#### **Pascal**

02 - Estruturas de Dados Homogêneas

Marcos Roberto Ribeiro



#### Introdução

- Os tipos básicos (*Integer*, *Real*, *Char*, e *Boolean*) não são suficientes para exprimir estruturas de dados complexas. Em determinadas situações é necessário utilizar estruturas de dados homogêneas;
- As estruturas de dados homogêneas permitem agrupar diversas informações dentro de uma mesma variável. Este agrupamento ocorre obedecendo sempre ao mesmo tipo de dado, e é por esta razão que estas estruturas são chamadas homogêneas. A grande vantagem das estruturas de dados homogêneas é a facilidade de acesso a posições indexadas dentro do agrupamento;
- As principais estruturas de dados homogêneas abordadas nesta disciplina serão<sup>1</sup>.
  - Vetores:
  - Matrizes;

<sup>&</sup>lt;sup>1</sup>Na verdade um **String** é um vetor de **Char** em Pascal, mas veremos isto posteriormente.

# Programa Média Simples (O que ele faz?)

```
program MediaSimples;
const
 NUM_ALUNOS = 10;
var
 Contador: Integer;
 Nota, Soma, Media: Real;
begin
 Soma := 0:
  for Contador := 1 to NUM_ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
   readln(Nota);
    Soma := Soma + Nota:
  end;
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é ', Media:0:2);
end.
```

#### O Problema: Notas acima da média

- Agora imagine que fosse necessário listar as notas acima da média no programa anterior. Como fazer isto de uma forma eficiente?
- O problema em questão é que usamos as notas dos alunos para calcular a média e depois precisamos novamente das notas de cada aluno, ou seja, precisamos guardar as notas dos alunos na leitura para utilizar futuramente;
- Uma das soluções para o problema seria declarar 10 variáveis uma para cada nota. Mas imagine uma turma de 50 alunos, ou se fosse preciso calcular a média de toda uma escola com milhares de alunos. A declaração de variáveis individuais seria praticamente impossível;
- Outra solução seria pedir para o usuário do programa informar as notas, realizar o cálculo da média e solicitar ao usuário que informe as notas novamente. Esta solução também não é eficiente, pois seria muito incômodo para o usuário, o qual pode digitar os dados errados na segunda vez;
- A melhor solução para o problema é a utilização de um vetor.

#### **Vetores**

- Um vetor (ou arranjo, ou ainda *array* em inglês) é um estrutura de dados homogênea unidimensional;
- Um vetor pode ser visto como uma lista de dados do mesmo tipo. Desta maneira, cada posição do vetor pode armazenar um valor diferente;
- Por exemplo, no vetor de inteiros representado abaixo, a posição 2 possui o valor 35, já a posição 6 possui o valor 52.

Posição:	1	2	3	4	5	6
Valor:	40	35	61	89	10	52

#### Vetores em Pascal

 Em Pascal, o primeiro passo é definir um tipo de vetor em type e depois declará-lo com o tipo definido em var. O ideal é utilizar uma constante para delimitar o tamanho do vetor.

```
const
  NUM_NOTAS = 10;
type
  TVetorNotas = array [1..NUM_NOTAS] of Integer;
var
  Notas: TVetorAlunos;
```

 Após a declaração do vetor, podemos acessar suas posições da seguinte maneira:

```
{Gravando 30 na posição 2 do vetor}
Notas[2] := 30;
{Lendo a posição 5 do vetor}
X := Notas[5];
```

### Programa com Vetor I

• O programa a seguir utiliza um vetor para resolver o problema das notas acima da média discutido anteriormente.

```
program NotasAcimaMedia;
const
  NUM_ALUNOS = 10;
type
  TVetorNotas = array [1..NUM_ALUNOS] of Real;
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real;
```

