5950257 - Programação de Computadores

Aula 08 - Arrays e Matrizes

Prof. Dr. Denis M. L. Martins
DCM | FFCLRP | USP

Objetivos de Aprendizagem

- Compreender o conceito de vetor (array) unidimensional e multidimensional em C
- Declarar e inicializar arrays corretamente
- Acessar e modificar elementos de um vetor
- Utilizar estruturas de repetição com arrays
- Desenvolver algoritmos utilizando arrays

> Para os exercícios desta aula, você pode utilizar IDEs online como:
>
> https://www.onlinegdb.com/online_c_compiler
>
> https://www.programiz.com/c-programming/online-compiler/

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Problema

Considere um programa que calcula a média de notas de um aluno.
Para cada prova é necessário criar uma variável que armazena o valor da nota.

```
int notaProva1 = 8;
int notaProva2 = 7;
int notaProva3 = 9;
```

Problemas:

- Se precisar adicionar mais uma prova, precisa criar mais uma variável.
- Variáveis estão desconectadas umas das outras, embora compartilhem o mesmo conceito.

Precisamos de um contêiner lógico que possa manter **múltiplos** elementos do mesmo tipo, na mesma ordem, sem poluir o código.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Solução com Arrays (Vetores)

Em vez de usar variáveis simples separadas, vamos utilizar uma variável composta.

```
int notas[3] = {8, 7, 9};
```

Vantagens:

- Código mais organizado: Mais fácil de manter (adaptar, modificar).
- Facilidade de processamento: Sem gerenciamento manual.
- Uso eficiente de memória: Acesso Rápido

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Array (ou Vetor)

Arrays são coleções de elementos do **mesmo tipo de dado**, armazenados em posições de memória sequenciais (contíguas).

```
tipo nome[tamanho];
```

Exemplo:

```
int notas[3];
```

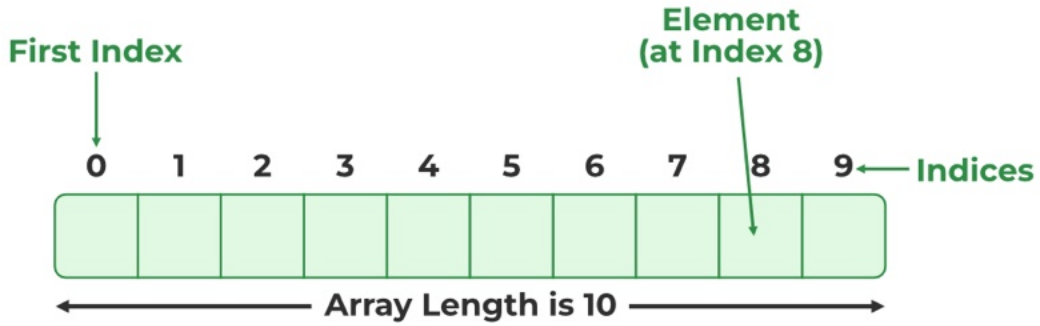
- **int**: tipo dos elementos
- **notas**: nome do vetor (variável)
- **[3]**: quantidade de elementos

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Acessando Elementos e Indexação Zero-Base

O acesso aos elementos é feito por meio de índices inteiros.

- A contagem de índices em C **sempre começa em zero (0)**.
- Em um array de tamanho N, os índices válidos são: $0, 1, 2, \ldots, N-1$.



> **Observação**: Em C, o tamanho de um array deve ser conhecido em tempo de compilação (compile-time constant).
>
> Fonte da imagem: GeeksForGeeks.

