

USP-ICMC-BInfo

# Vetores em C

SCC501 - ICC-II

2011

Prof. João Luís

```

#include <stdio.h>
main()
{
    int total;
    int custobala[3] = {20, 30, 5}; // cria e inicia vetor
    int bala[3]; // cria um vetor com três elementos
    bala[0] = 7; // atribui valor ao primeiro elemento
    bala[1] = 8;
    bala[2] = 6;

    printf("Total de balas = ");
    printf("%d\n", (bala[0] + bala[1] + bala[2]));
    printf("O pacote com %d balas custa ", bala[1]);
    printf("%d centavos por bala.\n", custobala[1]);

    total = bala[0] * custobala[0] + bala[1] * custobala[1];
    total = total + bala[2] * custobala[2];

    printf("A despesa total de balas %c %d centavos.\n", 130,
total);
    printf("\nTamanho do vetor bala = %d", sizeof bala);
    printf(" bytes.\n");
    printf("Tamanho de um elemento = %d", sizeof bala[0]);
    printf(" bytes.\n");
}

```

Saída:

Total de balas = 21

O pacote com 8 balas custa 30 centavos por bala.

A despesa total de balas é 410 centavos.

Tamanho do vetor bala = 12 bytes.

Tamanho de um elemento = 4 bytes.

```

#include <stdio.h>
#include <string.h>

#define Tam 15

main()
{
    char nome1[Tam]; // vetor vazio
    char nome2[Tam] = "Programa C"; // vetor iniciado

    printf("Ol%c! Eu sou o %s", 160, nome2);
    printf("! Qual %c o seu nome?\n", 130);
    gets(nome1);
    printf("Bem, %s, seu nome tem ", nome1);
    printf("%d letras e est%c armazenado\n", strlen(nome1), 160);
    printf("em um vetor de %d bytes.\n", sizeof nome1);
    printf("Sua inicial %c %c.\n", 130, nome1[0]);
    nome2[3] = '\0'; // caractere null
    printf("Aqui est%co os primeiros 3 caracteres do meu nome: ",
    198);
    printf("%s\n", nome2);
}

```

Saída:

Olá! Eu sou o Programa C! Qual é o seu nome?

João Luís

Bem, João Luís, seu nome tem 9 letras e está armazenado em um vetor de 15 bytes.

Sua inicial é J.

Aqui estão os primeiros 3 caracteres do meu nome: Pro

```
#include <stdio.h>
#define TamVet 16
void main()
{
    double fatoriais[TamVet];
    int i;
    fatoriais[1] = fatoriais[0] = 1.0;
    for (i = 2; i < TamVet; i++)
        fatoriais[i] = i * fatoriais[i-1];
    for (i = 0; i < TamVet; i++)
        printf("%d! = %g\n", i, fatoriais[i]);
}
```

0! = 1  
1! = 1  
2! = 2  
3! = 6  
4! = 24  
5! = 120  
6! = 720  
7! = 5040  
8! = 40320  
9! = 362880  
10! = 3.6288e+006  
11! = 3.99168e+007  
12! = 4.79002e+008  
13! = 6.22702e+009  
14! = 8.71783e+010  
15! = 1.30767e+012

```
#include <stdio.h>
#include <string.h>
#define TamVet 20

main()
{
    char palavra[TamVet];
    int i;
    printf("Entre com uma palavra: ");
    scanf("%s", palavra);
    // mostra as letras na ordem reversa
    for (i = strlen(palavra) - 1; i >= 0; i--)
        printf("%c", palavra[i]);
    printf("\n");
}
```

Saída:

Entre com uma palavra: Abracadabra  
arbadacarbA



```
#include <stdio.h>
#include <string.h>
#define TamVet 20

main()
{
    char palavra[TamVet];
    char temp;
    int i, j;
    printf("Entre com uma palavra: ");
    scanf("%s", palavra);
    // fisicamente modifica vetor
    for (j = 0, i = strlen(palavra) - 1; j < i; i--, j++)
    { // começa bloco
        temp = palavra[i];
        palavra[i] = palavra[j];
        palavra[j] = temp;
    } // termina bloco
    printf("%s\n", palavra);
}
```

```
#include <stdio.h>

main()
{
    int placarquiz[10] = {20, 20, 20, 20, 20, 19, 20,
        18, 20, 20};
    int i;
    printf("Fazendo certo:\n");
    for (i = 0; placarquiz[i] == 20; i++)
        printf("pergunta %d vale 20\n", i);
    printf("\nFazendo perigosamente errado:\n");
    for (i = 0; placarquiz[i] = 20; i++)
        printf("pergunta %d vale 20\n", i);
}
```

Fazendo certo:

pergunta 0 vale 20  
pergunta 1 vale 20  
pergunta 2 vale 20  
pergunta 3 vale 20  
pergunta 4 vale 20

Fazendo perigosamente errado:

pergunta 0 vale 20  
pergunta 1 vale 20  
pergunta 2 vale 20  
pergunta 3 vale 20  
pergunta 4 vale 20  
pergunta 5 vale 20  
pergunta 6 vale 20  
pergunta 7 vale 20  
pergunta 8 vale 20  
pergunta 9 vale 20  
pergunta 10 vale 20  
pergunta 11 vale 20  
pergunta 12 vale 20  
pergunta 13 vale 20  
pergunta 14 vale 20  
pergunta 15 vale 20  
pergunta 16 vale 20

pergunta 17 vale 20  
pergunta 18 vale 20  
pergunta 19 vale 20  
pergunta 20 vale 20  
pergunta 21 vale 20  
pergunta 22 vale 20  
pergunta 23 vale 20  
pergunta 24 vale 20  
pergunta 25 vale 20  
pergunta 26 vale 20  
pergunta 27 vale 20  
pergunta 28 vale 20  
pergunta 29 vale 20  
pergunta 30 vale 20  
pergunta 31 vale 20  
pergunta 32 vale 20  
pergunta 33 vale 20  
pergunta 34 vale 20  
pergunta 35 vale 20  
pergunta 36 vale 20  
pergunta 37 vale 20  
pergunta 38 vale 20  
pergunta 39 vale 20  
pergunta 40 vale 20  
pergunta 41 vale 20

```

#include <stdio.h>
#include <string.h> // protótipo para strcmp()

main()
{
    char palavra[5] = "?ate";
    char ch;
    for (ch = 'a'; strcmp(palavra, "mate"); ch++)
    {
        printf("%s\n", palavra);
        palavra[0] = ch;
    }
    printf("Depois que o loop termina, a palavra %c %s\n",
        130, palavra);
}

```

	fate
Saída:	gate
?ate	hate
aate	iate
bate	jate
cate	kate
date	late
eate	Depois que o loop termina, a palavra é mate <sup>12</sup>

```

#include <stdio.h>

const int TamVet = 8;

int som_vet(int vet[], int n); // protótipo

void main()
{
    int biscoitos[8] = {1,2,4,8,16,32,64,128};
    int soma = som_vet(biscoitos, TamVet);
    printf("Total de biscoitos comidos: %d\n", soma);
}

// retorna a soma de um vetor de inteiros
int som_vet(int vet[], int n)
{
    int i, total = 0;
    for (i = 0; i < n; i++)
        total = total + vet[i];
    return total;
}

```

Saída:

**Total de biscoitos comidos: 255**

```

#include <stdio.h>

const int TamVet = 8;

int som_vet(int vet[], int n);

void main()
{
    int biscoitos[8] = {1,2,4,8,16,32,64,128};
    int soma;
    printf("%d = endereço do vetor, ", biscoitos, 135);
    printf("%d = sizeof biscoitos\n", sizeof biscoitos);
    soma = som_vet(biscoitos, TamVet);
    printf("Total de biscoitos comidos: %d\n", soma);
    soma = som_vet(biscoitos, 3); // uma mentira
    printf("Os primeiros três comedores comeram %d
    biscoitos.\n", 136, soma);
    soma = som_vet(biscoitos + 4, 4); // outra mentira
    printf("Os últimos quatro comedores comeram %d
    biscoitos.\n", 163, soma);
}

```

```
// retorna a soma de um vetor de inteiros

int som_vet(int vet[], int n)
{
    int i, total = 0;
    printf("%d = vet, ", vet);
    printf("%d = sizeof vet\n", sizeof vet);
    for (i = 0; i < n; i++)
        total = total + vet[i];
    return total;
}
```

### Saída:

```
1245024 = endereço do vetor, 32 = sizeof biscoitos
1245024 = vet, 4 = sizeof vet
Total de biscoitos comidos: 255
1245024 = vet, 4 = sizeof vet
Os primeiros três comedores comeram 7 biscoitos.
1245040 = vet, 4 = sizeof vet
Os últimos quatro comedores comeram 240 biscoitos.
```

## Implicações do uso de vetores como argumentos

int soma\_vet (int vet[ ], int n);

diz qual é o endereço do vetor    diz quantos elementos  
processar

diz o tipo do vetor

o mesmo que \*vet, significa que vet é um ponteiro

Informa à função sobre um vetor



```
#include <stdio.h>

const int Max = 5;

// protótipos de função

int preenche_vetor(float ve[], int limite);

void mostra_vetor(const float ve[], int n); // não muda os dados

void reavalua(float r, float ve[], int n);

void main()
{
    float propriedades[5];
    float taxa;
    int tamanho = preenche_vetor(propriedades, Max);
    mostra_vetor(propriedades, tamanho);
    printf("Entre com a taxa de reavaliacao: ");
    scanf("%f", &taxa);
    reavalua(taxa, propriedades, tamanho);
    mostra_vetor(propriedades, tamanho);
}
```

```
int preenche_vetor(float ve[], int limite)
{
    float temp;
    int i;
    for (i = 0; i < limite; i++)
    {
        printf("Entre com valor #%d: ", (i + 1));
        scanf("%f", &temp);
        if (temp < 0)
            break;
        ve[i] = temp;
    }
    return i;
}
```

```
// a seguinte função pode usar, mas não alterar,  
// o vetor cujo endereço é ve
```

```
void mostra_vetor(const float ve[], int n)  
{  
    int i;  
    for (i = 0; i < n; i++)  
    {  
        printf("Propriedade #%d: $", (i + 1));  
        printf("%g\n", ve[i]);  
    }  
}
```

```
// multiplica cada elemento de ve[] por r
```

```
void reavalua(float r, float ve[], int n)  
{  
    int i;  
    for (i = 0; i < n; i++)  
        ve[i] *= r;  
}
```

Duas saídas:

Entre com valor #1: 100000

Entre com valor #2: 80000

Entre com valor #3: 222000

Entre com valor #4: 240000

Entre com valor #5: 118000

Propriedade #1: \$100000

Propriedade #2: \$80000

Propriedade #3: \$222000

Propriedade #4: \$240000

Propriedade #5: \$118000

Entre com a taxa de reavaliação: 1.10

Propriedade #1: \$110000

Propriedade #2: \$88000

Propriedade #3: \$244200

Propriedade #4: \$264000

Propriedade #5: \$129800

Entre com valor #1: 200000

Entre com valor #2: 84000

Entre com valor #3: 160000

Entre com valor #4: -2

Propriedade #1: \$200000

Propriedade #2: \$84000

Propriedade #3: \$160000

Entre com a taxa de reavaliação: 1.20

Propriedade #1: \$240000

Propriedade #2: \$100800

Propriedade #3: \$192000

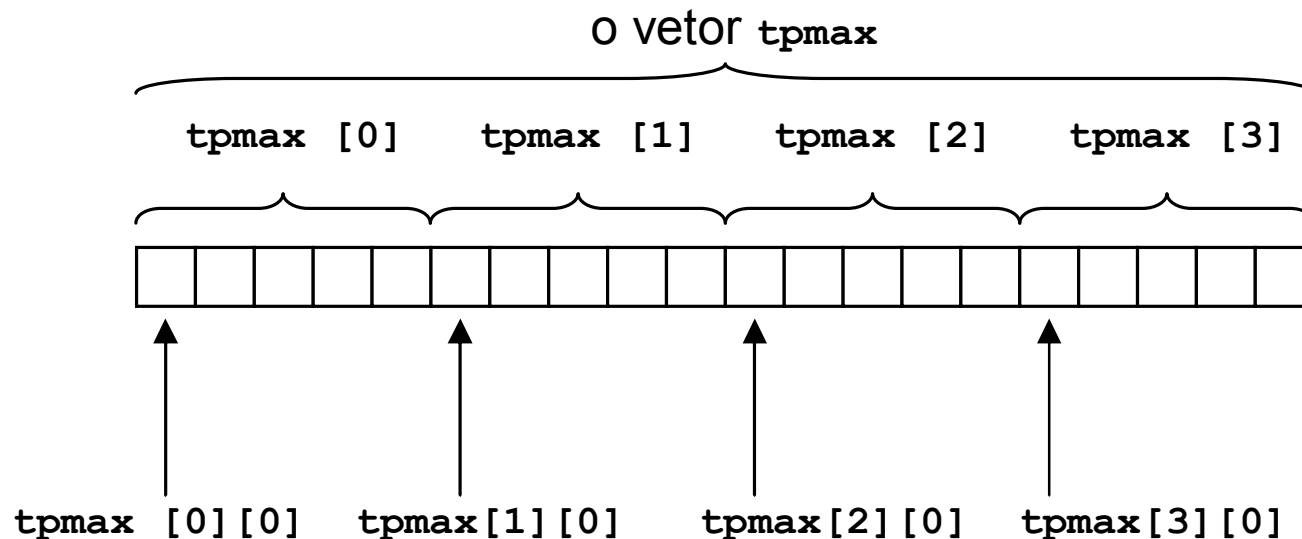
## VETORES BIDIMENSIONAIS (MATRIZES)

`int tpmx [4][5];` → `tpmx` é um vetor com 4 elementos. Cada um destes elementos é um vetor de 5 inteiros. Na verdade, trata-se de um vetor de duas dimensões de números inteiros.

`tpmx` é um vetor de 4 elementos

```
int tpmx [4][5];
```

cada elemento é um vetor de 5 *ints*.



```
int tpmx[4][5];
```

O vetor `tpmx` é visto como uma tabela (ou matriz):

		0	1	2	3	4
<code>tpmx[0]</code>	0	<code>tpmx[0][0]</code>	<code>tpmx[0][1]</code>	<code>tpmx[0][2]</code>	<code>tpmx[0][3]</code>	<code>tpmx[0][4]</code>
<code>tpmx[1]</code>	1	<code>tpmx[1][0]</code>	<code>tpmx[1][1]</code>	<code>tpmx[1][2]</code>	<code>tpmx[1][3]</code>	<code>tpmx[1][4]</code>
<code>tpmx[2]</code>	2	<code>tpmx[2][0]</code>	<code>tpmx[2][1]</code>	<code>tpmx[2][2]</code>	<code>tpmx[2][3]</code>	<code>tpmx[2][4]</code>
<code>tpmx[3]</code>	3	<code>tpmx[3][0]</code>	<code>tpmx[3][1]</code>	<code>tpmx[3][2]</code>	<code>tpmx[3][3]</code>	<code>tpmx[3][4]</code>

- Para imprimir todo o conteúdo de `tpmax`, use dois *loops for*, um para as linhas e outro para as colunas:

```
for (lin = 0; lin < 4; lin++)  
{  
    for (col = 0; col < 5; col++)  
        printf("%d\t", tpmax[lin][col]);  
    printf("\n");  
}
```

# *Iniciando um vetor bidimensional*

- Para um vetor de uma única dimensão, como já visto, basta colocar entre chaves, os valores das posições do vetor na ordem:

```
int vetor[5] = { 23, 26, 24, 31, 28 };
```

- Para um vetor de duas dimensões, inicia-se linha por linha:

```
int tpmax[4][5] =          // vetor de duas dimensões
{
    {27, 30, 22, 35, 33},    // valores para tpmax[0]
    {30, 31, 25, 38, 37},    // valores para tpmax[1]
    {26, 25, 24, 33, 36},    // valores para tpmax[2]
    {28, 32, 23, 37, 35}     // valores para tpmax[3]
};
```



```
#include <stdio.h>

#define Cidades 5
#define Anos 4

main()
{
    int cidade, ano;

    const char * cidades[Cidades] = // vetor de ponteiros
    { // para 5 strings
        "Acolandia",
        "Bertovila",
        "Nova Parada",
        "Sao Austerio",
        "Potrenquinho"
    };
};
```

```

int tpmax[Anos][Cidades] =           // vetor 2-D
{
    {27, 30, 22, 35, 33},           // valores para tpmax[0]
    {30, 31, 25, 38, 37},           // valores para tpmax[1]
    {26, 25, 24, 33, 36},           // valores para tpmax[2]
    {28, 32, 23, 37, 35}           // valores para tpmax[3]
};

printf("Temperaturas maximas para 2000 - 2005\n\n");

for (cidade = 0; cidade < Cidades; cidade++)
{
    printf("%s:\t", cidades[cidade]);
    for (ano = 0; ano < Anos; ano++)
        printf("%d\t", tpmax[ano][cidade]);
    printf("\n");
}
}

```

Saída:

**Temperaturas maximas para 2000 - 2005**

<b>Acolandia:</b>	<b>27</b>	<b>30</b>	<b>26</b>	<b>28</b>
<b>Bertovila:</b>	<b>30</b>	<b>31</b>	<b>25</b>	<b>32</b>
<b>Nova Parada:</b>	<b>22</b>	<b>25</b>	<b>24</b>	<b>23</b>
<b>Sao Austerio:</b>	<b>35</b>	<b>38</b>	<b>33</b>	<b>37</b>
<b>Potrenquinho:</b>	<b>33</b>	<b>37</b>	<b>36</b>	<b>35</b>