#### Programa com Vetor II

```
begin
 Soma := 0;
 for Contador := 1 to NUM_ALUNOS do begin
   writeln('Informe a nota do aluno ', Contador);
   readln(Nota[Contador]);
   Soma := Soma + Nota[Contador];
 end:
 Media := Soma / NUM_ALUNOS;
 writeln('A média da turma é ', Media:0:2);
 writeln('Os alunos com notas acima da média são:');
 for Contador := 1 to NUM_ALUNOS do begin
    if Nota[Contador] > Media then begin
      writeln(Contador, ':', Nota[Contador]:0:2);
   end;
 end;
end.
```

```
program NotasAcimaMedia;
 NUM ALUNOS = 10:
 TVetorNotas = array [1..NUM_ALUNOS] of Real;
var
 Contador: Integer;
 Nota: TVetorNotas;
 Soma, Media: Real;
end.
```

• (	Conta	dor	
• 5	Soma		
	Media		
		1	
		2	
	Nota	3	
		4	
		5	
	vota	6	
		7	
		8	
		9	
		10	

```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real;
begin
  Soma := 0;
end.
```

<ul><li>Conta</li></ul>	dor	
Soma	0	
Media	9	
	1	
	2	
	3	
	4	
Nota	5	
Nota	6	
	7	
	8	
	9	
	_10	

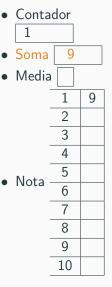
```
program NotasAcimaMedia;
const
 NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
 Contador: Integer;
 Nota: TVetorNotas;
 Soma, Media: Real:
begin
 Soma := 0;
  for Contador := 1 to NUM ALUNOS do begin
end.
```



```
program NotasAcimaMedia;
const.
 NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
 Contador: Integer;
 Nota: TVetorNotas;
 Soma, Media: Real:
begin
 Soma := 0;
 for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
   readln(Nota[Contador]);
end.
```



```
program NotasAcimaMedia;
const.
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0;
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```

Conta	laor	
2		
Soma	9	
Media		
	1	9
	2	
	3	
	4	
Nota	5	
NOLA	6	
	7	
	8	
	9	
	10	

```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0;
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```

)	Conta	Contador				
	2					
)	Soma	9				
,	Media	1				
	,	1	9			
		2	8			
		3				
	Nota	4				
,		5				
'	INOLA	6				
		7				
		8				
		9				
		10				

```
program NotasAcimaMedia;
const.
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```



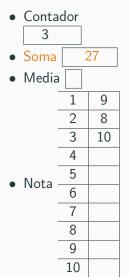
```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM_ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```

•	Conta	dor	
	3		
	Soma	1	7
	Media		
		1	9
		2	8
		3	
. No	•	4	
	Nota	5	
	INOLA	6	
		7	
		8	
		9	
		10	

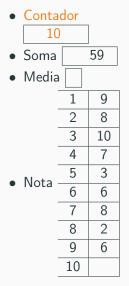
```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0;
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```

•	Conta	dor	
	3		
•	Soma	1	7
•	Media		
		1	9
	Nota -	2	8
•		3	10
		4	
		5	
		6	
		7	
		8	
		9	
		10	

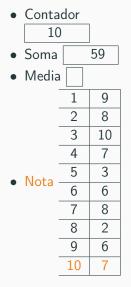
```
program NotasAcimaMedia;
const.
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```



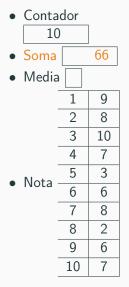
```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```



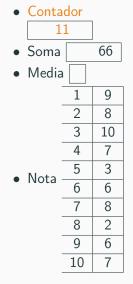
```
program NotasAcimaMedia;
const.
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0;
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```



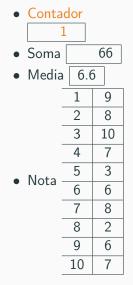
```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas;
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno ', Contador);
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é ', Media:0:2):
end.
```



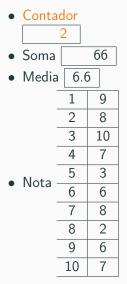
```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
  TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é '. Media:0:2);
  writeln():
  writeln('Os alunos com notas acima da média são:');
  for Contador := 1 to NUM_ALUNOS do begin
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é '. Media:0:2);
  writeln():
  writeln('Os alunos com notas acima da média são:');
  for Contador := 1 to NUM_ALUNOS do begin
    if Nota[Contador] > Media then begin
      writeln(Contador, ' : ', Nota[Contador]:0:2);
    end:
end.
```