Exemplo

```c
#include <stdio.h>

int main() {

    int idade[3];

    idade[0] = 18;
    idade[1] = 20;
    idade[2] = 25;

    printf("%d\n", idade[0]);
    printf("%d\n", idade[1]);
    printf("%d\n", idade[2]);

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Inicialização

Inicialização Direta:

```
int valores[5] = {1, 2, 3, 4, 5};
```

Inicialização Parcial:

```
int numeros[5] = {10, 20};
```

> Na inicialização parcial acima os demais valores são considerados zero: **10 20 0 0 0**

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exemplo

```c
int numeros[4] = {10, 20, 30, 40}; // Tamanho 4. Índices de 0 a 3.

printf("Primeiro elemento (Índice 0): %d\n", numeros[0]); // Imprime 10
printf("Último elemento (Índice 3): %d\n", numeros[3]); // Imprime 40

// ATENÇÃO!
// printf("%d", numeros[4]); // ERRO: Índice fora dos limites (Out-of-Bounds)
```

> **Atenção**: Nunca acesse um índice que exceda o tamanho declarado do array. Isso leva a **Comportamento Indefinido (Undefined Behavior)**, corrompendo a memória e causando falhas imprevisíveis.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Inferência de Tamanho

```
int x[] = {1, 2, 3};
```

O compilador calcula automaticamente o tamanho.

> O compilador ele executa uma contagem estática dos valores literais fornecidos e ajusta o metadata do array para garantir que o bloco de memória seja grande o suficiente (e somente o necessário).

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício

Crie um programa que:

1. Inicializa um vetor de tamanho 5 com valores: 1, 0, 0, 0, 2.
2. Imprime o primeiro e o último elementos.
3. Modifica o valor do elemento de índice 3 para 8.
4. Imprime o valor do elemento de índice 3.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Percorrendo Vetores com Loop

A maneira mais segura de processar todos os elementos é usando laços de repetição (**for** ou **while**).

```c
#include <stdio.h>
#define TAMANHO 5

int main() {
    int numeros[TAMANHO] = {10, 20, 30, 40, 50};
    for(int i = 0; i < TAMANHO; i++) {
        printf("%d\n", numeros[i]);
    }

    return 0;
}
```

> **Dica**: Usar constantes (**#define** ou **const int**) para definir o tamanho do array é uma prática essencial para evitar "números mágicos" (magic numbers).

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício

Modifique o programa abaixo para exibir os elementos do array **ordem reversa**:

```c
#include <stdio.h>
#define TAMANHO 5

int main() {
    int numeros[TAMANHO] = {10, 20, 30, 40, 50};
    for(int i = 0; i < TAMANHO; i++) {
        printf("%d\n", numeros[i]);
    }

