Vetores e Matrizes

Programação e Desenvolvimento de software I
Heitor Ramos

Alocação Estática de Memória

- Declaração de variáveis
- Espaço de memória suficiente é alocado

```
char c;     // 1 byte
short a;    // 2 bytes
int a;     // 4 bytes (depende do gcc na
verdade)
long b;     // 8 bytes
float x;    // 4 bytes
double y;   // 8 bytes
```

Problema 1: Como ler três notas com scanf

Problema 1: Como ler três notas com scanf

```c
#include <stdio.h>
int main(void) {
  int nota1;
  int nota2;
  int nota3;

  printf("Digite 3 notas separadas por espaço: ");
  scanf("%d %d %d", &nota1, &nota2, &nota3);

  return 0;
}
```

Problema 2: Como ler 100 notas com scanf

Problema 2: Como ler 100 notas com scanf

```c
#include <stdio.h>
int main(void) {
  int nota1;
  int nota2;
  int nota3;
  int nota3;
  // . . .
  int nota100;

  printf("Digite 100 notas separadas por espaço: ");
  // . . . scanf gigante!

  return 0;
}
```

Problema 2.1: Como pegar a nota do aluno 76

```
// . . .
if (num_aluno == 1) {
  printf(nota1);
} else if (num_aluno == 2) {
  printf(nota2);
// . . .
} else if (num_aluno == 75) {
  printf(nota75);
} else if (num_aluno == 76) {
  printf(nota76);
}
// . . .
```

vetores

Vetores

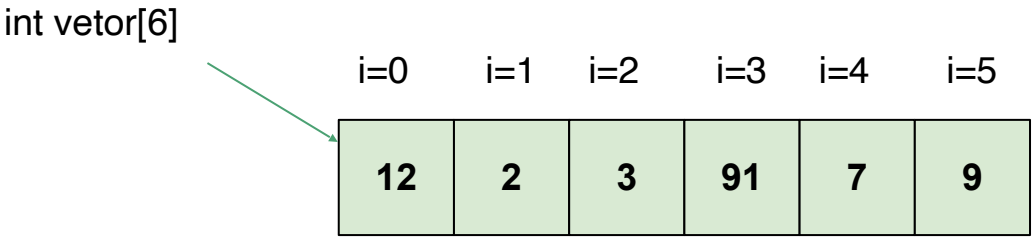
- Coleção de variáveis do mesmo tipo referenciada por um nome em comum
  - Acesso por meio de um índice
  - Posições contíguas na memória
  - Tamanho pré-definido
  - Índices fora dos limites pode causar comportamento anômalo no código
- Declaração de um vetor:
  - <tipo> identificador [<número de posições>];
  - A primeira posição de um vetor tem índice 0
  - A última posição de um vetor tem índice <número de posições> - 1

Vetores

- Aloca diversas variáveis
- Indexado por inteiros
- O valor entre chaves indica quantas vezes o espaço vai ser alocado
    - De forma simples, o número de elementos no vetor

```
int v[100];    // 100 * sizeof(int) = 400 bytes
long vl[100];   // 100 * sizeof(long) = 800 bytes
double z[100];  // 100 * sizeof(double) = 800 bytes

int vetor[3] = {15, 22, 7};
int vetor[] = {13, 98, 125};
```

Vetores na memória



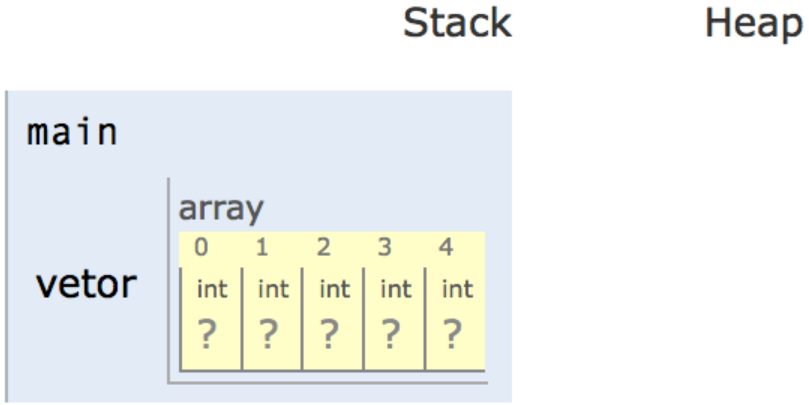
- Todo o vetor é alocado de forma contígua na memória
- Alocação estática
  - Mora na região do Stack
  - Vamos diferenciar o Stack e o Heap em outras aulas