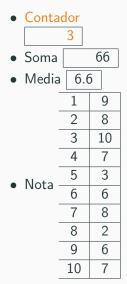
```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é '. Media:0:2);
  writeln():
  writeln('Os alunos com notas acima da média são:');
  for Contador := 1 to NUM_ALUNOS do begin
    if Nota[Contador] > Media then begin
      writeln(Contador, ' : ', Nota[Contador]:0:2);
    end:
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é '. Media:0:2);
  writeln():
  writeln('Os alunos com notas acima da média são:');
  for Contador := 1 to NUM_ALUNOS do begin
    if Nota[Contador] > Media then begin
      writeln(Contador, ' : ', Nota[Contador]:0:2);
    end:
  end:
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é '. Media:0:2);
  writeln():
  writeln('Os alunos com notas acima da média são:');
  for Contador := 1 to NUM_ALUNOS do begin
    if Nota[Contador] > Media then begin
      writeln(Contador, ' : ', Nota[Contador]:0:2);
    end:
end.
```



```
program NotasAcimaMedia;
const
  NUM ALUNOS = 10:
type
 TVetorNotas = array [1..NUM ALUNOS] of Real:
var
  Contador: Integer;
  Nota: TVetorNotas:
  Soma, Media: Real:
begin
  Soma := 0:
  for Contador := 1 to NUM ALUNOS do begin
    writeln('Informe a nota do aluno '. Contador):
    readln(Nota[Contador]);
    Soma := Soma + Nota[Contador]:
  end:
  Media := Soma / NUM_ALUNOS;
  writeln('A média da turma é '. Media:0:2);
  writeln():
  writeln('Os alunos com notas acima da média são:');
  for Contador := 1 to NUM_ALUNOS do begin
    if Nota[Contador] > Media then begin
      writeln(Contador, ' : ', Nota[Contador]:0:2);
    end:
  end:
end.
```



#### Exercícios

- 1. Modifique o programa anterior para contar o número de alunos com nota acima da média;
- 2. Simule a execução do algoritmo modificado.

## Programa Chamada I

 Observe o próximo programa utilizado para realizar a chamada de uma turma de alunos.

```
program Chamada1;

const
   NUM_ALUNOS = 10;

type
   TVetorPresenca = array [1..NUM_ALUNOS] of Integer;

var
   Contador, NumFaltas: Integer;
   Presenca: TVetorPresenca;
```

## Programa Chamada II

```
begin
 for Contador := 1 to NUM_ALUNOS do begin
    writeln('Aluno ', Contador, ', informe 0 (presente) ou 1 (faltou)');
   readln(Presenca[Contador]);
 end;
 writeln('Os alunos faltantes foram');
 NumFaltas := 0;
 for Contador := 1 to NUM_ALUNOS do begin
    if (Presenca[Contador] <> 0) then begin
      writeln(Contador):
      NumFaltas := NumFaltas + 1:
   end:
 end;
 writeln('Total de faltas: ', NumFaltas)
end.
```

## Problema: Chamada e verificação

- O programa anterior realiza a chamada de uma aula. Suponha agora que fosse necessário realizar a chamada de 20 aulas e, em seguida, o usuário pudesse verificar quais alunos faltaram em cada aula;
- Uma solução seria criar 20 vetores, um para cada aula, mas se o número de aulas aumentasse para 50?
- Neste caso seria interessante a utilização de matrizes.

#### **Matrizes**

- Uma matriz é uma estrutura de dados semelhante a um vetor, com a diferença de que uma matriz é multi-dimensional;
- As matrizes mais simples são as matrizes bi-dimensionais que podem ser vistas como tabelas e seus elementos podem ser referenciados através de uma linha e uma coluna;
- Por exemplo, na matriz 3x4 (3 linhas por 4 colunas) representada abaixo, a posição 2,1 (linha 2, coluna 1) possui o elemento 30.

		Colunas			
		1	2	3	4
	1	61	89	10	52
Linhas	2	30	23	17	28
	3	65	72	81	58

#### Matrizes em Pascal

• Declaração de uma matriz 3x4 em Pascal:

```
const
  NUM_LINHAS = 3;
  NUM_COLUNAS = 4;
type
  TMatriz = array [1..NUM_LINHAS, 1..NUM_COLUNAS] of Integer;
var
  Matriz: TMatriz;
```

• Acessando posições de uma matriz em Pascal:

```
{Gravando na linha 2 e coluna 1 da matriz}

Matriz[2,1] := 30;
{Lendo a linha 3 e coluna 4 da matriz}

X := Matriz[3,4];
```

# Programa de Chamada2 (Solução com matriz) I

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM AULAS = 5:
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer;
 Presenca: TMatrizPresenca;
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
   for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno ', ContAluno, ', informe 0 (presente) ou 1
readln(Presenca[ContAula, ContAluno]);
    end:
```

# Programa de Chamada2 (Solução com matriz) II

```
end:
 writeln('Informe a aula para listagem de faltas');
 readln(ContAula);
 writeln('Os alunos faltantes nesta aula foram');
 NumFaltas := 0;
 for ContAluno := 1 to NUM_ALUNOS do begin
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno);
      NumFaltas := NumFaltas + 1;
   end:
 end;
 writeln('Total de faltas: ', NumFaltas);
end.
```

```
program Chamada2;
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
end.
```

9 10

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
  end:
end.
```

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
    end:
  end:
end.
```

•	ContAluno 1							
•	Conf	tAula	1					
•	Num	Falta	as					
•	Pres 1	enca 2	3	4	5			
1								
2								
4								
5								
6								
7								
8								
9								
10								

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno ', ContAluno, ', informe O (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  for ContAluno := 1 to NUM ALUNOS do begin:
end.
```

•	ContAluno ContAula 1							
•	Num	Falta	as					
•	Pres 1	enca 2	3	4	5			
1	0							
2								
3								
4								
5								
6								
7								
8								
9								
10								

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

ContAluno

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno ', ContAluno, ', informe O (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

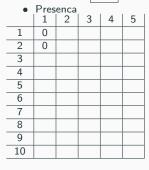
ContAluno

1

4

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

- ContAluno
  3
- ContAula 1
- NumFaltas



```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno ', ContAluno, ', informe O (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

3
• ContAula 1
• NumFaltas Presenca 1 2 3 4 5

ContAluno

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  for ContAluno := 1 to NUM ALUNOS do begin:
end.
```

- ContAluno
  10
- ContAula 1
- NumFaltas

•	Presenca						
	1	2	3	4	5		
1	0						
2	0						
	1						
4 5	0						
5	0						
6	0						
7	0						
8	1						
9	0						
10							

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno ', ContAluno, ', informe O (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

- ContAluno
- ContAula 1
- NumFaltas



```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  for ContAluno := 1 to NUM ALUNOS do begin:
end.
```

- ContAluno
- ContAula 1
- NumFaltas

<ul> <li>Presenca</li> </ul>							
	1	2	3	4	5		
1	0						
2	0						
3	1						
4	0						
5	0						
6	0						
7	0						
8	1						
9	0						
10	0						

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

ContAluno

1

3

4 0

6 0 7 0

8 1

9 0

10 | 0

0

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM_AULAS, 1..NUM_ALUNOS] of Integer;
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```

		1							
•	Cont	ContAula 2							
•	Num	NumFaltas							
•	Pres	enca							
	1	2	3	4	5				
	0								
	0								

4 0

6 0

8 1

9 0

10 | 0

0

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno ', ContAluno, ', informe O (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
end.
```



```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM AULAS = 5:
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  for ContAluno := 1 to NUM_ALUNOS do begin;
end.
```



```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas'):
  readln(ContAula):
end.
```



```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
end.
```

<ul> <li>ContAluno</li> </ul>										
11										
•	• ContAula 2									
•	Num	Falt	as (	0						
•	Pres	enca								
	1	2	3	4	5					
1	0	0	0	0	1					
2	0	1	0	0	0					
3	1	1	0	0	0					
4	0	0	0	0	0					
5	0	0	0	0	0					
6	0	0	0	0	0					
7	0	0	0	0	0					
8	1	1	0	0	0					
9	0	0	0	0	0					
10	0	0	0	0	0					

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0;
  for ContAluno := 1 to NUM ALUNOS do begin:
  end;
```

end.

•	ContAluno						
	1						
•	Cont	Aula		2			
•	Num	Falta	as (	0			
•	Pres	enca	_				
	1	2	3	4	5		
1	0	0	0	0	1		
2	0	1	0	0	0		
3	1	1	0	0	0		
4	0	0	0	0	0		
5	0	0	0	0	0		
6	0	0	0	0	0		
7	0	0	0	0	0		
8	1	1	0	0	0		
9	0	0	0	0	0		
10	0	0	0	0	0		

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
    end:
  end:
end.
```

•	<ul><li>ContAluno</li></ul>								
	1								
•	ContAula 2								
•	Num	Falta	as (	0					
•	Pres	enca							
	1	2	3	4	5				
1	0	0	0	0	1				
2	0	1	0	0	0				
3	1	1	0	0	0				
4	0	0	0	0	0				
5	0	0	0	0	0				
6	0	0	0	0	0				
7	0	0	0	0	0				
8	1	1	0	0	0				
9	0	0	0	0	0				
10	0	0	0	0	0				

```
program Chamada2;
const
  NUM ALUNOS = 10:
  NUM AULAS = 5:
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram'):
  NumFaltas := 0:
  for ContAluno := 1 to NUM_ALUNOS do begin;
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno):
      NumFaltas := NumFaltas + 1:
    end:
  end;
end.
```

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno);
      NumFaltas := NumFaltas + 1:
    end:
  end:
```

end.

•	ContAluno								
	2								
•	• ContAula 2								
•	Num	Falta	as (	0	]				
•	Pres	enca	_						
	1	2	3	4	5				
1	0	0	0	0	1				
2	0	1	0	0	0				
3	1	1	0	0	0				
4	0	0	0	0	0				
5	0	0	0	0	0				
6	0	0	0	0	0				
7	0	0	0	0	0				
8	1	1	0	0	0				
9	0	0	0	0	0				
10	0	0	0	0	0				

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno);
      NumFaltas := NumFaltas + 1:
    end:
  end:
```

end.

•	ContAluno					
•	Con	tAula		2		
•	Num	Falt	as	1		
•	Pres	enca	. –			
	1	2	3	4	5	
1	0	0	0	0	1	
2	0	1	0	0	0	
3	1	1	0	0	0	
4	0	0	0	0	0	
5	0	0	0	0	0	
6	0	0	0	0	0	
7	0	0	0	0	0	
8	1	1	0	0	0	
9	0	0	0	0	0	
10	0	0	0	0	0	

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM AULAS = 5:
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno):
      NumFaltas := NumFaltas + 1:
    end:
  end;
end.
```

•	ContAluno							
		3						
•	Cont	tAula		2				
•	Num	Falta	as	1	]			
•	Pres	enca	_					
	1	2	3	4	5			
1	0	0	0	0	1			
2	0	1	0	0	0			
3	1	1	0	0	0			
4	0	0	0	0	0			
5	0	0	0	0	0			
6	0	0	0	0	0			
7	0	0	0	0	0			
8	1	1	0	0	0			
9	0	0	0	0	0			
10	0	0	0	0	0			

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]);
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno);
      NumFaltas := NumFaltas + 1:
    end:
  end:
end.
```

•	ContAluno 3									
•	Conf	tAula	1	2						
•	Num	Falta	as	1	]					
•	Pres	enca			-					
	1	2	3	4	5					
1	0	0	0	0	1					
2	0	1	0	0	0					
3	1	1	0	0	0					
4	0	0	0	0	0					
5	0	0	0	0	0					
6	0	0	0	0	0					
7	0	0	0	0	0					
8	1	1	0	0	0					
9	0	0	0	0	0					
Λ	Λ	Λ	Λ	Λ	^					

```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno);
      NumFaltas := NumFaltas + 1:
    end:
  end:
```

end.

•	ContAluno						
•	Con						
•	Num	Falt	as	2			
•	Presenca						
	1	2	3	4	5		
1	0	0	0	0	1		
2	0	1	0	0	0		
3	1	1	0	0	0		
4	0	0	0	0	0		
5	0	0	0	0	0		
6	0	0	0	0	0		
7	0	0	0	0	0		
8	1	1	0	0	0		
9	0	0	0	0	0		
10	0	0	0	0	0		

```
program Chamada2;
const
  NUM ALUNOS = 10:
  NUM AULAS = 5:
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram'):
  NumFaltas := 0:
  for ContAluno := 1 to NUM_ALUNOS do begin;
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno):
      NumFaltas := NumFaltas + 1:
    end:
  end;
end.
```



```
program Chamada2;
const
  NUM_ALUNOS = 10;
  NUM_AULAS = 5;
type
  TMatrizPresenca = array [1..NUM AULAS, 1..NUM ALUNOS] of Integer:
var
  ContAula, ContAluno, NumFaltas: Integer:
  Presenca: TMatrizPresenca:
begin
  for ContAula:=1 to NUM_AULAS do begin
    writeln('Informe as faltas para a aula ', ContAula);
    for ContAluno := 1 to NUM_ALUNOS do begin
      writeln('Aluno', ContAluno,', informe 0 (presente) ou 1 (faltou)');
      readln(Presenca[ContAula, ContAluno]):
    end:
  end:
  writeln('Informe a aula para listagem de faltas');
  readln(ContAula):
  writeln('Os alunos faltantes nesta aula foram');
  NumFaltas := 0:
  for ContAluno := 1 to NUM ALUNOS do begin:
    if (Presenca[ContAula,ContAluno] <> 0) then begin
      writeln(ContAluno):
      NumFaltas := NumFaltas + 1:
    end:
  end:
  writeln('Total de faltas: '. NumFaltas):
end.
```

<ul> <li>ContAluno</li> </ul>								
	11							
•	ContAula 2							
•	Num	Falt	as	3				
Presenca								
	1	2	3	4	5			
1	0	0	0	0	1			
2	0	1	0	0	0			
3	1	1	0	0	0			
4	0	0	0	0	0			
5	0	0	0	0	0			
6	0	0	0	0	0			
7	0	0	0	0	0			
8	1	1	0	0	0			
9	0	0	0	0	0			
10	0	0	0	0	0			

#### Exercícios

Crie outras versões do programa anterior com as seguintes alterações:

- Após o preenchimento da chamada, o usuário deve informar um aluno. O programa deve mostrar as aulas que o aluno faltou e o total de faltas do aluno;
- Após o preenchimento da chamada, o programa deve exibir o número de alunos que faltou em cada aula. Dica: utilize mais um vetor;
- 3. Após o preenchimento da chamada, o programa deve listar o total de faltas de cada aluno. Dica: utilize mais um vetor.

#### Referências I



Canneyt, M. V.

Free pascal reference guide.

http://www.freepascal.org/docs-html/ref/ref.html.



Evaristo, J. (1999). Programando com Pascal.

Book Express, Rio de Janeiro, 2 edition.



Farrer, H., Becker, C. G., Faria, E. C., de Matos, H. F., dos Santos, M. A., and Maia, M. L. (2008).

Programação estruturada de computadores.

LTC, Rio de Janeiro, 3 edition.

#### Referências II

Forbellone, A. V. and Eberspacher, H. F. (2005). *Lógica de programação*.

Prentice Hall, São Paulo, 3 edition.

Manzano, J. A. N. G. and de Oliveira, J. F. (2011). *Algoritmos*.

Érica, São Paulo, 25 edition.

Manzano, J. A. N. G. and Yamatumi, W. Y. (2001). *Programando em turbo pascal 7.0 e free pascal compiler*. Érica, São Paulo, 9 edition.