):")
leia(n)

para cont1 de 1 ate n faca
  escreva("Somatório de ", cont1, " = ")
  se cont1 = 1 entao
    soma[cont1] := 1
  senao
    soma[cont1] := soma[cont1 - 1] + cont1
  fimse
  escreval(soma[cont1])
fimpara

Fimalgoritmo
```

```
Algoritmo "exemplo_aula"

Var

n, cont1, cont2 : inteiro
soma : vetor [1..100] de inteiro

Inicio

escreval("Infome o limite do somatório (Max. 100):")
leia(n)

para cont1 de 1 ate n faca
  escreva("Somatório de ", cont1, " = ")
  soma[cont1] := 0
  para cont2 de 1 ate cont1 faca
    soma[cont1] := soma[cont1] + cont2
  fimpara
  escreval(soma[cont1])
fimpara

Fimalgoritmo
```

```
Algoritmo "exemplo_aula"

Var

n, cont1, cont2 : inteiro
soma : vetor [1..100] de inteiro

Inicio

escreval("Infome o limite do somatório (Max. 100):")
leia(n)

para cont1 de 1 ate n faca
  escreva("Somatório de ", cont1, " = ")
  se cont1 = 1 entao
    soma[cont1] := 1
  senao
    soma[cont1] := soma[cont1 - 1] + cont1
  fimse
  escreval(soma[cont1])
fimpara

Fimalgoritmo
```

| Comparativo de Tempo em segundos | | |
|---|---|---|
| N | Somatório comum | Somatório Otimizado |
| 1000 | 0.0027 | 0.000008 |
| 10000 | 0.1527 | 0.0007 |
| 50000 | 3.7565 | 0.0003 |
| 500000 | 379.87 | 0.003 |

## Complexidade de Algoritmos - Para que saber isso?

- Com análises de complexidade, é possível criar códigos mais eficientes
- Ao imaginar programas simples, o impacto dessas otimizações é baixo.
- Agora, pense se o seu programa tenha uma entrada N na casa dos Milhões?
  - Milhões de clientes;
  - Milhões de produtos;
  - Milhões de históricos de compras;
- Uma análise de complexidade bem feita é o segredo de um sistema que sempre funciona e suporta bem uma quantidade massiva de usuários
  - Já perceberam como a maioria dos sistemas do governo não suportam uma quantidade grande de acessos simultâneos?
  - Será que eles analisam a complexidade?

# Para saber mais...

- Vetores e Matrizes
- Vetores e Strings
- Registros
- Intro a Complexidade (Barril!!)
- Complexidade de Algoritmos
- Comparando Ordenações