

Instalação Matlab

Instalação MATLAB:

<https://www.mathworks.com/academia/tah-portal/universidade-de-aveiro-40766421.html>

Use as suas credenciais de Utilizador Universal

Ajuda:

https://www.mathworks.com/support/contact_us.html?s_tid=tah_po_helpbutton_ua.pt

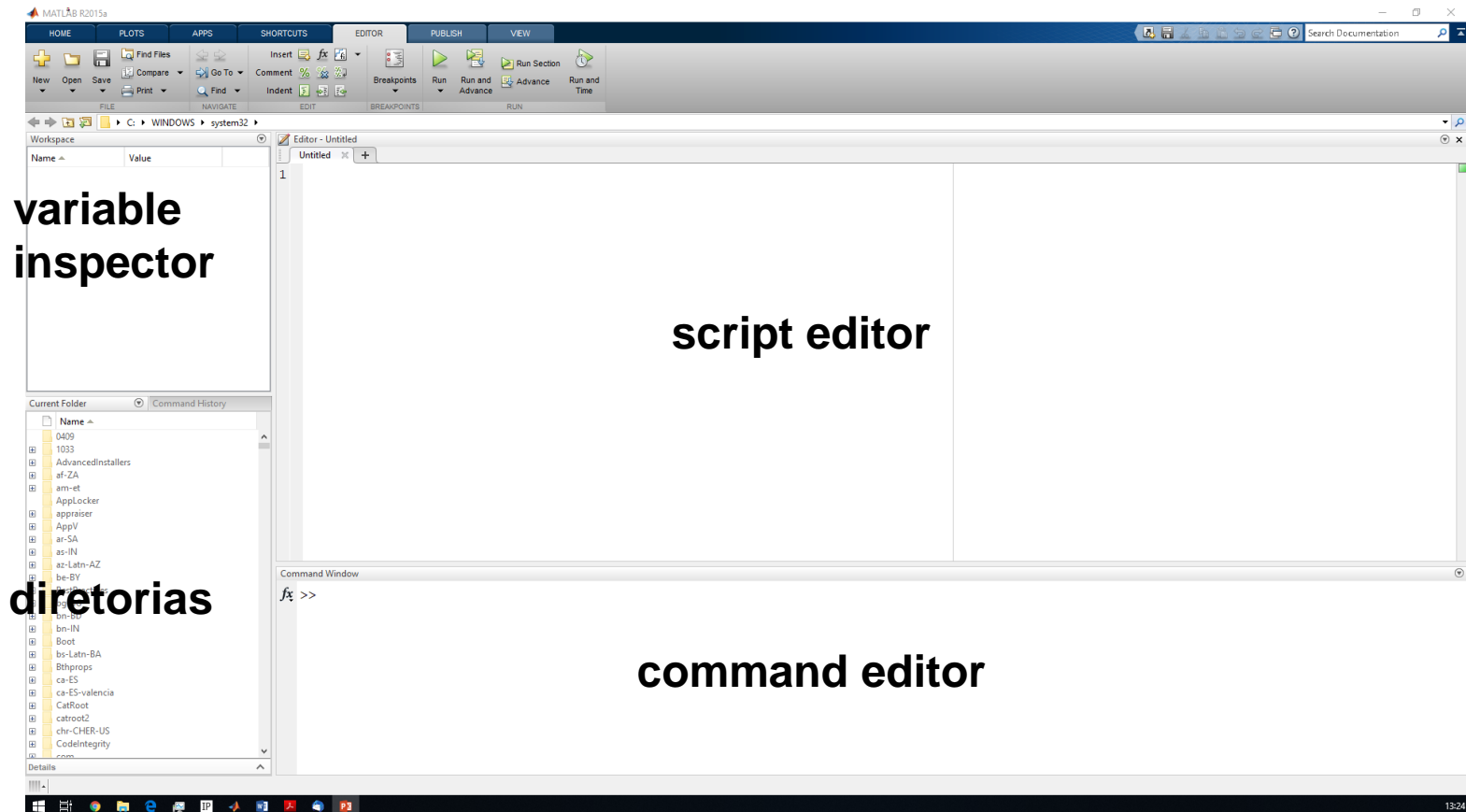
Aprenda MATLAB em duas horas:

Curso online [MATLAB Onramp](#)

Mais informações sobre este e outro software disponível:

<http://www.ua.pt/stic/page/16014>

0) IDE (integrated development environment))



1) Definição de Variáveis

NomeVariavel = Expressão

```
>> A=15  
>> a=2.3  
>> B2=A+4  
>> nome_longo = a*B2;  
>> matriz=[1 2 3 4; 5 6 7 8; 9 10 11 12]
```

- Uma variável é definida cada vez que é utilizada pela primeira vez. (noutras linguagens é preciso defini-las no início do programa ou de cada rotina)
- Os nomes são case sensitive
- Os nomes das variáveis começam com uma letra e podem conter até 31 caracteres.

Notas adicionais

- Os nomes devem ser criteriosamente escolhidos:
 - se forem variáveis com significado físico devem ter um nome que revele o seu conteúdo.
 - cada programador adquire os seus hábitos. Por exemplo, eu utilizo recorrentemente `i,j,k,ct`, para variáveis inteiras

É muito importante não chamar a coisas diferentes o mesmo nome (trivial mas “Never say No!”)

Variáveis

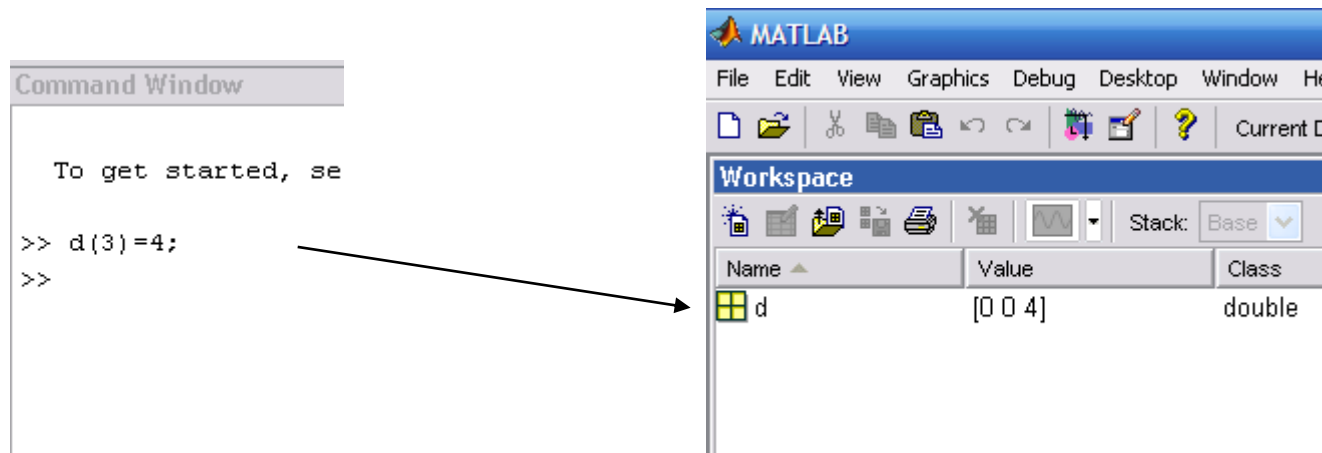
Há vantagem em “definir” a dimensão das variáveis desde o início.

As funções:

<code>zeros(N,M)</code>	gera uma matriz de zeros com N linha e M colunas
<code>ones(N,M)</code>	gera uma matriz de uns com N linha e M colunas
<code>rand(N,M)</code>	gera uma matriz de elementos aleatórios com N linha e M colunas
<code>eye(N)</code>	gera uma matriz identidade de dimensão N

reservam espaço de memória de acordo com as dimensões requeridas. Esse espaço é reservado (allocated) contiguamente.