Vetores

```c
#include <stdio.h>
int main(void) {
  double z[100];
  z[0] = 7.2;      // o primeiro índice é 0
  z[1] = 6.21;
  // . . .
  z[99] = 9882.2;
  printf("O segundo valor é %.3lf\n", z[1]);
  return 0;
}
```

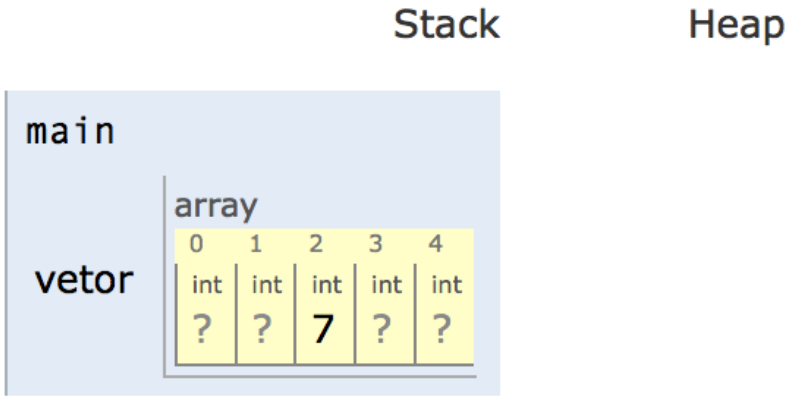
Índices

```c
#include <stdio.h>
int main(void) {
  int vetor[5];
  vetor[2] = 7;
  vetor[4] = 9;
  return 0;
}
```



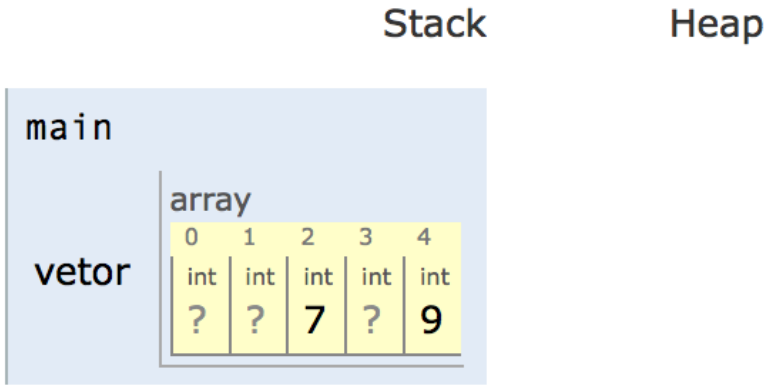
Índices

```c
#include <stdio.h>
int main(void) {
  int vetor[5];
  vetor[2] = 7;
  vetor[4] = 9;
  return 0;
}
```



Índices

```c
#include <stdio.h>
int main(void) {
  int vetor[5];
  vetor[2] = 7;
  vetor[4] = 9;
  return 0;
}
```



Índices

- Indicam a posição de memória onde buscar o valor
- Lembre-se também de inicializar os vetores

```
int num_elements = 1000;
float x[num_elements];

//Iniciando o vetor (boa prática)
int i = 0;
for (i = 0; i < num_elements; i++) {
        x[i] = i * 3.0;
}

int valor = x[20];

// x[20] está na posição x+20*sizeof(float)
// Qual número está armazenado em valor?
```

Erros!