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Leitura de Dados do Usuário

```c
#include <stdio.h>

int main() {

    int notas[4];

    for(int i = 0; i < 4; i++) {
        printf("Digite a nota: ");
        scanf("%d", &notas[i]);
    }

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício

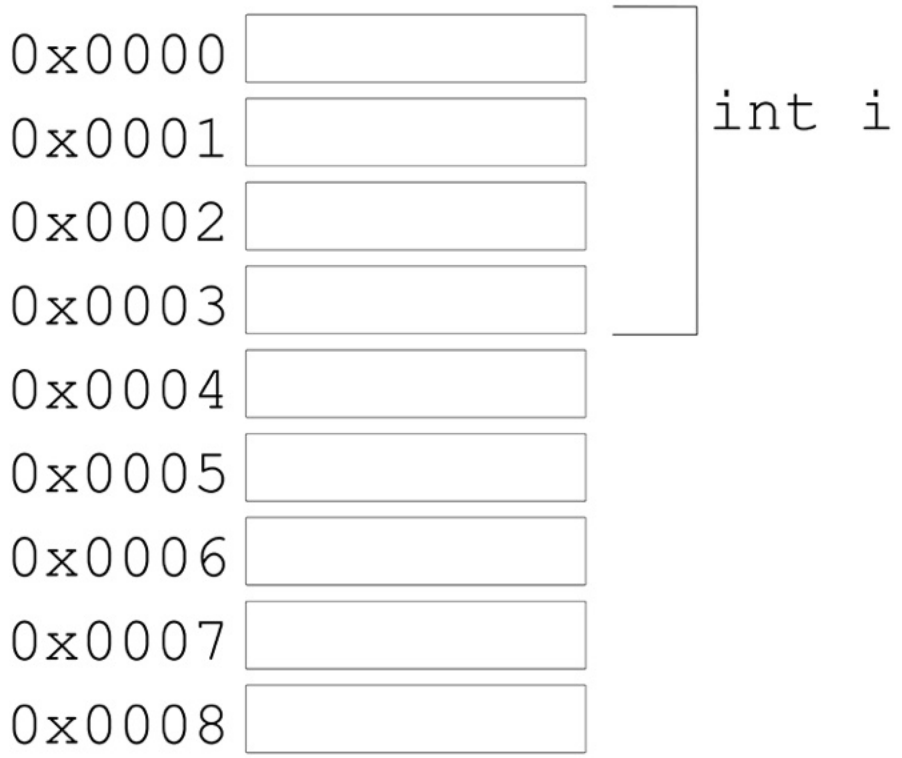
Crie um programa que:

- Leia 10 números inteiros e armazene em um vetor
- Calcule a Soma e a Média

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Lembrando: Variáveis em Memória

- Quando criamos uma variável, estamos alocando espaço na memória.
- Tipos diferentes de variável tomam espaços diferentes na memória. Exemplo:
  - **int**: 4 bytes
  - **double**: 8 bytes

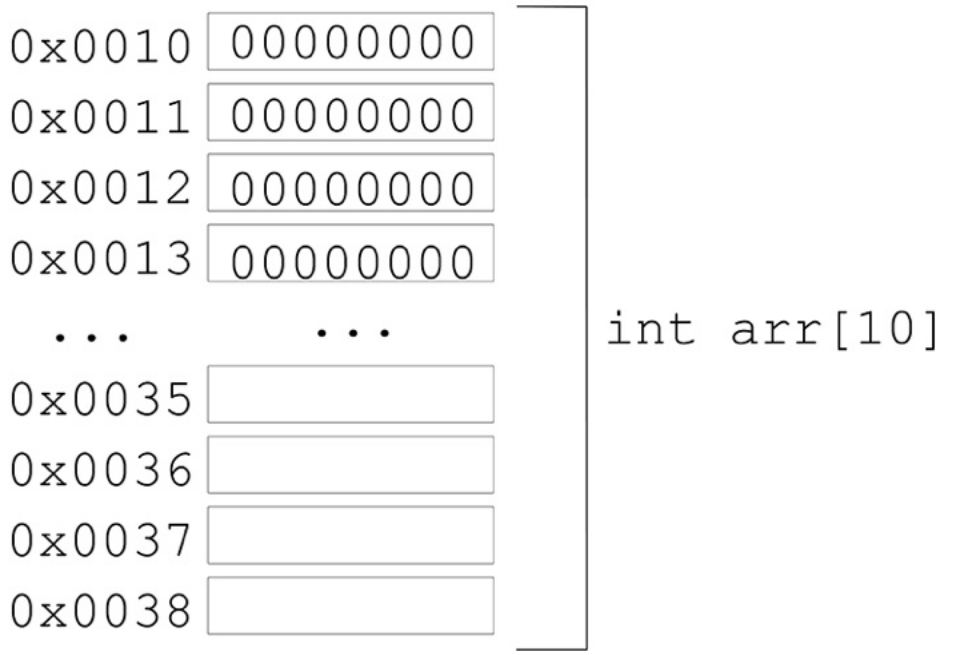


> Fonte da imagem: openframeworks.cc

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Array em Memória

- Um array, por definição, é um bloco de memória contíguo.
- Todos os elementos do array ocupam posições de endereços de memória adjacentes (um após o outro).
- Sem nenhum espaço vazio entre eles.



> Fonte da imagem: openframeworks.cc

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Tamanho de um Vetor: `sizeof`

```c
#include <stdio.h>

int main() {

    int numeros[5];

    printf("Tamanho do array em bytes:\n");
    printf("%lu", sizeof(numeros));

    return 0;
}
```

Se `int` ocupa 4 bytes:

```
5 x 4 = 20 bytes
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Erros Comum: Acesso Fora do Limite

```
int v[3];

v[3] = 10;
```

Índices válidos:

```
0, 1, 2
```

Consequências:

- Comportamento indefinido
- Corrupção de memória

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Procurando um Valor em um Vetor

```c
#include <stdio.h>

int main() {
    int numeros[5] = {10, 20, 30, 40, 50};
    int busca = 30;

    for(int i = 0; i < 5; i++) {

        if(numeros[i] == busca) {
            printf("Encontrado!");
        }
    }

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Encontrando o Maior Valor em um Vetor

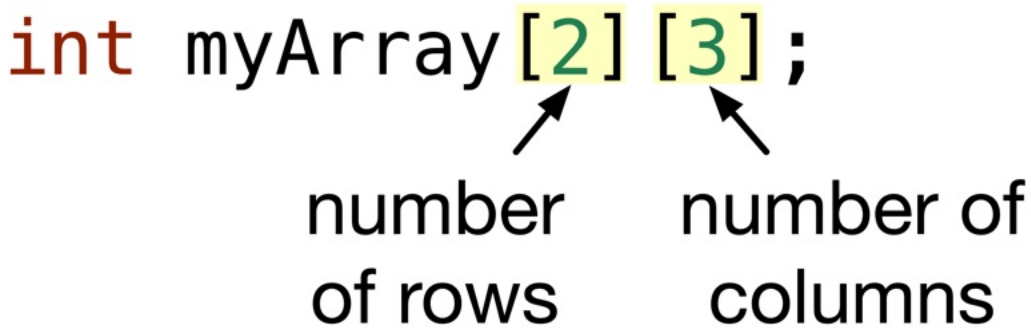
```c
#include <stdio.h>

int main() {
    int numeros[5] = {3, 7, 2, 9, 1};
    int maior = numeros[0];

    for(int i = 1; i < 5; i++) {

        if(numeros[i] > maior) {
            maior = numeros[i];
        }
    }

    printf("Maior = %d", maior);

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Matriz em C

Uma matriz é:

- Um array multidimensional
- Uma estrutura de dados organizada em **linhas** e **colunas**
- Utilizada para representar tabelas, imagens, mapas e grades



Sintaxe Geral

```
tipo nome[linhas][colunas];
```

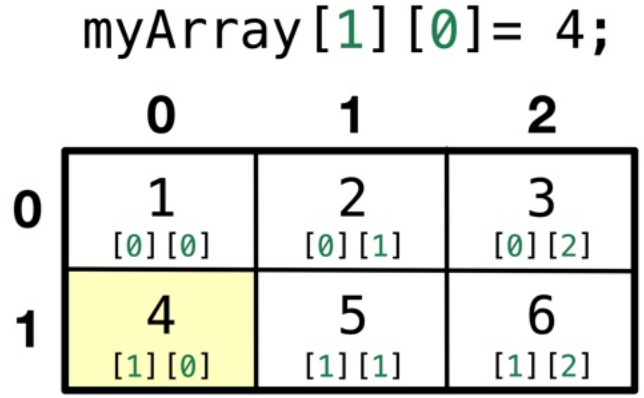
> Fonte da imagem: learningc.org

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Matrizes em C

C não possui "matrizes" como objeto nativo distinto. Ele trata matrizes como arrays de arrays.

```
int myArray[2][3]
myArray[0][0] = 1;
myArray[0][1] = 2;
myArray[0][2] = 3;
myArray[1][0] = 4;
myArray[1][1] = 5;
myArray[1][2] = 6;
```



> Em C, o tamanho da segunda dimensão (colunas) é obrigatório na declaração. O compilador precisa saber o tamanho do bloco para calcular endereços corretamente. Fonte da imagem: learningc.org

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criando e Inicialização de Matrizes

```c
int matrix[2][3]; // Contém valores de "garbage" (indeterminados)
// SEMPRE INICIALIZAR EXPLICITAMENTE EM C!
```

```c
int matrix[2][3] = {
    {1, 2, 3},
    {4, 5, 6}
};
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exemplo

Armazenar notas de:

- 3 alunos
- 2 provas

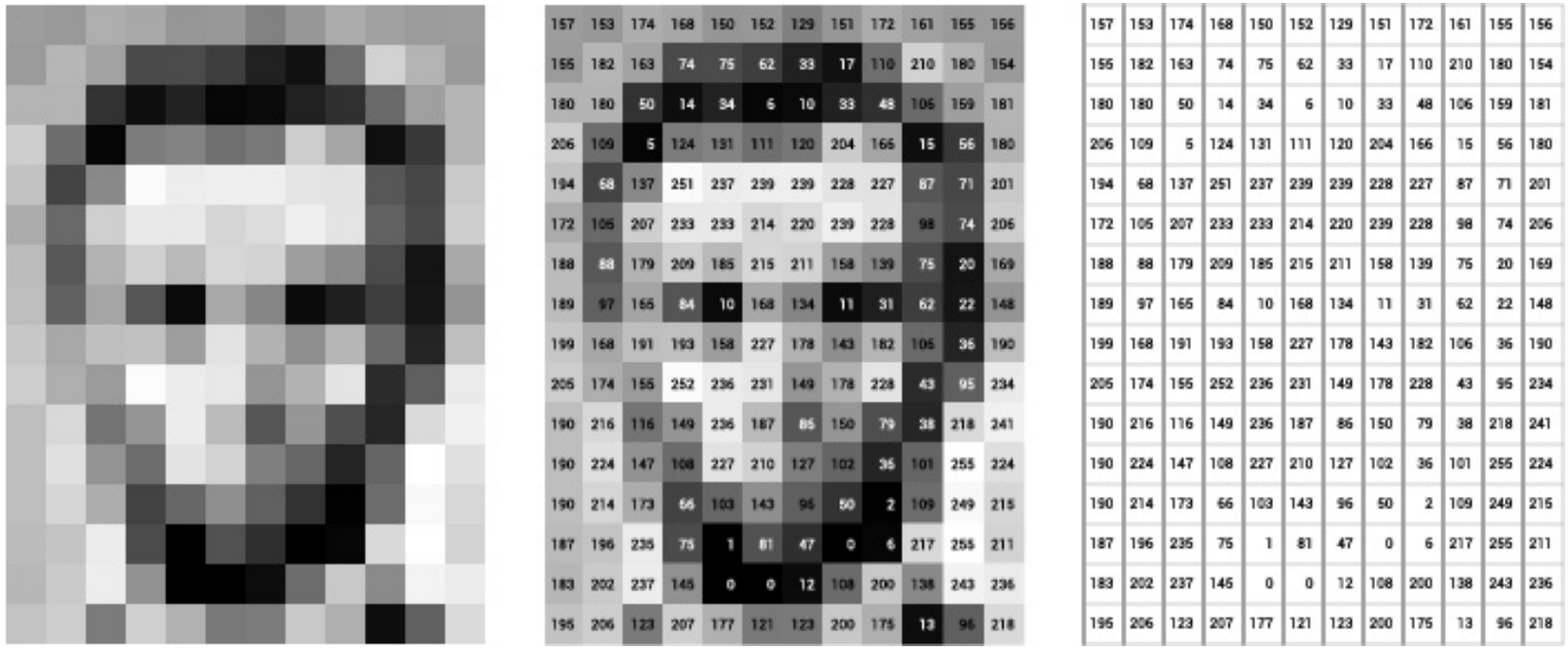
```
float notas[3][2];
```

```
Aluno 0 -> Prova 1 e Prova 2
Aluno 1 -> Prova 1 e Prova 2
Aluno 2 -> Prova 1 e Prova 2
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Matrizes em Processamento de Imagens

Matrix $M_{m \times n} \in \mathbb{Z}_0^{L-1}$, onde $L = 2^k$ e k é o número de bits por pixel.



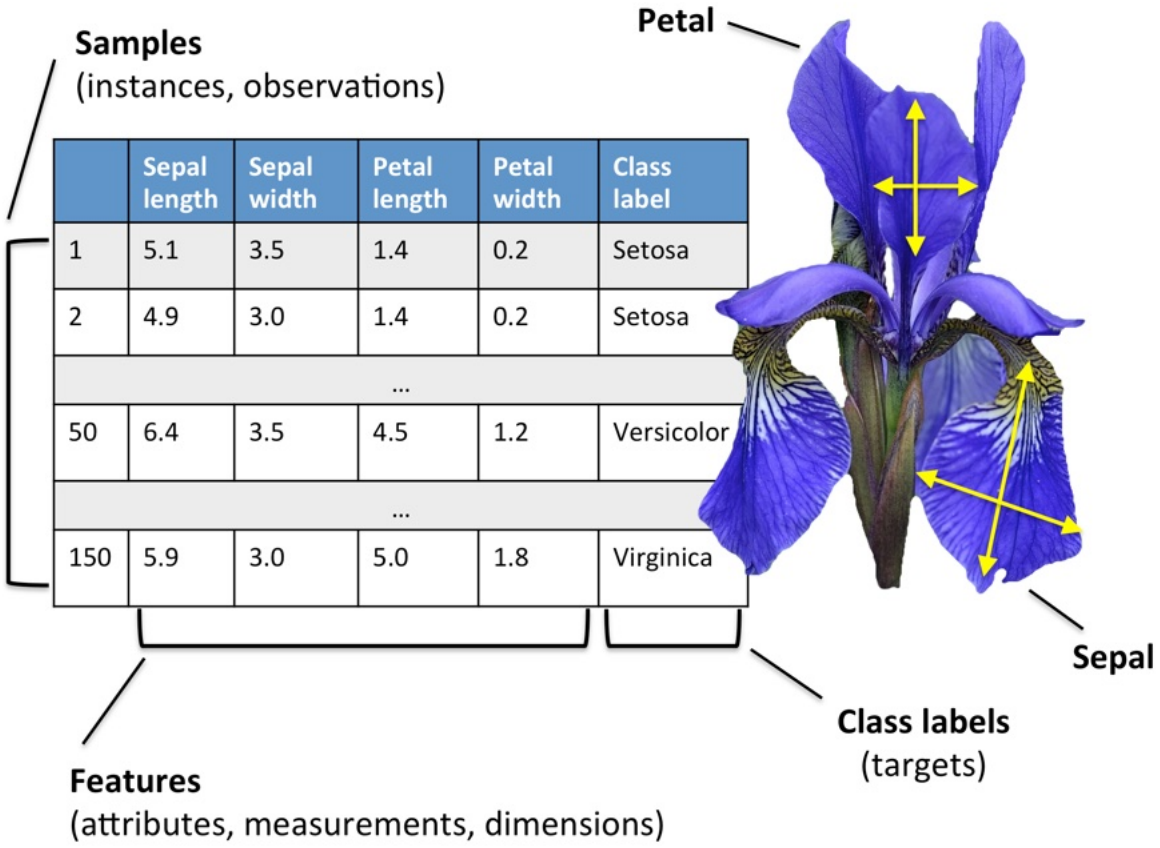
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

> Na imagem: Representação de pixels de uma imagem. Fonte: OpenFrameworks - Image Processing.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Matrizes em Machine Learning



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

> Fonte da imagem: Sebastian Raschka

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Navegando na Matriz

```c
int matrix[2][3] = {
    {1, 2, 3},
    {4, 5, 6}
};
```

Acesso Direto:

```c
printf("%d", matriz[0][1]); // Imprime o elemento da linha 0, coluna 1
```

Iteração (Loop Aninhado):

```c
for (int i = 0; i < 2; i++) {       // Percorre Linhas
    for (int j = 0; j < 3; j++) {  // Percorre Colunas
        printf("%d ", matriz[i][j]);
    }
    printf("\n"); // Quebra linha após cada linha da matriz
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exemplo: Soma de Elementos

```c
#include <stdio.h>

int main() {
    int matriz[2][2] = {
        {1,2},
        {3,4}
    };
    int soma = 0;
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            soma += matriz[i][j];
        }
    }
    printf("Soma = %d", soma);
    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exemplo: Lendo Matriz do Teclado

```c
#include <stdio.h>

#define TAMANHO_LINHAS 3
#define TAMANHO_COLUNAS 3

int main() {
    int matriz[TAMANHO_LINHAS][TAMANHO_COLUNAS];
    printf("Por favor, insira os 9 valores da matriz:\n\n");
    for (int i = 0; i < TAMANHO_LINHAS; i++) {
        // A cada nova linha, pedimos ao usuário para preencher os valores da coluna.
        printf("Insira os %d valores da Linha %d:\n", TAMANHO_COLUNAS, i + 1);
        for (int j = 0; j < TAMANHO_COLUNAS; j++) {
            printf("  Elemento [%d][%d]: ", i, j);
            scanf("%d", &matriz[i][j]);
        }
    }
    printf("\nLeitura da matriz concluída com sucesso!\n");

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício A

Crie um programa que leia uma matriz 4x4 imprima:

- maior valor
- menor valor
- soma da diagonal principal

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício B

Crie um programa que inicializa duas matrizes 5x3 e pede que o usuário selecione uma das opções abaixo:

- **Opção 1**: Calcular a soma das matrizes
- **Opção 2**: Calcular a subtração das matrizes

O programa deve exibir o resultado da operação selecionada.

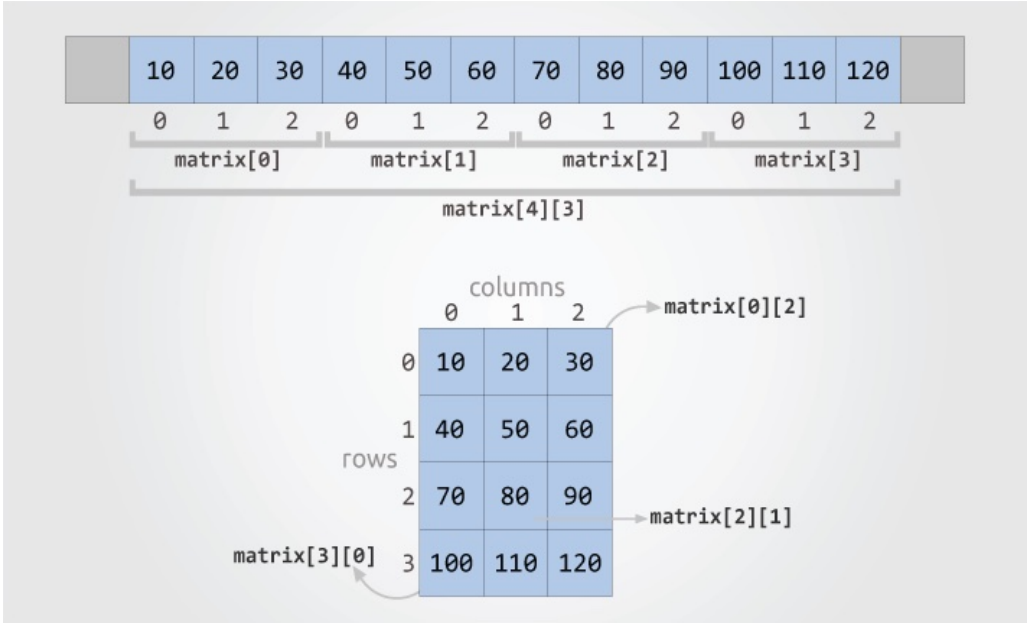
Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Matriz em Memória

Row-Major Order: C armazena matrizes na ordem das linhas.

- Todas as colunas da linha 0 vêm primeiro.
- Depois, todas as colunas da linha 1.

```
Endereço Base: &mat[0][0]

| [0][0] | [0][1] | [0][2] | <- Linha 0 (4 bytes * 3)
+------------------------+
| [1][0] | [1][1] | [1][2] | <- Linha 1
```



> Fonte da Imagem: codeforwin.org.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Conclusão

Arrays são fundamentais na programação em C porque:

- Permitem armazenar múltiplos dados
- Tornam algoritmos mais eficientes

Matrizes permitem representar dados bidimensionais

- Amplamente utilizadas em computação científica
- Servem como base para: imagens, simulações, jogos, IA

Próximo Tópico: Strings em C

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Dúvidas e Discussão

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício e Questões

Questão 1

Um array **float temperaturas[7]** foi declarado. Se um programador tenta acessar o elemento na posição **temperaturas[8]**, qual é a consequência mais provável, do ponto de vista da segurança em C?

A) O programa irá compilar com erro e não rodará.
B) O compilador emitirá um aviso (warning), mas o programa funcionará corretamente.
C) O programa tentará ler/escrever fora dos limites alocados, resultando em Comportamento Indefinido (Undefined Behavior).
D) O sistema operacional irá travar imediatamente.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Questão 2

Qual é o erro no código abaixo?

```
int v[5];

v[5] = 10;
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Questão 3

Qual das alternativas representa corretamente uma matriz 3x4 em C?

A)

```
int matriz[3,4];
```

B)

```
int matriz[3][4];
```

C)

```
int matriz(3)(4);
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Questão 4

Qual é o erro abaixo?

```c
int matriz[2][2];

matriz[2][0] = 5;
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 1

Crie um programa que:

- Leia um vetor com 8 posições
- Conte quantos números são pares
- Conte quantos números são ímpares

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 2

Crie um programa que:

- Leia 10 números
- Armazene em um vetor
- Ordene os números em ordem crescente
- Exiba o vetor ordenado

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 3

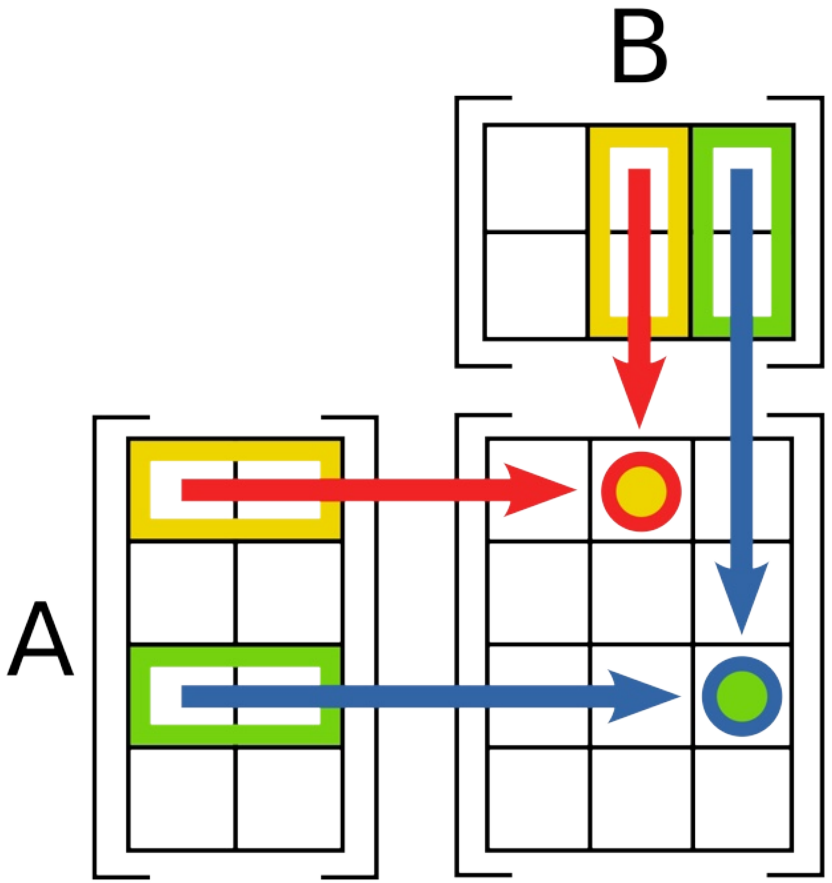
Crie um programa que inicialize uma matriz 4 x 3 e imprima a matriz transposta.

> Transposta: Dado uma matriz M de $C \times R$, a matriz transposta N é definida como $N[i][j] = M[j][i]$.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 4

Crie um programa que calcule e imprime a multiplicação de duas matrizes. Se as matrizes forem incompatíveis, o programa deve exibir uma mensagem de erro.



> Fonte da Imagem: Wikipedia.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 5

Crie um programa que lê uma matriz quadrada do usuário e verifica se ela é simétrica, ou seja $M[i][j] == M[j][i]$ para todo i, j.

O programa deve imprimir a mensagem "SIM", se a matriz for simétrica. Senão, deve imprimir "NÃO".

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br