

# **INF01202**

## **Algoritmos e Programação**

### **Modalidade Ead – Turma H**

**Material de apoio: capacitação à programação  
usando funções com parâmetros tipo ponteiro  
(cont.)**

---

**Exemplo 1:**  
**Passagem**  
**de**  
**arranjo multidimensional**  
**para função, como parâmetro**  
**por endereço**

Observar na sequência o protótipo da função `exibe_2d_matriz` e sua chamada.

A primeira dimensão da matriz está sem indicação de número de elementos [ ].

Isso é possível porque o nome da matriz corresponde ao endereço de sua posição inicial, a posição zero.

Mas, para que o sistema possa determinar as posições de memória onde o armazenamento dos dados das demais dimensões tem início, ele exige que o número de elementos das demais dimensões seja fornecido.

Lembrar que os elementos de uma matriz estão em posições contíguas (contínuas) de memória.

**//apresenta matrizes**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define MAXCOL 10**

**void exhibe\_2d\_matriz(int [ ][MAXCOL], int, int );**

**int main ( )**

**{**

**int a[1][MAXCOL] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}};**

**int b[2][MAXCOL] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},  
{11, 12, 13, 14, 15, 16, 17, 18, 19, 20}};**

**int c[3][MAXCOL] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},  
{11, 12, 13, 14, 15, 16, 17, 18, 19, 20},  
{21, 22, 23, 24, 25, 26, 27, 28, 29, 30}};**

**system("color 70");**

**exibe\_2d\_matriz(a, 1, MAXCOL);**

**printf("\n\n");**

**exibe\_2d\_matriz(b, 2, MAXCOL);**

**printf("\n\n");**

**exibe\_2d\_matriz(c, 3, MAXCOL);**

**printf("\n\n");**

**system("pause");**

**return 0;**

**}**

**void exhibe\_2d\_matriz(int matriz[ ][MAXCOL], int linhas, int colunas)**

**{**

**int i, j;**

**for (i = 0; i < linhas; i++)**

**{**

**printf("\n\n");**

**for (j = 0; j < 10; j++)**

**printf("%6d", matriz[i][j]);**

**}**

**}**

**A primeira dimensão  
não precisa ser  
especificada, mas as  
demais sim!!!**

# Execução

**E:\INF0120220081Aula21\matrizbicomoparametro.exe**

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

11	12	13	14	15	16	17	18	19	20
----	----	----	----	----	----	----	----	----	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

11	12	13	14	15	16	17	18	19	20
----	----	----	----	----	----	----	----	----	----

21	22	23	24	25	26	27	28	29	30
----	----	----	----	----	----	----	----	----	----

Pressione qualquer tecla para continuar. . . \_

---

## Exemplo 2:

Gera a matriz1.

Apresenta a matriz1.

Copia a matriz1 para a matriz2.

Gera em matriz2 uma cópia elemento a elemento das duas matrizes.

Apresenta a matriz2.

```

#include <stdio.h>
#include <stdlib.h>
#define MAXCOL 10
#define MAXLIN 3
void exhibe_matriz(int [ ][MAXCOL], int , int );
void copiamatriz(int [ ][MAXCOL], int [ ][MAXCOL], int, int);
void somamatrizeselementoaelemento(int [ ][MAXCOL], int [ ][MAXCOL], int, int);
int main ( )
{
    int matriz1 [MAXLIN][MAXCOL] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                                      {11, 12, 13, 14, 15, 16, 17, 18, 19, 20},
                                      {21, 22, 23, 24, 25, 26, 27, 28, 29, 30}};

    int matriz2 [MAXLIN][MAXCOL];
    system("color 70");
    printf("\nMatriz 1\n");
    exhibe_matriz(matriz1, MAXLIN, MAXCOL);
    printf("\n\n");
    copiamatriz(matriz1, matriz2, MAXLIN, MAXCOL);
    somamatrizeselementoaelemento(matriz1, matriz2, MAXLIN, MAXCOL);
    printf("\nMatriz 2\n");
    exhibe_matriz(matriz2, MAXLIN, MAXCOL);
    printf("\n\n");
    system("pause");
    return 0;
}

```

cont.

```
void exibe_matriz(int matriz[ ][MAXCOL], int linhas, int colunas)
```

```
{  
    int i, j;  
    for (i = 0; i < linhas; i++)  
    {  
        printf("\n\n");  
        for (j = 0; j < colunas; j++)  
            printf("%6d", matriz[i][j]);  
    }  
}
```

```
void copiamatriz(int matriz1[ ][MAXCOL], int matriz2[ ][MAXCOL],  
                int linhas, int colunas)
```

```
{  
    int i, j;  
    for (i = 0; i < linhas; i++)  
        for (j = 0; j < colunas; j++)  
            matriz2[i][j] = matriz1[i][j];  
}
```

```
void somamatrizeselementoaelemento(int matriz1[ ][MAXCOL], int matriz2[ ][MAXCOL],  
                                     int linhas, int colunas)
```

```
{  
    int i, j;  
    for (i = 0; i < linhas; i++)  
        for (j = 0; j < colunas; j++)  
            matriz2[i][j] += matriz1[i][j];  
}
```



# Execução

```

C:\ E:\INF0120220081Aula21\matrizbicomoparametrosomaelementoaelemento.exe
Matriz 1
      1      2      3      4      5      6      7      8      9     10
    11     12     13     14     15     16     17     18     19     20
    21     22     23     24     25     26     27     28     29     30

Matriz 2
      2      4      6      8     10     12     14     16     18     20
    22     24     26     28     30     32     34     36     38     40
    42     44     46     48     50     52     54     56     58     60

Pressione qualquer tecla para continuar. . .

```

---

**Exemplo 3:**  
**Matriz bidimensional**  
**tratada**  
**como**  
**unidimensional**  
**(com notação de vetor)**

Observar na sequência a apresentação dos elementos dos arranjos **b** e **c**.

Como os elementos de um arranjo são armazenados em posições contíguas de memória, é possível acessar o arranjo eventualmente com um só índice, mesmo que ele tenha sido declarado com mais de uma dimensão.

Esse foi o recurso utilizado para apresentação dos arranjos **b** e **c**.

```

#include <stdio.h>
#include <stdlib.h>

long soma_matriz(int [ ], int );
long soma_matriz2(int [ ], int, int );

int main ( )
{
    int a[10]    = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int b[2][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                    {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}};
    int c[3][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                    {11, 12, 13, 14, 15, 16, 17, 18, 19, 20},
                    {21, 22, 23, 24, 25, 26, 27, 28, 29, 30}};
    system("color 70");
    printf("\nSoma dos elementos da primeira matriz = %d\n",
           soma_matriz(a, 10 ));
    printf("\nSoma dos elementos da segunda matriz = %d\n",
           soma_matriz2(b[0], 2, 10 ));
    printf("\nSoma dos elementos da terceira matriz = %d\n",
           soma_matriz2(c[0], 3, 10 ));
    printf("\n\n");
    system("pause");
    return 0;
}

```

**cont.**

---

```
long soma_matriz(int matriz[ ], int elementos)
{
    long soma = 0;
    int i;
    for (i = 0; i < elementos; i++)
        soma +=matriz[i];
    return soma;
}
```

```
long soma_matriz2(int matriz[ ], int el1, int el2)
{
    long soma = 0;
    int i;
    for (i = 0; i < el1 * el2; i++)
        soma +=matriz[i];
    return soma;
}
```

 E:\INF0120220081Aula21\teste.exe

Soma dos elementos da primeira matriz = 55

Soma dos elementos da segunda matriz = 210

Soma dos elementos da terceira matriz = 465

Pressione qualquer tecla para continuar. . . \_

---

**Exemplo 4:**  
**Matriz bidimensional**  
**tratada**  
**como**  
**unidimensional**  
**(com notação de ponteiro)**

-

Esse exemplo é o mesmo que o exemplo anterior, apenas com uma outra notação.

Observar que nas chamadas de `soma_matriz2`, `b[0]` e `c[0]` podem ser substituídos por `(int*)b` e `(int*)c`.

Tentativas de utilizar só `b` ou `c` nas chamadas resulta em erro de compilação.



```

#include <stdio.h>
#include <stdlib.h>

long soma_matriz(int *, int );
long soma_matriz2(int *, int, int );

int main ( )
{
    int a[10]    = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int b[2][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                    {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}};
    int c[3][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                    {11, 12, 13, 14, 15, 16, 17, 18, 19, 20},
                    {21, 22, 23, 24, 25, 26, 27, 28, 29, 30}};
    system("color 70");
    printf("\nSoma dos elementos da primeira matriz = %d\n",
           soma_matriz(a, 10 ));
    printf("\nSoma dos elementos da segunda matriz = %d\n",
           soma_matriz2(b[0], 2, 10 ));
    printf("\nSoma dos elementos da terceira matriz = %d\n",
           soma_matriz2(c[0], 3, 10 ));
    printf("\n\n");
    system("pause");
    return 0;
}

```

**cont.**

---

```
long soma_matriz(int matriz[ ], int elementos)
{
    long soma = 0;
    int i;
    for (i = 0; i < elementos; i++)
        soma += *(matriz + i);
    return soma;
}
```

```
long soma_matriz2(int matriz[ ], int el1, int el2)
{
    long soma = 0;
    int i;
    for (i = 0; i < el1 * el2; i++)
        soma += *(matriz + i);
    return soma;
}
```

Lembrar que os elementos de um arranjo são armazenados em posições contíguas de memória!

---

# Matrizes de *strings*:

Passagem de  
parâmetros para  
programas:

**argc** e **argv**

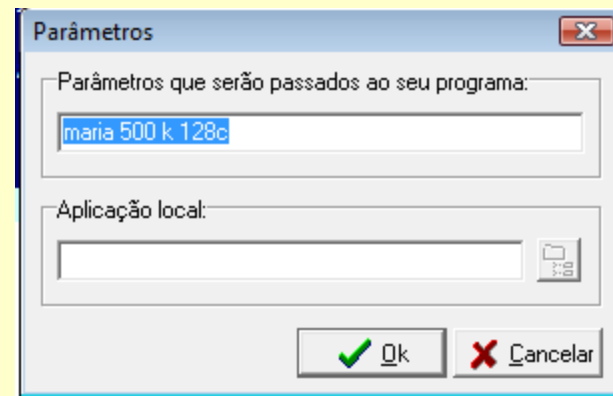
# Passagem de parâmetros para programa

A passagem de parâmetros para programas faz-se através da linha de comando com dois parâmetros na função *main* : **argc** e **argv**.

- **argc** é um parâmetro inteiro que indica o número de parâmetros que foram passados ao programa. O nome do programa sempre está incluído. É declarado como **int argc**.
- **argv** é um vetor de *strings* com todas as *strings* passadas na linha de comando. Pode ser declarado de duas formas: **char \*argv[ ]** ou **char \*\*argv**.

Definição dos valores dos parâmetros no ambiente Dev-C++:

**Menu Execute opção Parameters**



```
/*
```

Exemplo 1 de uso de argc e argv: apenas o nome do programa eh parametro.

```
*/
```

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

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

```
#include <string.h>
```

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

```
{
```

```
    system("color f1");
```

```
    printf("\nNumero de parametros = %d \n", argc);
```

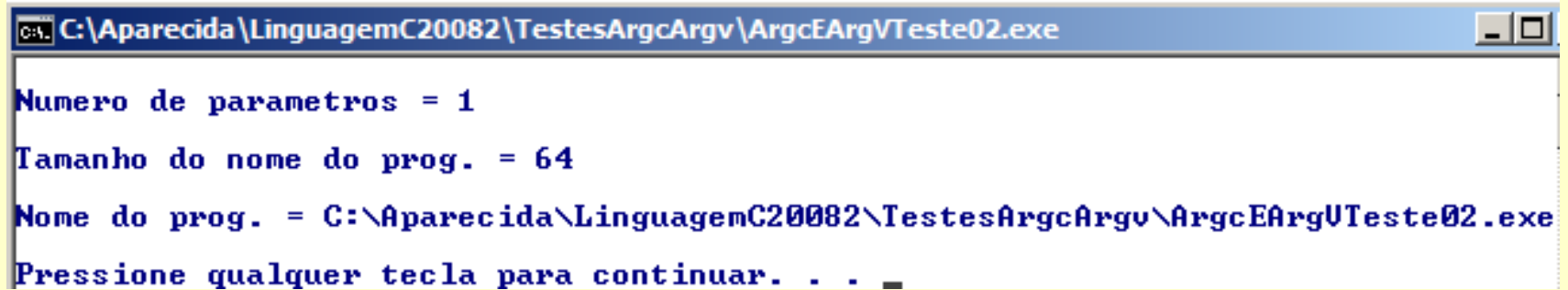
```
    printf("\nTamanho do nome do prog. = %d\n ", strlen(argv[0]));
```

```
    printf("\nNome do prog. = %s\n", argv[0]);
```

```
    system("pause");
```

```
    return 0;
```

```
}
```



A screenshot of a Windows command prompt window. The title bar at the top is blue and contains the text "C:\Aparecida\LinguagemC20082\TestesArgcArgv\ArgcEArgVTeste02.exe" and standard window control buttons. The main area of the window is white and displays the following text in a blue monospaced font: "Numero de parametros = 1", "Tamanho do nome do prog. = 64", "Nome do prog. = C:\Aparecida\LinguagemC20082\TestesArgcArgv\ArgcEArgVTeste02.exe", and "Pressione qualquer tecla para continuar. . . \_".

```
C:\Aparecida\LinguagemC20082\TestesArgcArgv\ArgcEArgVTeste02.exe  
Numero de parametros = 1  
Tamanho do nome do prog. = 64  
Nome do prog. = C:\Aparecida\LinguagemC20082\TestesArgcArgv\ArgcEArgVTeste02.exe  
Pressione qualquer tecla para continuar. . . _
```

/\*

Exemplo 2 de uso de argc e argv: alem do nome do programa, mais dois valores sao passados como parametro.

\*/

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

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

```
#include <string.h>
```

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

```
{
```

```
    system("color f1");
```

```
    printf("\nNumero de parametros = %d \n", argc);
```

```
    printf("\nTamanho do nome do prog. = %d\n ", strlen(argv[0]));
```

```
    printf("\nNome do prog. = %s\n", argv[0]);
```

```
    printf("\nPrimeiro valor apos nome do programa = %s\n", argv[1]);
```

```
    if (argc > 1)
```

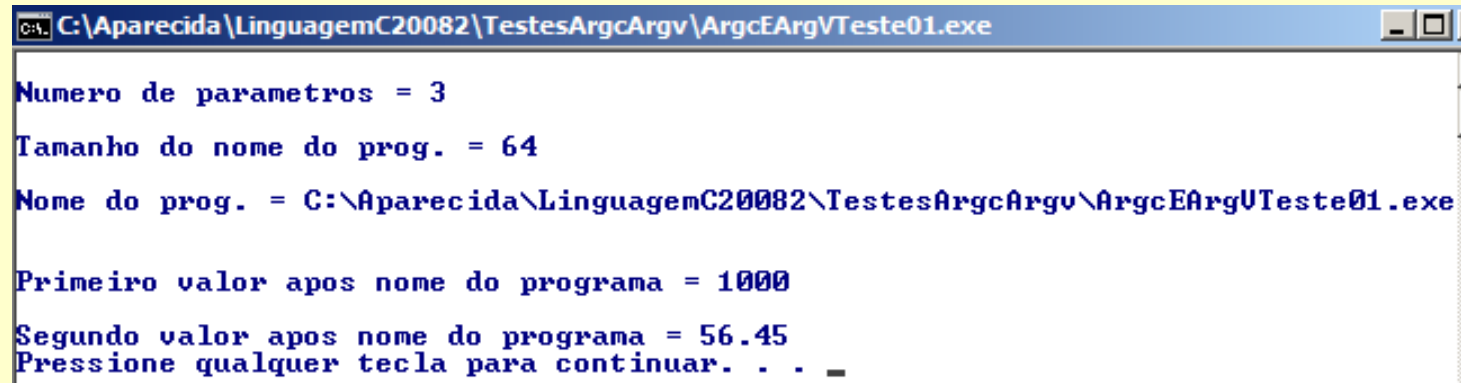
```
        printf("\nSegundo valor apos nome do programa = %s\n", argv[2]);
```

```
    system("pause");
```

```
    return 0;
```

```
}
```

---



```
C:\Aparecida\LinguagemC20082\TestesArgcArgv\ArgcEArgVTeste01.exe

Numero de parametros = 3
Tamanho do nome do prog. = 64
Nome do prog. = C:\Aparecida\LinguagemC20082\TestesArgcArgv\ArgcEArgVTeste01.exe

Primeiro valor apos nome do programa = 1000
Segundo valor apos nome do programa = 56.45
Pressione qualquer tecla para continuar. . . _
```



Seja o programa:

**exemploargvargc**

executado com os seguintes parâmetros

argv[0]	c	:	\	U	s	e	r	...	\0
argv[1]	m	a	r	i	a	\0			
argv[2]	5	0	0	\0					
argv[3]	k	\0							
argv[4]	1	2	8	c	\0				
argv[5]	N	U	L	L					

todos os parâmetros são armazenados sob a forma de *strings* (mesmo os valores numéricos).

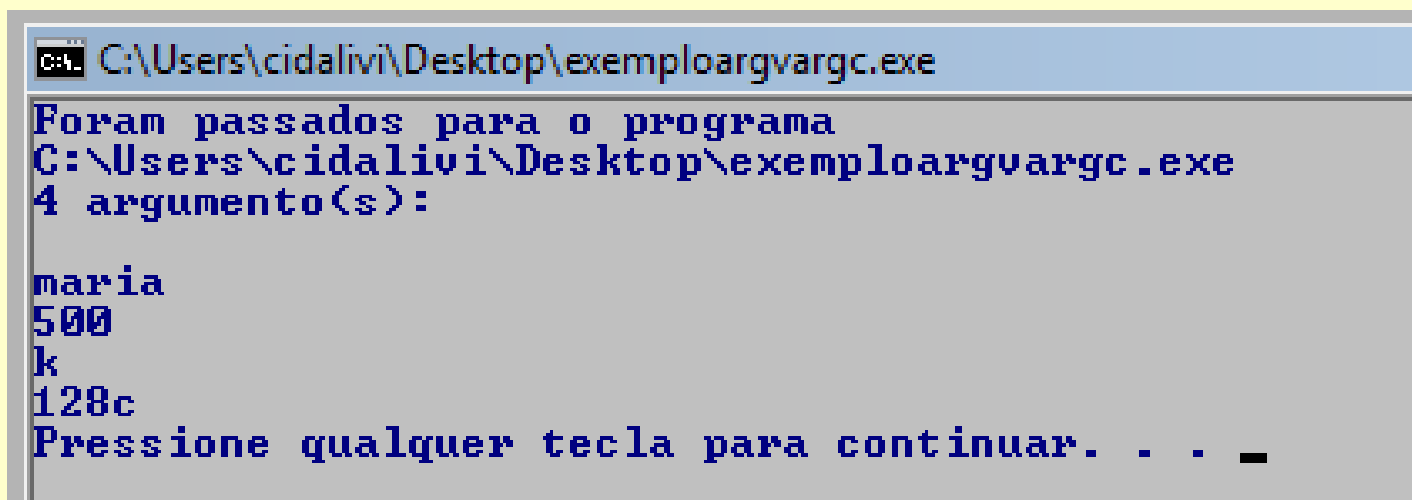
/\*

Exemplo 3 de uso de argc e argv: alem do nome do programa, mais alguns valores sao passados como parametro.

\*/

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
//char **argv: ponteiro para ponteiro
//pode ser escrito tambem como char *argv[ ]
{
    int i;
    system("color 71");
    if (argc < 2 )
        printf("\nNenhum parametro foi passado!\n");
    else
    {
        printf
            ("Foram passados para o programa \n%s \n%d argumento(s):\n\n",
             argv[0] , argc-1);
        for(i = 1; i < argc; i++)
            printf("%s\n", argv[i]);
    }
    system("pause");
    return 0;
}
```

# Execução



```
C:\Users\cidalivi\Desktop\exemploargvargc.exe
Foram passados para o programa
C:\Users\cidalivi\Desktop\exemploargvargc.exe
4 argumento(s):

maria
500
k
128c
Pressione qualquer tecla para continuar. . . _
```

---

## Exemplo 5 (versão 1):

**Matrizes de strings:  
Leitura e apresentação de  
nomes e sobrenomes  
armazenados em uma matriz de  
*strings***

Neste exemplo o arranjo **nomes** armazena para várias pessoas seu primeiro nome e mais seus sobrenomes, até um máximo de 9 sobrenomes.

Na declaração do arranjo ele aparece com três dimensões, porque os nomes são armazenados em vetores, já que **strings** em C são vetores de caracteres.

Assim, neste exemplo a primeira dimensão (linha) define a pessoa de quem está se falando; a segunda dimensão define o nome a que se está referindo para a pessoa que está sendo acessada; e a terceira dimensão, que se refere aos *strings* dos nomes, só será usada se for desejado alterar algum caractere de algum nome.

Os nomes de uma determinada pessoa estão então em um vetor de *strings*. Neste problema, o final deste vetor é sinalizado com um *string* vazio, apenas contendo '\0'.

Na apresentação dos nomes de uma pessoa, a posição **nomes[i] [j] [0]** é testada, e quando for zero, o laço de apresentação dos nomes é interrompido:

```
for (j = 0; nomes[i] [j] [0]; j++)  
    printf("%s ", nomes[i][j]);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define NOMES 4
#define NUMNOMES 11
//primeiro nome, mais sobrenomes, ate no maximo 9 mais posição para \0

#define TAMNOMES 20
int main( )
{
    char nomes[NOMES][NUMNOMES][TAMNOMES];
    //primeira dimensao: define quantas pessoas que serão processadas
    //segunda dimensao: define quantos nomes para cada pessoa
    //terceira dimensao: define o tamanho dos strings dos nomes
    int i, j, numsobr;
    system("color 70");
```

cont.

---

```
for (i = 0; i < NOMES; i++)
{
    printf("\nDigite o primeiro nome da pessoa %d:   ", i + 1);
    fflush(stdin);
    fgets(nomes[i] [0] , sizeof(nomes[i] [0]), stdin);
    if (nomes[i] [0][strlen(nomes[i] [0]) - 1] == '\n')
        nomes[i] [0][strlen(nomes[i] [0]) - 1] = '\0';

do
{
    printf("\n  Numero de sobrenomes (maximo de %d): \n", NUMNOMES - 2);
    scanf("%d", &numsobr);
    if ((numsobr > (NUMNOMES - 1)) || (numsobr < 1))
        printf("\n  Numero de sobrenomes: maior que zero e menor que %d\n",
            NUMNOMES - 1);
}
while ((numsobr > (NUMNOMES - 1)) || (numsobr < 1));
```

cont.