- **O compilador C não vai avisar se você está acessando uma região errada!**
- Um erro em tempo de execução vai ocorrer

```
float x[1000];
// … código aqui inicializando etc...
float y = x[2000]; // não dá erro de
   compilação
```

- Seu trabalho é garantir que tais erros não ocorram!
- Programe bem!
- Erro mais comum é o **segmentation fault**

Exemplo de Erro

```c
#include <stdio.h>
int main(void) {
  double z[3];
  z[3] = 7.2;
  printf("Valor z[3] é %.1f\n", z[3]);
  return 0;
}
```

Voltando ao problema: como ler 100 notas com o scanf?

Problema

Faça um programa que:

1. Lê um número (`n`) que indica o grau de um polinômio.
2. Em um segundo passo, vamos ler os `n` coeficientes (a_i)
3. Leia o valor de `x`
4. Por fim, resolva o polinômio

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0 x^0$$

Polinomios

```c
#include <math.h>
#include <stdio.h>
int main(void) {
  // 1. Lendo n
  int n;
  printf("Digite o grau: ");
  scanf("%d", &n);
  // 2. Coeficientes
  int coeficientes[n];
  printf("Digite os coeficientes: ");
  for (int i = 0; i < n; i++)
        scanf("%d", &coeficientes[i]);
  // 3. Lendo x
  int x;
  printf("Digite o valor de x: ");
  scanf("%d", &x);
  // 4. Resultado
  int resultado = 0;
  for (int i = 0; i < n; i++)
        resultado += coeficientes[i] * pow(x, i+1);
  printf("O resultado foi %d\n", resultado);
  return 0;
}
```

Problema: Somar os elementos do vetor

```c
#include <stdio.h>
int main(void) {
  int lista[5];
  int soma = 0;
  for (int i=0; i<5; i++)
      soma = soma + lista[i];
  return 0;
}
```

Problema: encontre o maior valor em um array

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  int i, A[5] = {3,18,2,51,45};
  int ma = A[0];

  for(i=1; i<5; i++){
    if(ma < A[i])
        ma = A[i];
  }

  printf("Maior = %d\n", ma);

  return 0;
}
```

Copiando um array

- Não se pode fazer atribuição de arrays inteiros, apenas de suas posições individualmente

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  int v[5] = {1,2,3,4,5};
  int v1[5];

  v1 = v; //ERRADO!

  int i;
  for(i=0; i<5; i++)
    v1[i] = v[i]; //CORRETO

  return 0;
}
```

Matrizes

Novo problema

Suponha uma turma com 50 alunos matriculados em 4 disciplinas. Como ler as notas de todos os alunos e tirar a média do semestre? Todos os alunos estão nas mesmas disciplinas.

Novo problema

Suponha uma turma com 50 alunos matriculados em 4 disciplinas. Como ler as notas de todos os alunos e tirar a média do semestre? Todos os alunos estão nas mesmas disciplinas.

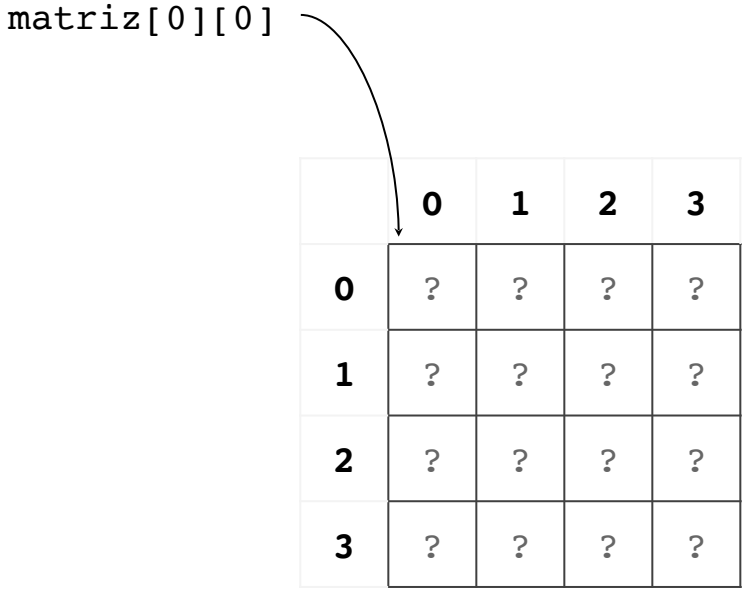
```c
#include <stdio.h>

int main(void) {
  // Cria 4 vetores para cada
matéria
  int nota_materia_1[50];
  int nota_materia_2[50];
  int nota_materia_3[50];
  int nota_materia_4[50];
  return 0;
}
```

