

ESTRUTURAS COMPOSTAS

Variáveis Compostas Unidimensionais VETOR

slides desenvolvidos pela Profa. Rosely Sanches

ESTRUTURAS COMPOSTAS

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

VETOR

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

Calcular a média aritmética das notas de 3 alunos.

Exibir a média e as notas que estão abaixo da média

Calcular a média aritmética das notas de 3 alunos.

Exibir a média e as notas que estão abaixo da média

1ª Solução (PÉSSIMA)

- Ler as três notas uma primeira vez para calcular a média.
- Ler novamente cada nota para comparar com a média e verificar quais notas estão abaixo da média

Algoritmo PESSIMO

...declarações omitidas....

início

S = 0

para I = 1 até 3

faça início

ler(NOTA)

 S = S + NOTA

fim

fim para

MEDIA = S/3

escrever (MEDIA)

para I = 1 até 3

faça início

ler (NOTA)

se NOTA < MEDIA

então escrever (NOTA, "abaixo da média")

fim-se

fim

fim para

fim

Ler as notas uma primeira vez para calcular a média

Ler novamente cada nota para comparar com a média e verificar quais notas estão abaixo da média

programa PESSIMO
declarações omitidas....

inicio

S= 0

para l= 1 até 3

faça início

ler(NOTA)

 S = S + NOTA

fim

fim para

MEDIA = S/3

escrever (MEDIA)

para l= 1 até 3

faça início

ler (NOTA)

se NOTA < MEDIA

então escrever (NOTA, "abaixo da média")

fim-se

fim

fim para

fim

INEFICIENTE

Calcular a média aritmética das notas de 3 alunos.

Exibir a média e as notas que estão abaixo da média

2ª Solução (RUIM)

- Ler as três notas e armazenar na memória do computador, dando um nome diferente para cada nota.

programa RUIIM
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
vale para três notas

IMPRATICÁVEL!

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

Associarmos um nome para cada
nota?

Calcular a média aritmética das notas de 1000 alunos.

Exibir a média e as notas que estão abaixo da média

3ª Solução (ABORDAGEM MAIS REALISTA)

- Associar o nome NOTA ao CONJUNTO ORDENADO de notas

$$\text{NOTA} = \{N1, N2, \dots, N1000\}$$

$1^a \quad 2^a \quad 1000^a$

Calcular a média aritmética das notas de 1000 alunos.

Exibir a média e as notas que estão abaixo da média

3ª Solução (ABORDAGEM MAIS REALÍSTICA)

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]

Calcular a média aritmética das notas de 1000 alunos.

Exibir a média e as notas que estão abaixo da média

3ª Solução (ABORDAGEM MAIS REALÍSTICA)

- Chamada **VARIÁVEL INDEXADA**
- Junta-se o nome dado ao conjunto ordenado com um índice

■ para
uma

■ Exemplo:

- a 3ª nota é indicada por `NOTA[3]`
- a 1000ª nota é indicada por `NOTA[1000]`
- uma k ª nota é indicada por `NOTA[k]`

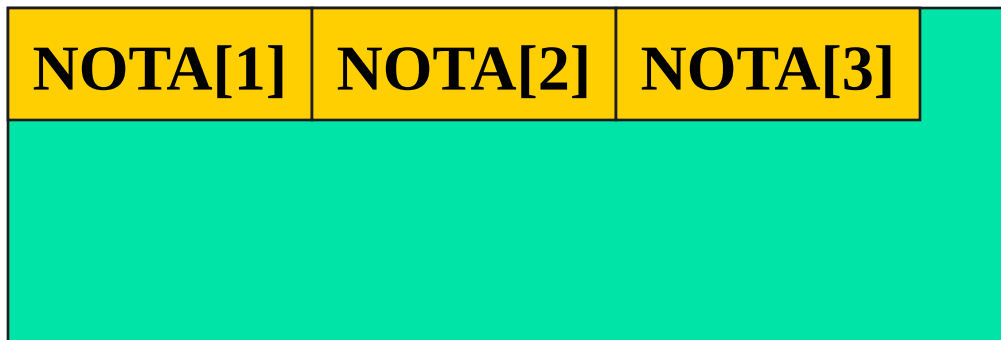
NOME

ÍNDICE

`NOTA[k]`

VARIÁVEL INDEXADA

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



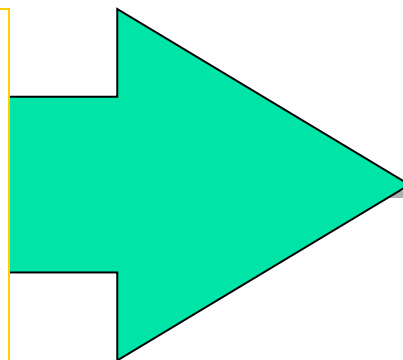
VETOR

- 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.

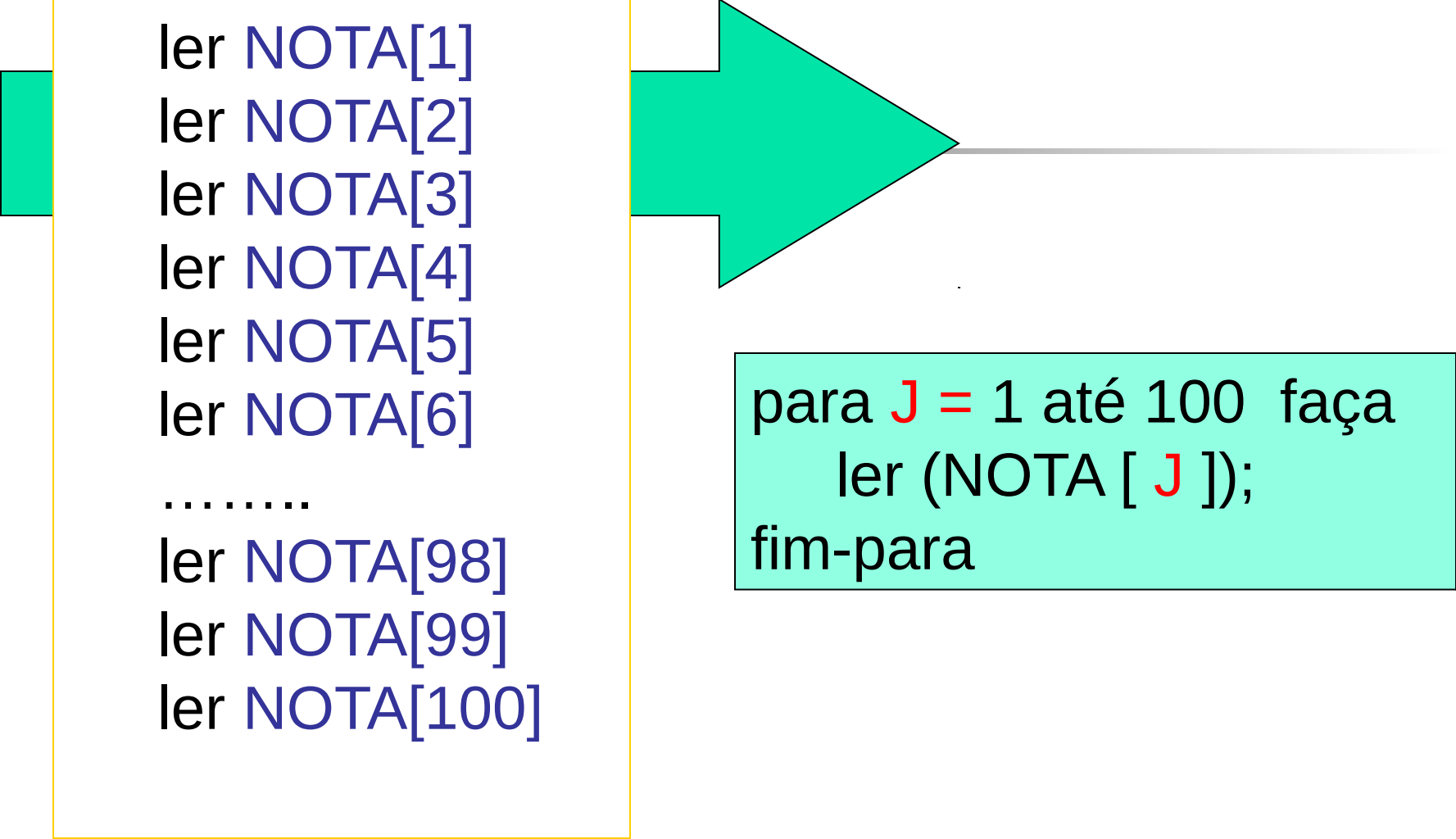
```
ler NOTA[1]
ler NOTA[2]
ler NOTA[3]
ler NOTA[4]
ler NOTA[5]
ler NOTA[6]
.....
ler NOTA[98]
ler NOTA[99]
ler NOTA[100]
```



**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
```

I, J e K
são apenas
índices que
assumem um
valor e que
junto com o
nome dado ao
conjunto
formam a
variável
indexada
NOTA

```
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...

iniciopara **J=** 1 até 100faça ler (NOTA[**J**])fim parapara **J=** 1 até 100faça escrever(NOTA[**J**])fim parafim

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 omi
inicio

```
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

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.

VETOR - Exemplo 2

Algoritmo VET2

...declarações omitidas...

inicio

```
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 \equiv 1 até 100

faça ler (NOTA[I])

fim para

SOMA \leftarrow 0

para I \equiv 1 até 100

faça SOMA = SOMA + NOTA[I]

fim para

para I \equiv 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)
```

Aumento do contador
quando a nota é 10.0

fim

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 ≡ 1 até N
  faça ler (NOTA[I])
fim para
```

```
SOMA = 0
para I ≡ 1 até N
  faça SOMA = SOMA + NOTA[I]
fim para
MEDIA = SOMA/N
```

```
para I ≡ 1 até N
  faça se NOTA[I] < MEDIA
    então escrever (NOTA[I])
  fim-se
fim para
```

fim

VETOR13a Exemplo 4

Leitura das notas

Cálculo da Média

Escrita das notas
abaixo da média

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

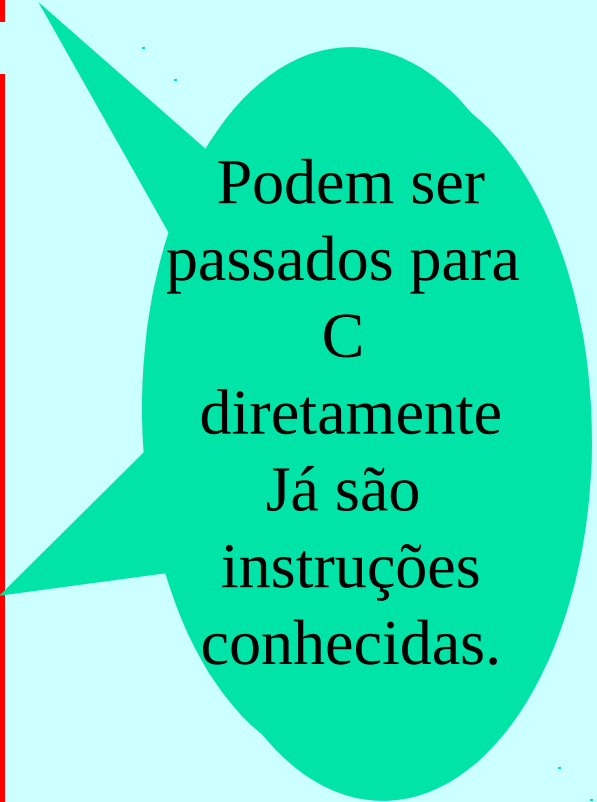
 então escrever (NOTA[I])

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

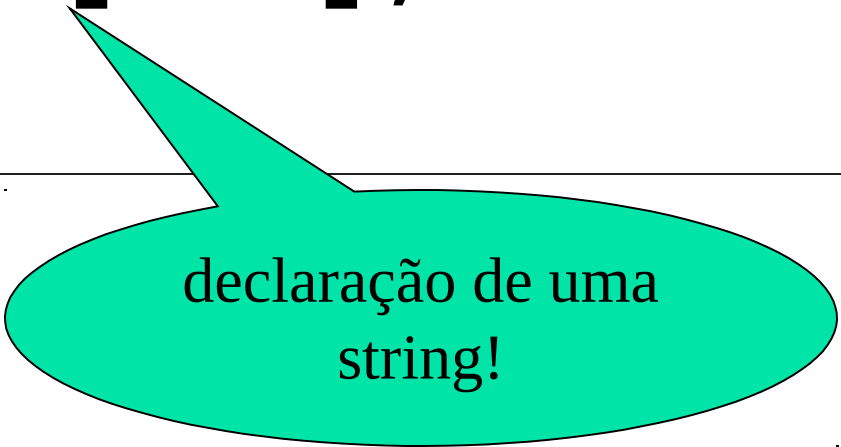
Número máximo de elementos do conjunto

```
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];
```

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**

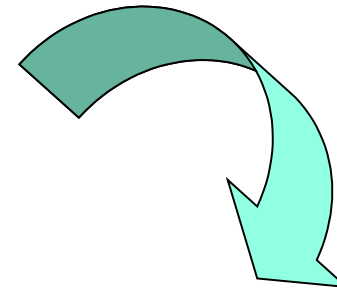
faça se NOTA[I] < MEDIA

então escrever (NOTA[I])

fim-se

fim para

fim



Passando para



```
#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

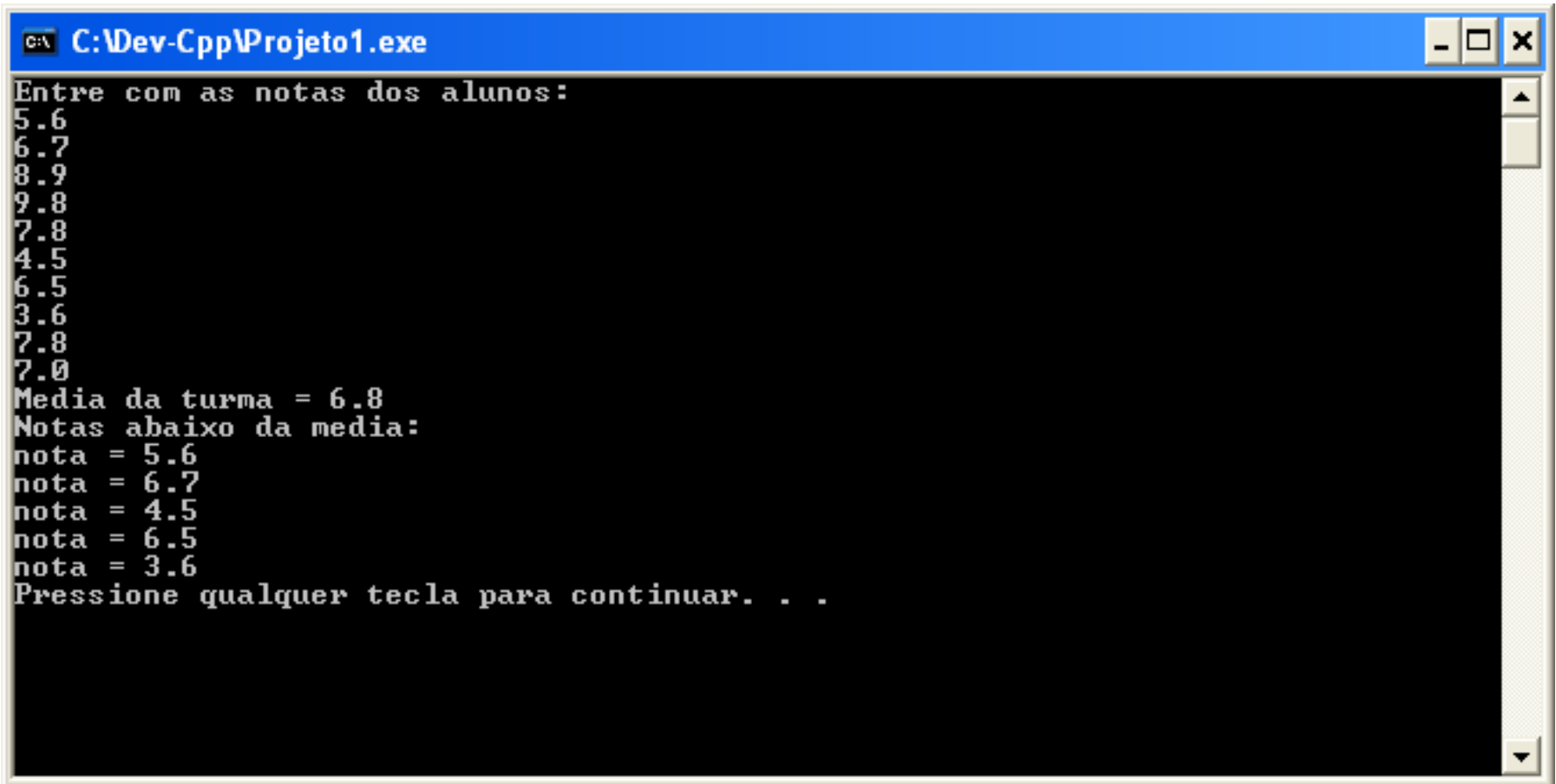
```
printf("Notas abaixo da media:\n");
for(i=0; i<MAX; i++)
    if (nota[i] < media)
        printf("nota = %.1f\n", nota[i]);
```

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.

programa MAIOR
 declarações
 início

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

Lê a quantidade
 de números

Lê os números

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

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

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]);
```

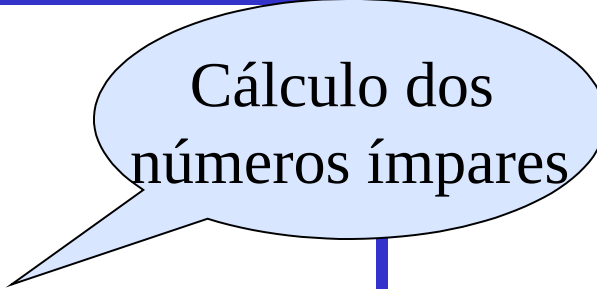


Leitura dos números

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

- **CARACTER:** 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);
```

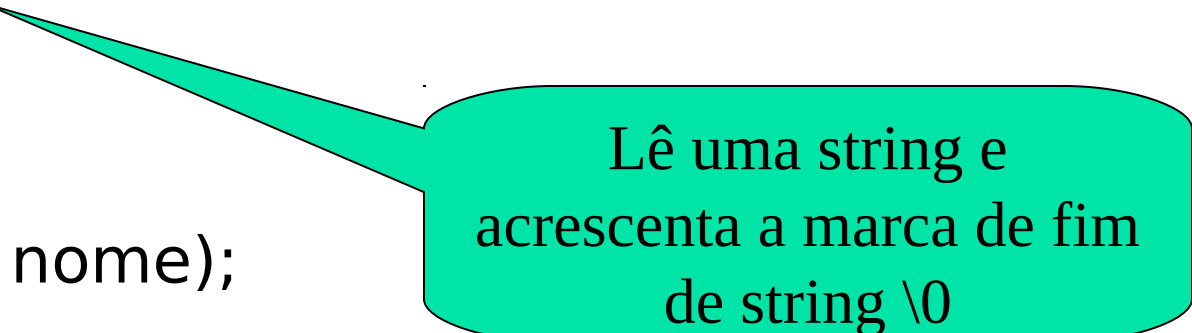
Manipulação

- **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 seqüência 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 $== 0$

Ex1: [string-compara.c](#)

Exemplos

- Ex2: Comparar duas strings, considerando letras maiúsculas e minúsculas.

`string-compara2.c`

- Ex3: Transformar uma string de entrada em letras maiúsculas e minúsculas.

`string-converte-letras.c`

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, calcula 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. 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.
2. 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**.
3. Desenvolva uma solução (pode ser somente os passos) para **ordenar** um vetor de 100 números.

