

# Vetores Multidimensionais

Prof.: Leonardo Tórtoro Pereira

[leonardop@usp.br](mailto:leonardop@usp.br)

# Vetores Multidimensionais [1]

- Os vetores podem ser combinados em “vetores de vetores”
- Isso pode ser estendido para um número qualquer de dimensões
- Quando usamos duas dimensões, costumamos chamar de matriz, pois a referência a seus valores é similar
- Para declarar uma dimensão extra basta colocar novos colchetes no vetor, com o tamanho desta nova dimensão

# Vetores Multidimensionais

→ `int notas[2][3] = {{10, 9, 8}, {5, 6, 7}};`

|    |   |   |
|----|---|---|
| 10 | 9 | 8 |
| 5  | 6 | 7 |

# Vetores Multidimensionais

```
#include <stdio.h>
int main()
{
    int notas[2][3] = {{10, 9, 8}, {5, 6, 7}};
    for(int i = 0; i < 2; ++i)
    {
        for(int j = 0; j < 3; ++j)
            printf("%2d ", notas[i][j]);
        printf("\n");
    }
    return 0;
}
```

# Vetores Multidimensionais [1]

- No caso de matrizes, o acesso à “primeira” dimensão corresponde às linhas
- Já o acesso à “segunda” dimensão corresponde às colunas
- Vejamos o exemplo a seguir

# Vetores Multidimensionais

→ notas[1][2] = ???

|    |   |   |
|----|---|---|
| 10 | 9 | 8 |
| 5  | 6 | 7 |

# Vetores Multidimensionais

→ notas[1][2] = ???

|    |   |   |
|----|---|---|
| 10 | 9 | 8 |
| 5  | 6 | 7 |

# Vetores Multidimensionais [1]

- Um vetor pode ter um número qualquer de dimensões
- Mas cuidado pois o tamanho dele é a MULTIPLICAÇÃO de cada índice!
- `int cubo [100][100][100]`
- Este exemplo teria  $100 \times 100 \times 100 \times 4$  bytes de memória
  - ◆  $8 \times 10^6 = 8$  milhões de bytes!
  - ◆ Isso normalmente nem é possível alocar na pilha
- `int cubo [100][100][10]` é possível normalmente :)



# Vetores Multidimensionais

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int cubo [100][100][10];
    printf("Size of cubo: %lu\n", sizeof(cubo));
    return 0;
}
```

# Vetores Multidimensionais [1]

- Na verdade, vetores multidimensionais são praticamente iguais vetores unidimensionais!
- Apenas facilitam a abstração pros programadores

```
int notas[2][3]  
int notas[6]
```

- Ambas são equivalentes, mas no primeiro caso o compilador automaticamente guarda a profundidade de cada dimensão imaginária

# Vetores Multidimensionais

```
#include <stdio.h>
#define LINHAS 3
#define COLUNAS 4
int main() {
    int notas[LINHAS][COLUNAS] = {{10, 9, 8, 9},
                                   {5, 6, 7, 8}, {3, 5, 8, 9}};
    for(int i = 0; i < LINHAS; ++i) {
        for(int j = 0; j < COLUNAS; ++j)
            printf("%2d ", notas[i][j]);
        printf("\n");
    }
    return 0;
}
```

# Vetores Multidimensionais

```
#include <stdio.h>
#define LINHAS 3
#define COLUNAS 4
int main() {
    int notas[LINHAS*COLUNAS] = {10, 9, 8, 9,
                                   5, 6, 7, 8,
                                   3, 5, 8, 9};
    for(int i = 0; i < LINHAS; ++i) {
        for(int j = 0; j < COLUNAS; ++j)
            printf("%2d ", notas[i*COLUNAS+j]);
        printf("\n");
    }
    return 0;
}
```

## Vetores Multidimensionais [2]

- Você PRECISA declarar o tamanho de qualquer dimensão da matriz que não seja a mais da esquerda!

```
int a[][][2] = { {{1, 2}, {3, 4}}, {{5, 6}, {7, 8}}
```

- O exemplo acima NÃO FUNCIONA!
- Mas este abaixo funciona:

```
int a[][2][2] = { {{1, 2}, {3, 4}}, {{5, 6}, {7, 8}}
```

# Vetores Multidimensionais

- Um detalhe importante é que matrizes também podem ser compostas de caracteres
- Um ótimo jeito de armazenar uma lista de palavras!

# Vetores Multidimensionais

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

#define NPALAVRAS 3
#define MAXLETRAS 15

int main() {
    char palavras[NPALAVRAS][MAXLETRAS];
    for(int i = 0; i < NPALAVRAS; ++i)
        scanf("%s", palavras[i]);
    for(int i = 0; i < NPALAVRAS; ++i)
        printf("%s\n", palavras[i]);
    return 0;
}
```

# Referências

1. <http://www.cplusplus.com/doc/tutorial/arrays/>
2. <https://www.geeksforgeeks.org/g-fact-44/>