Aumentando o problema

E se forem 20 práticas de uma turma de 100 alunos?!
Ou o histórico de todas das disciplinas?!
Problema sem fim!

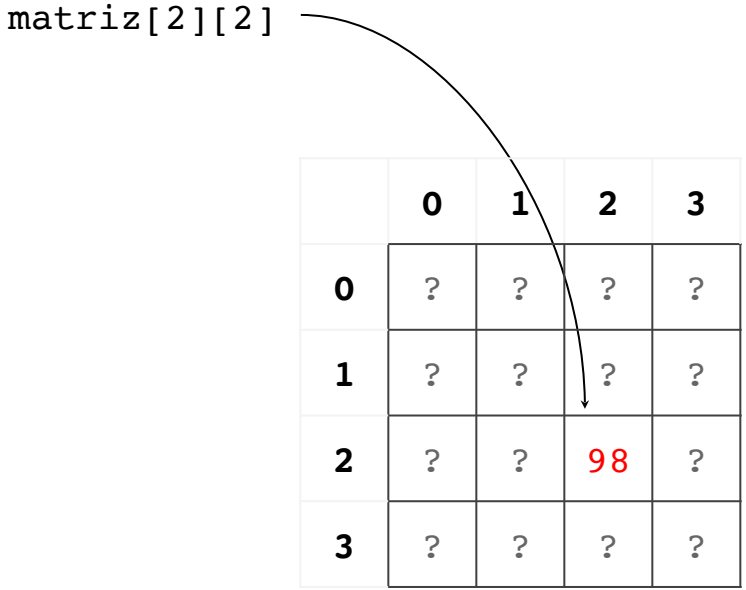
Matrizes

```c
#include <stdio.h>
int main(void) {
  // tipo nome[dim1]
[dim2]
  int matriz[4][4];
  // . . .
  // indexando
  matriz[2][2] = 98;
  // . . .
  return 0;
}
```



Matrizes

```c
#include <stdio.h>
int main(void) {
  // tipo nome[dim1]
[dim2]
  int matriz[4][4];
  // . . .
  // indexando
  matriz[2][2] = 98;
  // . . .
  return 0;
}
```