O mesmo não se passa necessariamente no seguinte exemplo:



Se se definir uma nova variável *a*, e depois outra atribuição envolvendo o array *d* de dimensões ainda não especificadas, as regiões reservadas a *d*(1..3) podem não ficar contíguas a *d*(4..10)

```
Command Window

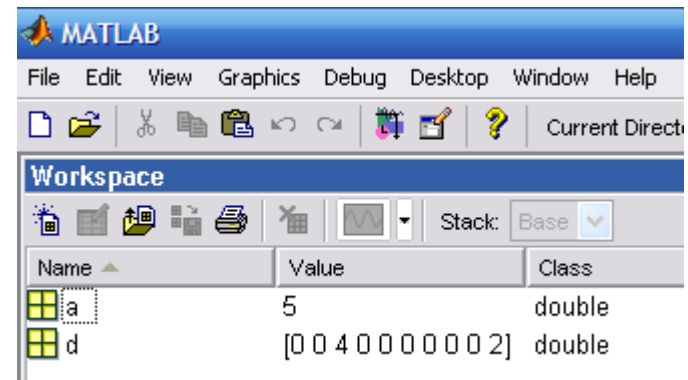
To get started, select M

>> d(3)=4;
>> a=5

a =

    5

>> d(10)=2;
>> |
```



Matrizes e Vectors

🕒⌘ Matriz:

- $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
- $B = [9, 8, 7; 6, 5, 4; 3, 2, 1]$

Operadores matriciais: $+$, $-$, $*$, $/$, $^$

Operadores elemento a elemento: $+$, $-$, $.*$, $./$, $.^$

Comandos Muito Úteis:

- `size(d)` vector com as dimensões associadas a cada índice
- `numel(d)` número de elementos de `d`
- `length(d)` número de elementos de um vector `d`

Exercício 1

Se introduzir:

```
x=zeros(4,5)
y=randn(6,4)
z=zeros(1,10)
x(2,4)=5
y(5,2)=3
z(3)=100
v=2:4
```

O que se obtém com:

```
size(x)           % número das dimensões de x
xy = [x y]

xy = [ x y']

a = xy(:,3)

b= xy(2,:)

numel(y)

c=y(y>0)

numel(y(y>0))

y(2,v)
```


Exercício 2

Se introduzir:

nome='X='

valor='23'

O que se obtém com:

```
size(nome)           % número das dimensões de nome
```

```
eq = [nome valor]
```

```
eq2= strcat(nome,valor)
```

```
eq==eq2
```

```
strcmp(eq,eq2)
```

```
eq(3:end)
```

```
eq(3:end)==23        % tipos diferentes?
```

```
s=abs(eq)             % codigos ascii
```

```
char(s(1))
```

```
strfind(eq,'23')      % retorna posição onde 23 está na string eq
```

```
valor+2               % erro ?
```

```
str2num(valor)+2      % número com o qual se fazem contas...
```

Gravar e ler dados guardados

1) Gravar

Formato otimizado .mat

```
save test.mat var      % command form  
save('test.mat','var') % function form
```

Formato otimizado .txt

```
save('test.txt','var','-ascii') % graver em format txt
```

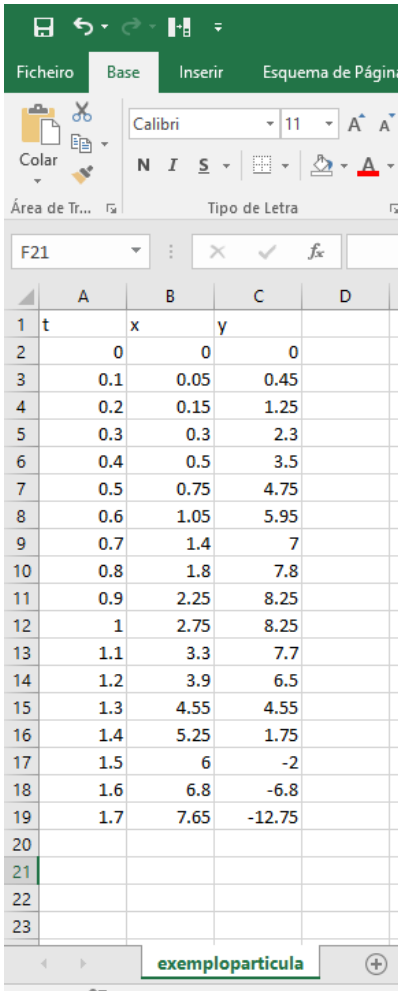
Caso não se mencionem as variáveis, grava todas as variáveis do workspace.

2) Ler

```
load(filename)  
load(filename,variables)  
load(filename,'-ascii')
```

S = load(____) -> neste caso a variável S fica com os dados no ficheiro

Recente introdução de mais uma estrutura de dados: **tabelas**



The screenshot shows a spreadsheet application with a table containing 4 columns (A, B, C, D) and 19 rows of data. The first row (row 1) contains the headers 't', 'x', and 'y' in columns A, B, and C respectively. The subsequent rows (rows 2 to 19) contain numerical values. The spreadsheet is titled 'exemploparticula'.

	A	B	C	D
1	t	x	y	
2	0	0	0	
3	0.1	0.05	0.45	
4	0.2	0.15	1.25	
5	0.3	0.3	2.3	
6	0.4	0.5	3.5	
7	0.5	0.75	4.75	
8	0.6	1.05	5.95	
9	0.7	1.4	7	
10	0.8	1.8	7.8	
11	0.9	2.25	8.25	
12	1	2.75	8.25	
13	1.1	3.3	7.7	
14	1.2	3.9	6.5	
15	1.3	4.55	4.55	
16	1.4	5.25	1.75	
17	1.5	6	-2	
18	1.6	6.8	-6.8	
19	1.7	7.65	-12.75	
20				
21				
22				
23				

- Gravar num ficheiro em formato csv
- Ler no matlab com o comando `readtable`

`T = readtable(filename)`

Será criada uma tabela T onde as variáveis T.t, T.x e T.y têm as variáveis nas colunas. Notar como a instrução `readtable` vai buscar à primeira linha o nome a atribuir às variáveis.

Nalguns casos a primeira linha pode não conter essa informação. Nesse caso, temos opções na instrução para evitar essa leitura:

`T = readtable(filename, 'readvariablenames', false)`

sendo os nomes das variáveis designados Var1, Var2, Var3...

Quando se definem as variáveis para uma tabela têm que ser colunas!!!!

Tal como noutras instruções, o readtable tem muitas opções. Podem por exemplo considerar ler uma tabela em que as linhas têm nomes.

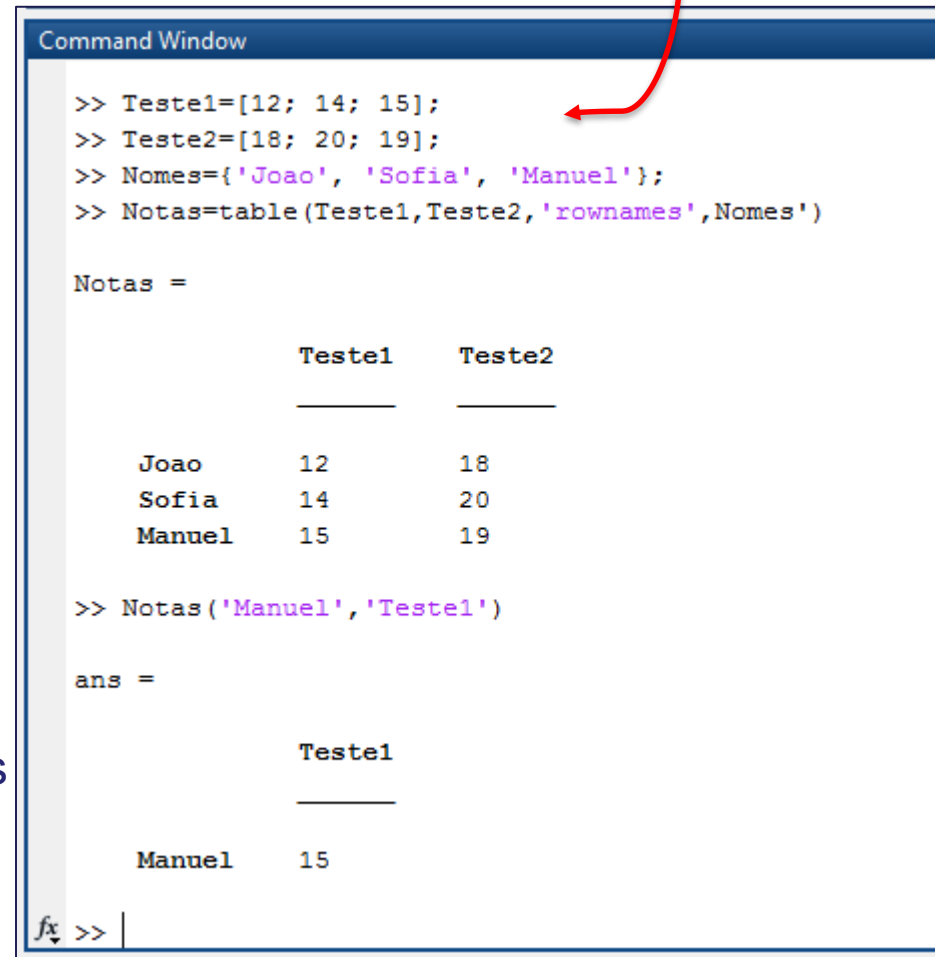
Nesse caso podem ler esses nomes usando:

`readtable(filename,'readrownames', true).`

Esta opção pode representar uma grande vantagem, pois desta forma pode-se ler o conteúdo de um valor da tabela usando essa indexação, que é muito mais legível.

Para gravar usar:

`>> writetable(Notas,'Notas.csv','writerownames',true)` **Notar que 1º é o nome da tabela e depois o nome do ficheiro!!!!**



```
>> Teste1=[12; 14; 15];
>> Teste2=[18; 20; 19];
>> Nomes={'Joao', 'Sofia', 'Manuel'};
>> Notas=table(Teste1,Teste2,'rownames',Nomes)

Notas =

           Teste1    Teste2
           _____    _____
Joao      12         18
Sofia     14         20
Manuel    15         19

>> Notas('Manuel','Teste1')

ans =

           Teste1
           _____
Manuel    15
```

Posso criar tabelas também recorrendo a variáveis e depois especificando nomes para as linhas (rownames) e colunas (variablenames).

```
>> a=[1 2 3 4]';
>> b={'a','b','c','dd'}';
>> T=table(a,b,'rownames',rows,'variablenames',{'Va','Vb'})
```

T =

	Va	Vb
	—	—
1st	1	'a'
2nd	2	'b'
3rd	3	'c'
4th	4	'dd'

Para aceder uso:

```
>> T.Va('1st')
```

ans =

1

```
>> T.Va(1)
```

ans =

1

Posso também usar o acesso normal mas obtenho uma sub-tabela, ou então com chavetas, e obtenho o elemento da tabela

```
>> T(1,1)
```

ans =

Va

—

1st 1

```
>> T{1,1}
```

ans =

1

Instrução Plot

```
>> x=[0 1 2 0];
```

```
>> y=[0 1 0 0];
```

```
>> plot(x,y)
```

Representação de uma linha unindo os vários pontos dos vectores x e y.

Instrução Pause

```
>> x=[0 1 2 0];
```

```
>> y=[0 1 0 0];
```

```
>> plot(x,y)
```

```
>> pause
```

```
>> x=[0 2 2 0];
```

```
>> y=[0 2 0 0];
```

```
>> plot(x,y)
```

A cada plot um novo gráfico é representado.

Fixar limites dos eixos.

```
>> AXIS([XMIN XMAX YMIN YMAX])
```

ou

```
>> xlim([XMIN XMAX])
```

```
>> ylim([YMIN YMAX])
```


Apagar figura ou sobrepor:

hold on

hold off

hold: alterna entre fazer hold on e hold off.

Parar a execução por t segundos:

pause(t)

Controlo de Fluxo num algoritmo

As estruturas de decisão permitem decidir o rumo do algoritmo:

1. Instrução IF

2. Instrução SWITCH

Para repetir uma parte do algoritmo várias vezes usam-se:

1. Loops FOR

2. Loops WHILE

Instrução IF (3 formas)

1. IF-END
2. IF-ELSE-END
3. IF-ELSEIF-ELSE-END

Síntaxe da Instrução IF-END:

IF condition (is true)

statement 1;

statement 2;

:

:

statement n;

END

Condições

Relacionam valores através de uma operação de relação ($>$, $>=$, $==$, $\sim=$) ou operação lógica

1. condicao1 & condicao2

2. condicao1 | condicao2

3. ~condicao

4. xor(condicao1, condicao2)

Exemplo

Calcular a seguinte soma $S = \sum_{n=1}^N \frac{1}{n^2}$

Com o ciclo **for** calculam-se os primeiros N termos do somatório

```
S= 0;  
for n= 1:N,  
    S= S + 1/n^2;  
end
```

Exemplo menos convencional

```
s=[24 2 -24];  
total=0;  
    for n= s  
        total= total + n;  
    end
```

Nos exemplos precedentes perdemos os valores dos cálculos intermédios. E se fôr importante guardá-los?

Exemplo:

Bola lançada para cima: $y(t) = y_0 + v_0 t - \frac{1}{2} g t^2$


```
1 clear all
2 close all
3 clc
4 %%% solução inadequada %%%
5 tempo=0:0.1:10; % interessa-nos analisar o movimento em décimas de segundo
6 v0= 10;
7 g=9.8;
8 y0=5;
9 figure(1)
10 for t=tempo
11     y= y0 + v0*t-0.5*g*t^2;
12     plot(y, '.', 'MarkerSize', 15);
13     pause(0.1)
14
15 end
16
```

Em situações mais complexas o cálculo de y requer um ciclo independente.

Exemplo:

Bola lançada para cima: $y(t) = y_0 + v_0 t - \frac{1}{2} g t^2$

```
1 clear all
2 close all
3 clc
4 %%% solução inadequada %%%
5 tempo=0:0.1:10; % interessa-nos analisar o movimento em décimas de segund
6 v0= 30;
7 g=9.8;
8 y0=5;
9 figure(1)
10 for i=1:length(tempo)
11     y(i)= y0 + v0*tempo(i)-0.5*g*tempo(i).^2;
12 end
13 ymax=max(y);
14 ymin=min(y);
15 for i=1:length(tempo)
16     plot(y(i),'.','MarkerSize',15);
17     ylim([ymin ymax])
18     pause(0.1)
19 end
20
```

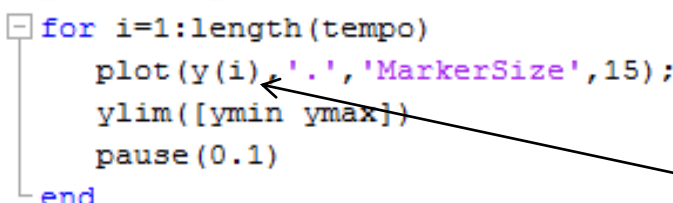


O caso precedente necessita de fixar os eixos. Para escolher eixos adequados preciso de fazer os cálculos primeiro e só depois executar a animação:

Exemplo:

Bola lançada para cima: $y(t) = y_0 + v_0 t - \frac{1}{2} g t^2$

```
1  clear all
2  close all
3  clc
4  %%% solução inadequada %%%
5  tempo=0:0.1:10;  % interessa-nos analisar o movimento em décimas de segundo
6  v0= 30;
7  g=9.8;
8  y0=5;
9  figure(1)
10 y= y0 + v0*tempo-0.5*g*tempo.^2;
11 ymax=max(y);
12 ymin=min(y);
13 for i=1:length(tempo)
14     plot(y(i),'.','MarkerSize',15);
15     ylim([ymin ymax])
16     pause(0.1)
17 end
```



percorre índices inteiros do vetor

Instruções Continue e Break

Continue : volta à condição de teste tal como que já tivesse completado a iteração

Break: salta fora do ciclo FOR

SWITCH

SWITCH condição a testar a igualdade

CASE ao que deve ser igual (opção 1)
instrução a executar se verdadeiro

CASE ao que deve ser igual (opção 2)
instrução a executar se verdadeiro

CASE ao que deve ser igual (opção 3)
instrução a executar se verdadeiro

Exemplo:

```
>> v= [2 3 4]
```

```
v =
```

```
     2     3     4
```

```
>> switch v(2)
```

```
case 2|
```

```
v(3)=20
```

```
case 3
```

```
v(3)=100
```

```
end
```

```
v =
```

```
     2     3    100
```

Ciclo WHILE

```
WHILE condição  
instrução I1;  
instrução I2;  
:  
instrução In;  
END
```

Ciclo `while`

Exemplo

Calcular a soma dada por $S = \sum_{n=1}^N \frac{1}{n^2}$

até que $\frac{1}{n^2}$ seja menor que 10^{-7}

```
S=0 ; n=1 ;  
while 1/n^2 >= 1e-7,  
    S= S+1/n^2 ;  
    n= n+1 ;  
end
```