```

for (j = 1; j <= numsobr; j++)
{
    printf("\n  Sobrenome %d com maximo de %d caracteres: \n",
        j, TAMNOMES);
    fflush(stdin);
    fgets(nomes[i][j], sizeof(nomes[i][j]), stdin);
    if (nomes[i][j][strlen(nomes[i][j]) - 1] == '\n')
        nomes[i][j][strlen(nomes[i][j]) - 1] = '\0';
}
nomes[i][j][0] = '\0'; // sinalizando o final do vetor de strings
}
printf("\nNomes lidos\n");
for (i = 0; i < NOMES; i++)
{
    printf("\n");
    for (j = 0; nomes[i][j][0]; j++)
        printf("%s ", nomes[i][j]);
}
printf("\n");
system("pause");
return 0;
}

```



---

## Exemplo 5 (versão 2):

Matrizes de *strings*:  
Leitura e apresentação de  
nomes e sobrenomes  
armazenados em uma matriz de  
*strings*

---

Nesta segunda versão do problema anterior, um vetor auxiliar é utilizado para armazenar o número de sobrenomes das pessoas, **numsobr[NOMES]**.

Na apresentação dos nomes de uma pessoa, esse vetor é utilizado para controlar quantos nomes devem ser apresentados:

```
for (j = 0; j <= numsobr[i]; j++)  
    printf("%s ", nomes[i][j]);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define NOMES 4
#define NUMNOMES 11
//primeiro nome, mais sobrenomes, ate no maximo 9
```

```
#define TAMNOMES 20
```

```
int main( )
```

```
{
```

```
    char nomes[NOMES][NUMNOMES][TAMNOMES];
```

```
    //primeira dimensao: define quantas pessoas que serão processadas
```

```
    //segunda dimensao: define quantos nomes para cada pessoa
```

```
    //terceira dimensao: define o tamanho dos strings dos nomes
```

```
    int i, j, numsobr[NOMES];
```

```
    //vetor numsobr: armazena o numero de nomes de cada pessoa
```

```
    system("color 70");
```

cont.

```

for (i = 0; i < NOMES; i++)
{
    printf("\nDigite o primeiro nome da pessoa %d:   ", i + 1);
    fflush(stdin);
    fgets(nomes[i] [0] , sizeof(nomes[i] [0]), stdin);
    if (nomes[i] [0][strlen(nomes[i] [0]) - 1] == '\n')
        nomes[i] [0][strlen(nomes[i] [0]) - 1] = '\0';
    do
    {
        printf("\n  Numero de sobrenomes (maximo de %d): \n", NUMNOMES - 2);
        scanf("%d", &numsobr[i]);
        if ((numsobr[i] > (NUMNOMES - 1)) || (numsobr[i] < 1))
            printf("\n  Numero de sobrenomes: maior que zero e menor que %d\n",
                NUMNOMES - 1);
    }
    while ((numsobr[i] > (NUMNOMES - 1)) || (numsobr[i] < 1));
}

```

**cont.**

```

for (j = 1; j <= numsobr[i]; j++)
{
    printf("\n  Sobrenome %d com maximo de %d caracteres: \n",
           j , TAMNOMES);
    fflush(stdin);
    fgets(nomes[i] [j] , sizeof(nomes[i] [j]), stdin);
    if (nomes[i] [j][strlen(nomes[i] [j]) - 1] == '\n')
        nomes[i] [j][strlen(nomes[i] [j]) - 1] = '\0';
}
}
printf("\nNomes lidos\n");
for (i = 0; i < NOMES; i++)
{
    printf("\n");
    for (j = 0; j <= numsobr[i]; j++)
        printf("%s ", nomes[i][j]);
}
printf("\n");
system("pause");
return 0;
}

```