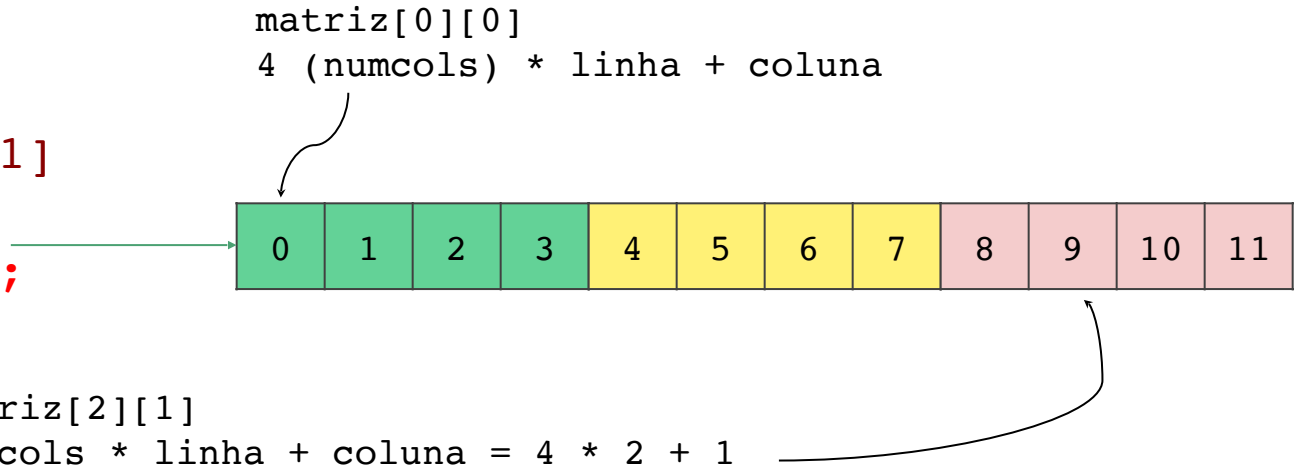


Olhando para o baixo nível

Na memória, uma matriz é bem similar a um vetor.
A linguagem C (e outras linguagens) apenas oculta este detalhe para os programadores.

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

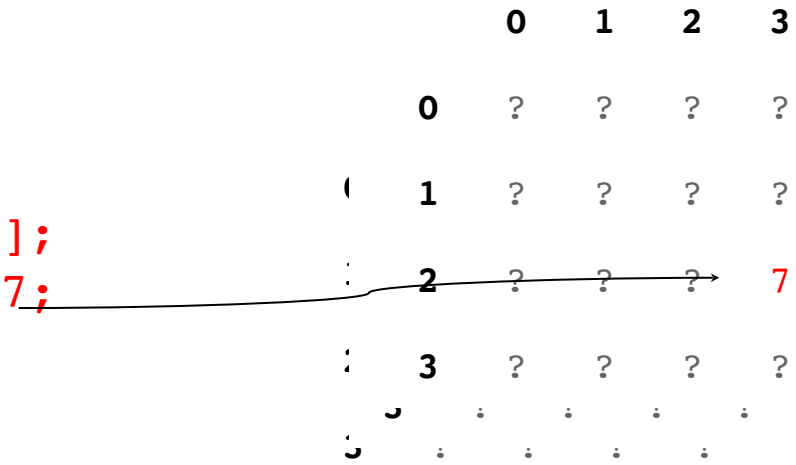
```c
#include <stdio.h>
int main(void) {
  // tipo nome[dim1]
[dim2]
  int matriz[3][4];
  return 0;
}
```



Mais dimensões

Você pode declarar quantas dimensões quiser. Chamamos de vetores/arrays n-dimensionais ou tensores.

```c
#include <stdio.h>
int main(void) {
  int tensor[3][4][4];
  tensor[2][2][3] = 7;
  return 0;
}
```



Problema 1

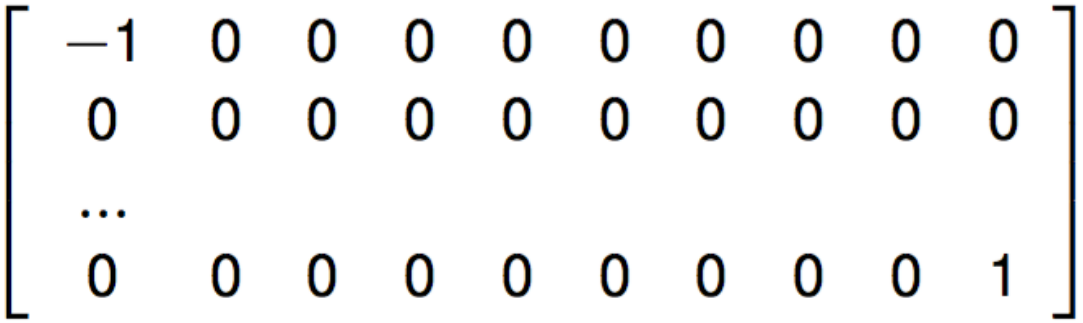
Faça um programa onde o usuário indica o número de linhas e colunas de uma matriz. