

ESTRUTURAS COMPOSTAS

Variáveis Compostas Unidimensionais VETOR

Profa. Dra. Elisa Yumi Nakagawa
1. Semestre de 2017

ESTRUTURAS COMPOSTAS

- Pode-se organizar tipos simples em tipos mais complexos formando as ESTRUTURAS COMPOSTAS
- Exemplo:
 - variáveis compostas unidimensionais (VETOR)

VETOR

- Conceito de VETOR será introduzido através de um exemplo.
- Suponhamos o seguinte problema:

Calcular a média aritmética das notas de 3 alunos. Exibir as notas que estão abaixo da média

1ª Solução (RUIM!!)

- Ler as três notas e armazenar em três variáveis distintas
- Calcular a média
- Mostrar a média
- Comparar cada nota com a média calculada e mostrar as notas que são menores que a média.

programa RUIM

declarações omitidas...

inicio

ler (NOTA1,NOTA2,NOTA3)

MEDIA = (NOTA1 + NOTA2 + NOTA3)/3

escrever (MEDIA)

se NOTA1 < MEDIA

então escrever (NOTA1,"abaixo da média")

fim-se

se NOTA2 < MEDIA

então escrever (NOTA2,"abaixo da média")

fim-se

se NOTA3 < MEDIA

então escrever (NOTA3,"abaixo da média")

fim-se

fim

programa RUIM
declarações omitidas...
inicio

```
ler (NOTA1,NOTA2,NOTA3)
MEDIA = (NOTA1 + NOTA2 + NOTA3)/3
escrever (MEDIA)
se NOTA1 < MEDIA
    então escrever (NOTA1,"abaixo da média")
fim-se
se NOTA2 < MEDIA
    então escrever (NOTA2,"abaixo da média")
fim-se
se NOTA3 < MEDIA
    então escrever (NOTA3,"abaixo da média")
fim-se
```

fim

O programa só
funciona para três notas

Qual seria o algoritmo para uma
relação de **1000** notas?

Associarmos um variável para cada
nota?

IMPRATICÁVEL!

Solução (MAIS REALISTA)

- Associar a variável NOTA ao CONJUNTO
ORDENADO de notas

$$\text{NOTA} = \{ \underset{1^{\text{a}}}{\text{N1}}, \underset{2^{\text{a}}}{\text{N2}}, \dots, \underset{1000^{\text{a}}}{\text{N1000}} \}$$

Calcular a média aritmética das notas de 1000 alunos.
Exibir a média e as notas que estão abaixo da média

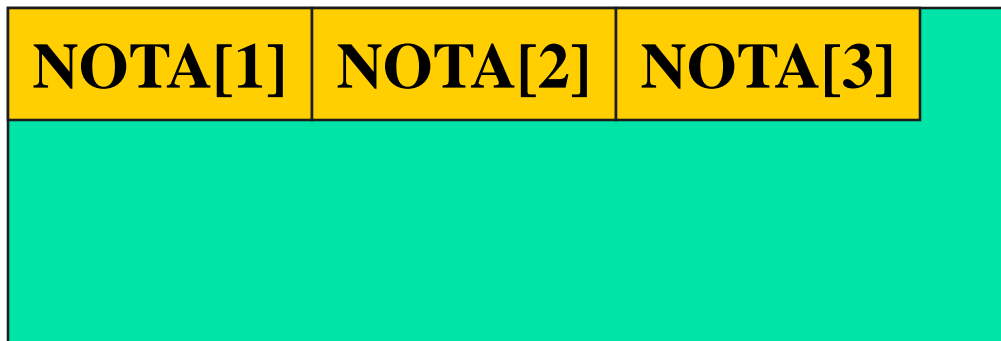
2ª Solução (MAIS REALISTA)

NOTA = {N1, N2, ... N1000}
 1ª 2ª 1000ª

- para fazer referência ou selecionar uma nota específica usar um índice
- Exemplo:
 - a 3ª nota é indicada por NOTA[3]
 - a 1000ª nota é indicada por NOTA[1000]
 - uma kª nota é indicada por NOTA[k]

VARIÁVEL INDEXADA

- Cada **variável indexada** é associada a uma posição de memória, como acontece com variáveis simples.
- Exemplo:

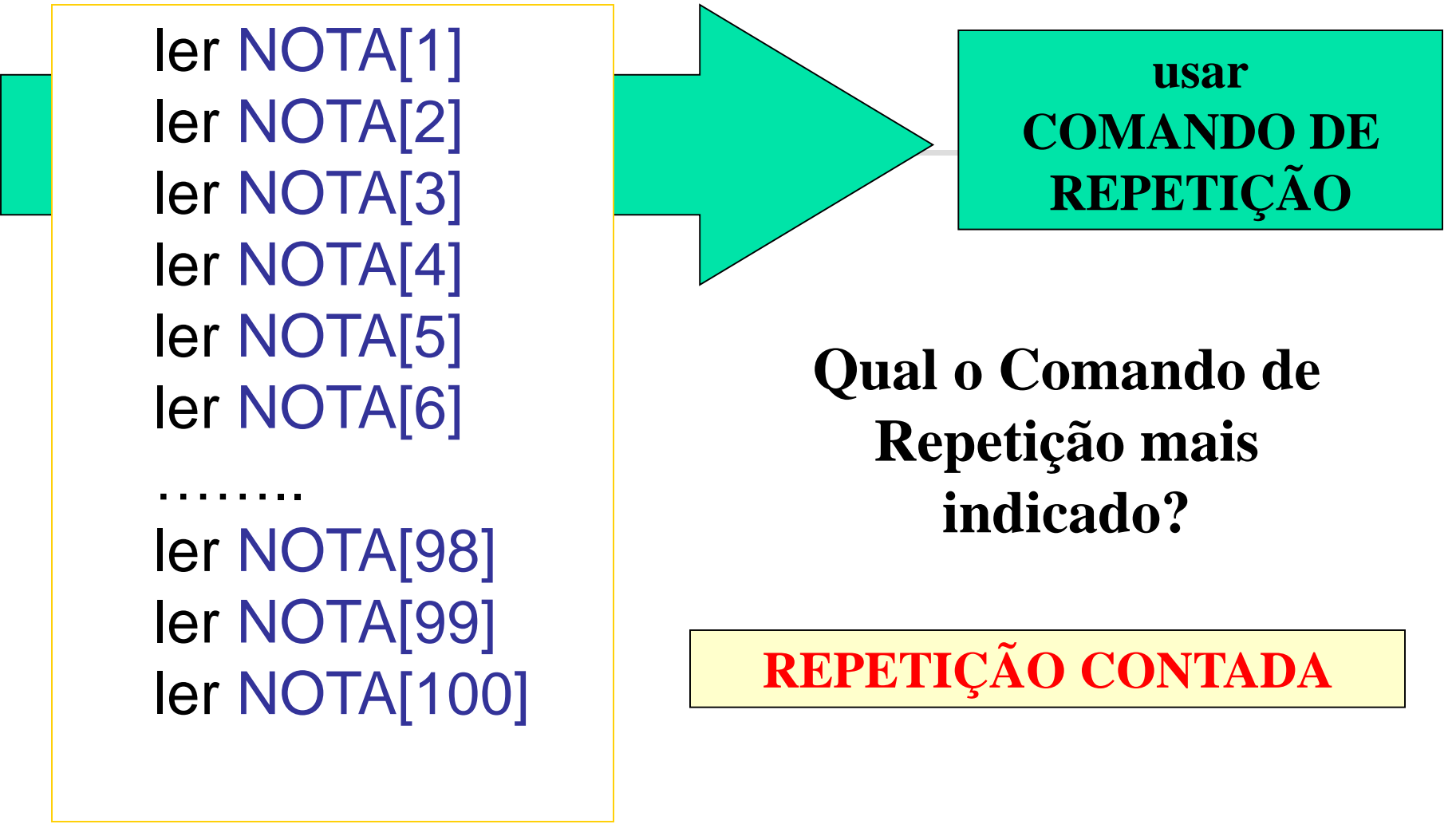


VETOR

- Um **VETOR** é um conjunto ordenado que contém um número fixo de elementos
- Todos os elementos do vetor devem ser do mesmo tipo

VETOR - Exemplo 1

- Ler um conjunto de 100 notas, armazená-las no vetor denominado NOTA e escrever este vetor.



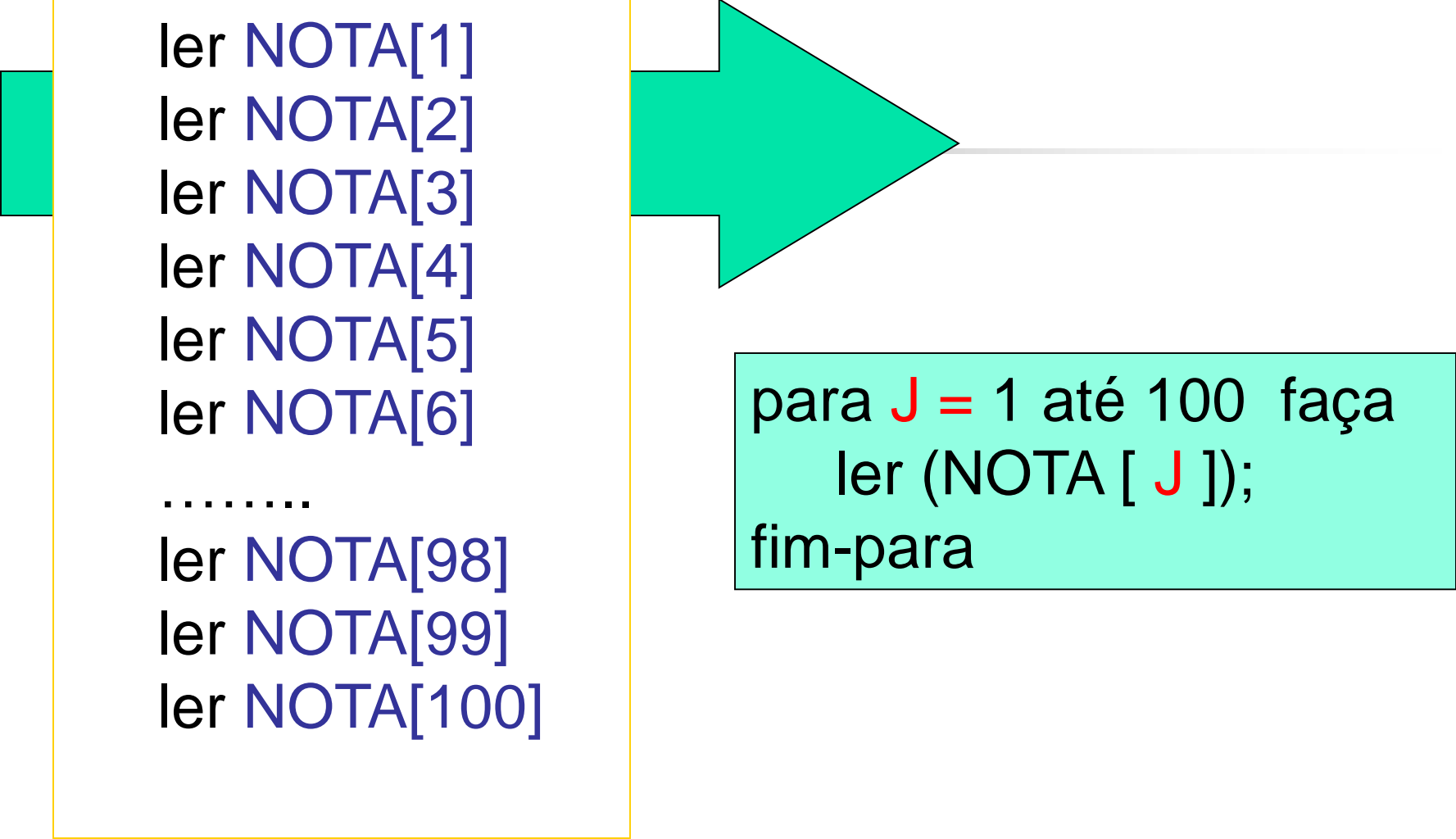
```
ler NOTA[1]
ler NOTA[2]
ler NOTA[3]
ler NOTA[4]
ler NOTA[5]
ler NOTA[6]

.....
ler NOTA[98]
ler NOTA[99]
ler NOTA[100]
```

**usar
COMANDO DE
REPETIÇÃO**

**Qual o Comando de
Repetição mais
indicado?**

REPETIÇÃO CONTADA



```
ler NOTA[1]
ler NOTA[2]
ler NOTA[3]
ler NOTA[4]
ler NOTA[5]
ler NOTA[6]
.....
ler NOTA[98]
ler NOTA[99]
ler NOTA[100]
```

```
para J = 1 até 100 faça
    ler (NOTA [ J ]);
fim-para
```

ler NOTA[1]
ler NOTA[2]
ler NOTA[3]
ler NOTA[4]
ler NOTA[5]
ler NOTA[6]

.....
ler NOTA[98]
ler NOTA[99]
ler NOTA[100]

para **J** = 1 até 100 faça
 ler (NOTA [**J**])
fim-para

mesmo efeito que

para **I** = 1 até 100 faça
 ler (NOTA [**I**])
fim-para

mesmo efeito que

para **K** = 1 até 100 faça
 ler (NOTA [**K**])
fim-para

Algoritmo VET1

...declarações omitidas...

inicio

para **J**=1 até 100
 faça ler (NOTA[**J**])
fim para

para J = 1 até 100
 faça escrever (NOTA[J])
fim para

fim

Leitura das notas

Algoritmo VET1

...declarações omitidas...

inicio

para J = 1 até 100

faça ler (NOTA[J])

fim para

para J = 1 até 100

faça escrever (NOTA[J])

fim para

fim

Escrita das notas

Algoritmo VET1

...declarações om
início

para J = 1 até 100

faça ler (NOTA[J])

fim para

para J = 1 até 100

faça escrever (NOTA[J])

fim para

fim

Importante!

Não usar o mesmo laço!

Isto diminui a legibilidade

Algoritmo VET1

...declarações omitidas...

inicio

para J=1 até 100

faça ler (NOTA[J])

fim para

para J=1 até 100

faça escrever(NOTA[J])

fim para

fim

VETOR - Exemplo 2

- Ler um conjunto de 100 notas, armazená-las no vetor denominado NOTA, calcular a soma dessas notas, escrever o vetor de notas e a soma das notas.

Algoritmo VET2

VETOR - Exemplo 2

...declarações omitidas...

início

```
para I = 1 até 100  
  faça ler (NOTA[I])  
fim para
```

Leitura das notas

SOMA = 0

```
para I = 1 até 100  
  faça SOMA = SOMA + NOTA[I]  
fim para
```

Soma das notas

```
para I = 1 até 100  
  faça escrever (NOTA[I])  
fim para  
escrever (SOMA)
```

Escrita das notas

fim

programa VET2

...declarações omitidas...

inicio

para I = 1 até 100

faça ler (NOTA[I])

fim para

SOMA = 0

para I = 1 até 100

faça SOMA = SOMA + NOTA[I]

fim para

para I = 1 até 100

faça escrever (NOTA[I])

fim para

escrever (SOMA)

fim

VETOR - Exemplo 3

- Ler um conjunto de 100 notas armazenando no vetor denominado NOTA.
- Verificar se existe nota 10.0. Se existir, dizer quantas existem.

Algoritmo VET3

...declarações omitidas...

inicio

```
para I = 1 até 100  
  faça ler (NOTA[I])  
fim para
```

Leitura das notas

```
CONTADOR = 0
```

Inicialização
do contador

```
para I = 1 até 100
```

```
  faça se (NOTA[I] == 10.0)
```

Verificação se
a nota é 10.0

```
    então CONTADOR = CONTADOR + 1
```

```
  fim-se
```

```
fim para
```

```
escrever (CONTADOR)
```

fim

Aumento do contador
quando a nota é 10.0

VETOR - Exemplo 3

programa VET3

...declarações omitidas...

inicio

para I = 1 até 100

faça ler (NOTA[I])

fim para

CONTADOR = 0

para I = 1 até 100

faça se (NOTA[I] == 10.0)

então CONTADOR = CONTADOR + 1

fim-se


fim para

escrever (CONTADOR)

fim

VETOR - Exemplo 4

- Ler um conjunto de 100 notas armazenando no vetor denominado NOTA.
- Calcular a média, verificar e exibir as notas abaixo da média.



Mesmo exemplo
do início

Algoritmo VETOR13a

definir constante **N=100**

...declarações omitidas...

inicio

para **I** \equiv 1 até **N**

faça ler (NOTA[**I**])

fim para

SOMA = 0

para **I** \equiv 1 até **N**

faça SOMA = SOMA + NOTA[**I**]

fim para

MEDIA = SOMA/**N**

para **I** \equiv 1 até **N**

faça se NOTA[**I**] < MEDIA

então escrever (NOTA[**I**])

fim-se

fim para

fim

Leitura das notas

Cálculo da Média

Escrita das notas
abaixo da média

Exemplo 4

Algoritmo VETOR13a

definir constante **N=100**

...declarações omitidas...

inicio

para **I** \equiv 1 até **N**

faça ler (NOTA[**I**])

fim para

SOMA = 0

para **I** \equiv 1 até **N**

faça SOMA = SOMA + NOTA[**I**]

fim para

MEDIA = SOMA/**N**

para **I** \equiv 1 até **N**

faça se NOTA[**I**] < MEDIA

então escrever (NOTA[**I**])

fim-se

fim para

fim

VETOR - Exemplo 4

Podem ser
passados para
C
diretamente
Já são
instruções
conhecidas.

Algoritmo VETOR13a

VETOR - Exemplo 4

definir constante **N=100**

...declarações omitidas...

INICIO

para **I** \equiv 1 até **N**

faça ler (NOTA[**I**])

fim para

SOMA = 0

para **I** \equiv 1 até **N**

faça SOMA = SOMA + NOTA[**I**]

fim para

MEDIA = SOMA/**N**

para **I** \equiv 1 até **N**

faça se NOTA[**I**] < MEDIA

então escrever (NOTA[**I**])

fim-se

fim para

fim

Como são
declaradas
as variáveis
indexadas
unidimensionais?

Declaração (C) de Variável Indexada Unidimensional

- Deve ser especificado o número máximo de elementos do conjunto
- Deve ser especificado o tipo dos elementos do conjunto
- Exemplo:

```
float x[100];
```

Declaração (C) de Variável Indexada Unidimensional

Tipo dos elementos do conjunto

Nome da Variável

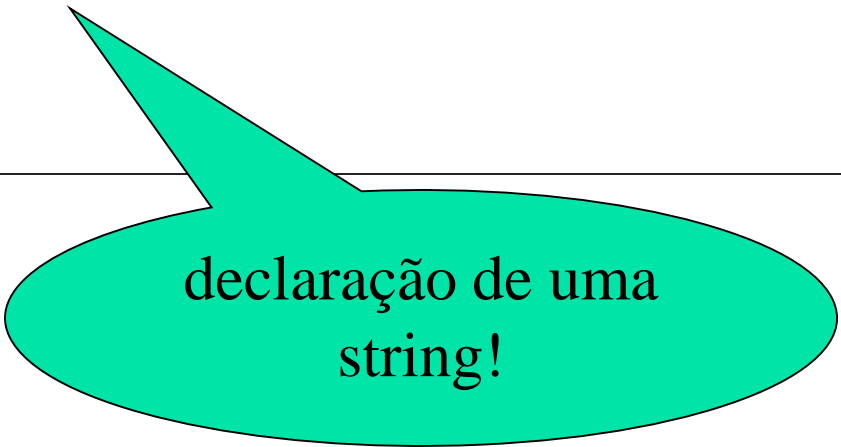
Número máximo de elementos do conjunto

```
float x[100];
```

Declaração (C) de Variável Indexada Unidimensional - Exemplos

- Declaração de um vetor S com no máximo 100 elementos do tipo character

```
char S[100];
```



declaração de uma
string!

Declaração (C) de Variável Indexada Unidimensional

- Pode ser definida uma **constante** e esta ser utilizada no dimensionamento.
- Exemplo:

```
//Início do programa C  
const int MAX = 5; OU  
#define MAX 5  
//declaração:  
float X[MAX];
```



```
#include <stdio.h>
#include <stdlib.h>
# define MAX 10
```

```
int main()
```

```
{
```

```
    float nota[MAX], media, soma = 0;
```

```
    int i;
```

```
    printf("Entre com as notas dos alunos:");
```

```
    for(i=0; i<MAX; i++) //leitura das notas
```

```
        scanf("%f", &nota[i]);
```



Leitura das notas

```
for(i=0; i<MAX; i++)  
    soma = soma + nota[i];  
media = soma/MAX;
```

Cálculo da média

```
printf("média da turma = %.1f\n", media);
```

Exibe a média

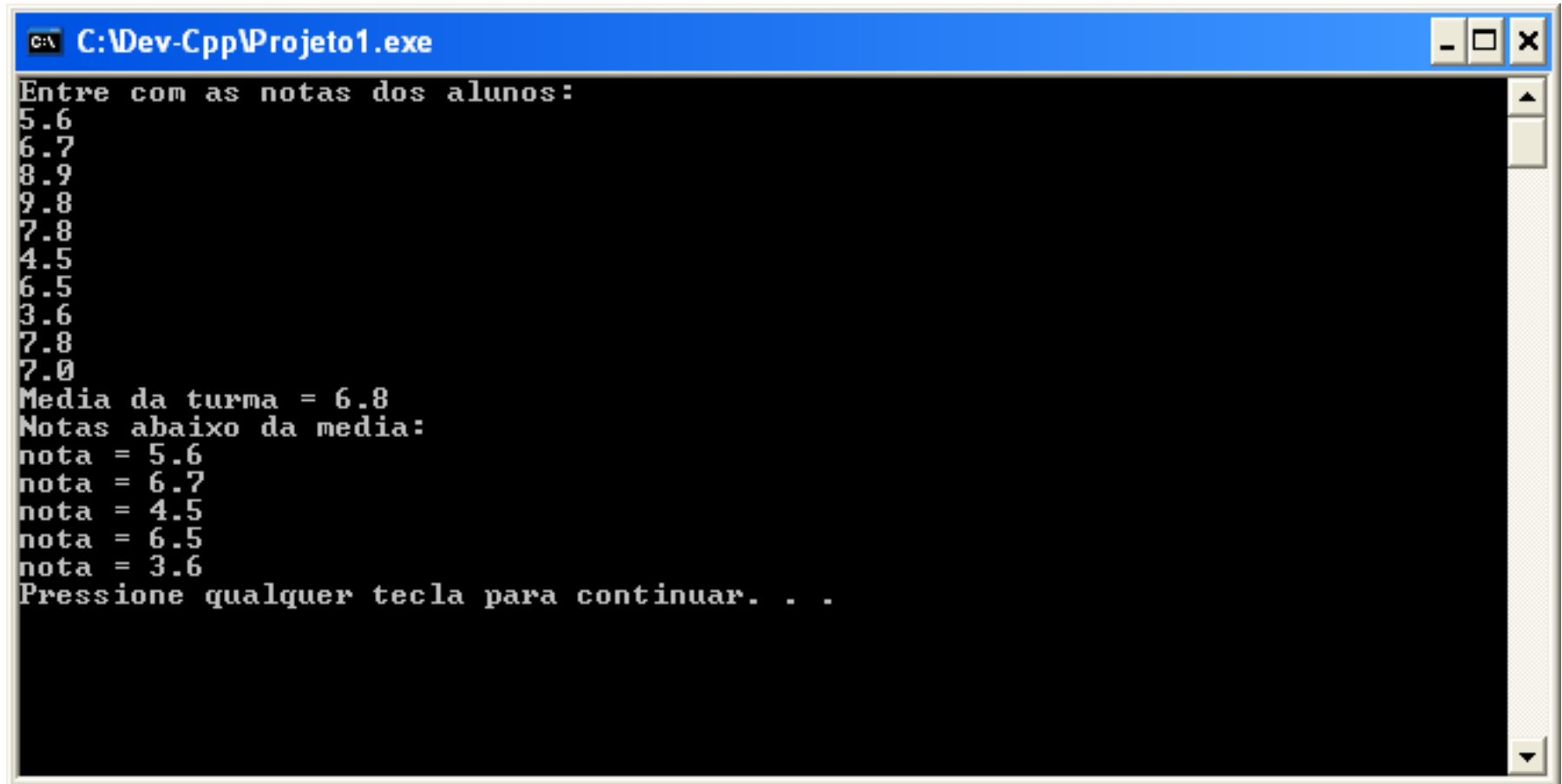
```
printf("Notas abaixo da media:\n");  
for(i=0; i<MAX; i++)  
    if (nota[i] < media)  
        printf("nota = %.1f\n", nota[i]);
```

Escrita de todas
as notas

```
system("PAUSE");
```

```
return 0;
```

```
}
```



```
C:\Dev-Cpp\Projeto1.exe
Entre com as notas dos alunos:
5.6
6.7
8.9
9.8
7.8
4.5
6.5
3.6
7.8
7.0
Media da turma = 6.8
Notas abaixo da media:
nota = 5.6
nota = 6.7
nota = 4.5
nota = 6.5
nota = 3.6
Pressione qualquer tecla para continuar. . .
```

Exercício 1: em classe

Usando algoritmos:

- Ler um conjunto de **N** números inteiros (N é lido e é menor que 100).
- Encontrar e exibir o maior deles.

Algoritmo

programa MAIOR
declarações
inicio

ler (N)

Lê a quantidade
de números

para J = 1 até N

Lê os números

faça ler (NRO[J])

fim para

MAIOR = NRO[1]

Assume que o
primeiro elemento do conjunto
é o maior de todos

para I = 2 até N

faça se NRO[I] > MAIOR

| então MAIOR = NRO[I]

| fim-se

fim para

Compara o
elemento assumido
como maior com todos
os outros, colocando em
MAIOR aquele elemento
que realmente for o
maior de todos

escrever (MAIOR)

fim-programa

programa MAIOR

declarações

inicio

ler (**N**)

para **J** = 1 até **N**

faça ler (NRO[**J**])

fim para

 MAIOR = NRO[1]

 para **I** = 2 até **N**

faça se NRO[**I**] > MAIOR

 | então MAIOR = NRO[**I**]

fim-se

fim para

 escrever (MAIOR)

fim-programa

Exercício 2: em classe

- Elaborar um código em linguagem C que lê um conjunto de 30 valores inteiros e os coloca em um vetor. Calcular e mostrar:
 - Os números pares;
 - A quantidade de números pares;
 - Os números ímpares
 - A quantidade de números ímpares;

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
const int MAX = 30;
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    int num[MAX], i, qpar=0, qimpar=0;
```

```
    printf("Entre com os numeros:");
```

```
    for(i=0; i<MAX; i++) //leitura das notas
```

```
        scanf("%d", &num[i]);
```

Leitura dos números

Cálculo dos
números pares

```
printf(" Os numeros pares são: \n");  
for(i=0; i<MAX; i++)  
if (num[i] % 2 == 0) // eh par  
{  
    qpar++;  
    printf("%d\n", num[i]);  
}  
printf("O total de numeros pares eh: %d\n", qpar);
```

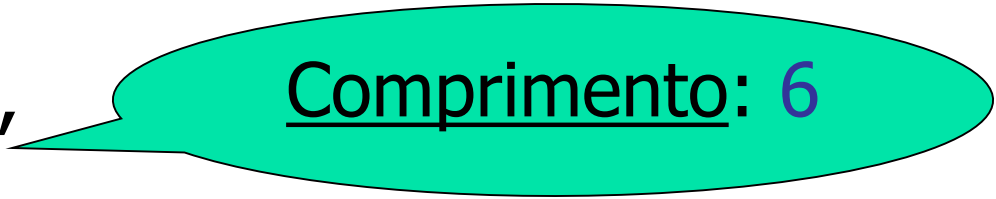
```
printf(" Os numeros impares são: \n");  
for(i=0; i<MAX; i++)  
if (num[i] % 2 != 0) // eh impar  
{  
    qimpar++;  
    printf("%d\n", num[i]);  
}  
printf("O total de numeros impares eh: %d\n", qimpar);  
  
system("PAUSE");  
return 0;  
}
```

Cálculo dos
números ímpares

Cadeia de Caracteres

- **CHARACTER:** letras, dígitos e símbolos
 - Exemplo: 'a', '%', '2'
- **CADEIA DE CARACTERES:** um conjunto de caracteres
 - Exemplo: "A B3*g", "1234"

Cadeia de Caracteres

- **COMPRIMENTO DA CADEIA:** número de caracteres que formam a cadeia
- Exemplo: "A B3*g"  Comprimento: 6

Declaração

- Cadeia de caracteres ou strings são vetores:
char nome[20], alunos[40][20];
char B;

Manipulação

- É possível acessar uma posição da string:

```
char nome[20] = "JOAO";
```

```
printf("%c", nome[0]);
```

```
printf("%s", nome);
```

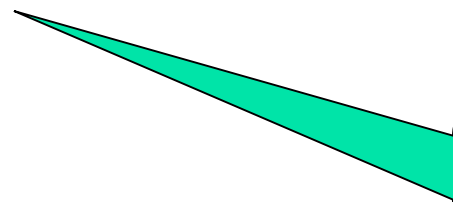
Manipulação

- Leitura de uma string:

```
char nome[20];
```

```
gets(nome);
```

```
printf("%s", nome);
```



Lê uma string e acrescenta a marca de fim de string \0

Manipulação

■ Atribuição de uma string:

```
char nome[20];
```

```
strcpy(nome, "JOAO");
```

```
printf("%s", nome);
```

A ATRIBUIÇÃO DE STRINGS
EM C USA A FUNÇÃO
STRCPY(). SIMILAR A:
Nome = "JOAO"

inserir biblioteca:
string.h

Expressões com Strings

- Os operadores relacionais podem ser usados com operandos do tipo caracter
- Operadores Relacionais

Para efeito de comparação entre os caracteres, toma-se como base a sequência comparativa do código

ASCII

==

<

>

!=

<=

>=

Expressões com Strings

a) Entre as letras, vale a ordem alfabética

`'C' < 'D '`
`"ABACATE" < "ABACAXI"`

b) Para os dígitos, vale a ordem numérica

`'1' < '3'`

Expressões com Strings

c) O branco sempre é menor que qualquer letra ou dígito

d) Os dígitos são menores que as letras

`'9' < 'a'`

e) Letras maiúsculas são menores que letras minúsculas

`'M' < 'a'`

Expressões com Strings

- Para comparar duas strings em C:

```
char n1[20], n2[20];
```

```
strcpy(n1, "ANA");
```

```
strcpy(n2, "ANAMARIA");
```

```
x = strcmp(n1, n2);
```

inserir biblioteca:
string.h

Se $n1 < n2 \rightarrow x$ recebe valor < 0

Se $n1 > n2 \rightarrow x$ recebe valor > 0

Se $n1 == n2 \rightarrow x$ recebe valor $== 0$

Exemplos

- Ex2: Comparar duas strings, considerando letras maiúsculas e minúsculas.
- Ex3: Transformar uma string de entrada em letras maiúsculas e minúsculas.

Concatenação de Strings

- Permite concatenar (juntar) duas strings em uma só.

- **strcat(s1, s2)**



s1 = "ANAMARIA"

```
char s1[20], s2[20];  
  
strcpy(s1, "ANA");  
strcpy(s2, "MARIA");  
  
strcat(s1, s2);
```

Exercícios

- Faça um programa que lê uma frase e mostra a quantidade de palavras da frase.

```
int main(){
    char frase[100];
    int i=0,count=0;
    fgets(frase,100,stdin);
    while(i< strlen(frase)){
        while (isspace(frase[i++]));
        ++count;
        while (!isspace(frase[i++]));
    }
    printf("%d", count);
    return 0;
}
```


Exercícios propostos

1. Faça um algoritmo que lê um **vetor de 30 números** inteiros e um número **n** a ser procurado no vetor. Escrever **quantas vezes n** aparece no vetor e **em quais posições**.
2. Escrever um algoritmo que lê **dois vetores** de 10 elementos inteiros e multiplica os elementos de **mesmo índice**, colocando o resultado em um terceiro vetor. No final, mostrar os dois vetores lidos e o vetor resultante.
3. Desenvolva uma solução para **ordenar** um vetor de 100 números.



Exercícios propostos

8

```
1  #include <stdio.h>
2  #include <string.h>
3  void main()
4  {
5      int vet[100], n, c, d, descubra, T;
6
7      printf("Entre com numeros inteiros\n");
8      scanf("%d", &n);
9
10     printf("Entre %d inteiros\n", n);
11
12     for ( c = 0 ; c < n ; c++ )
13         scanf("%d", &vet[c]);
14
15     for ( c = 0 ; c < ( n - 1 ) ; c++ )
16     {
17         descubra = c;
18
19         for ( d = c + 1 ; d < n ; d++ )
20         {
21             if (vet[descubra] > vet[d])
22                 descubra = d;
23         }
24         if (descubra != c )
25         {
26             T = vet[c];
27             vet[c] = vet[descubra];
28             vet[descubra] = T;
29         }
30     }
31
32     printf("O que este algoritmo faz?:\n");
33
34     for ( c = 0 ; c < n ; c++ )
35         printf("%d\n", vet[c]);
36
37     return;
38 }
39
```

Exercícios propostos

```
1  #include <stdio.h>
2  #define TAM 10
3  int main() {
4      float vetor[TAM], soma;
5      int i;
6      for(i = 0; i < TAM; i++) {
7          printf("Digite o valor da posicao %d: ", i);
8          scanf("%f", &vetor[i]);
9      }
10     for(i = 0; i < (TAM/2); i++) {
11         soma += vetor[i*2+1];
12     }
13     printf("A soma vale %g\n", soma);
14     return 0;
15 }
16
```

Exercícios propostos

```
1  #include <stdio.h>
2  #include <ctype.h>
3  #define MAX_TAM 100
4  int main() {
5      char palavra[MAX_TAM], palavra_final[MAX_TAM];
6      char c;
7      int i = 0, j = 0;
8      printf("Digite uma palavra: ");
9      scanf("%s", palavra);
10     getchar();
11     printf("Digite um caracter: ");
12     scanf("%c", &c);
13     c = tolower(c);
14     while(palavra[i] != '\0') {
15         if(tolower(palavra[i]) != c) {
16             palavra_final[j] = palavra[i];
17             j++;
18         }
19         i++;
20     }
21     palavra_final[j] = '\0';
22     printf("A palavra '%s' sem o caracter '%c' eh '%s'", palavra, c, palavra_final);
23     return 0;
24 }
25
```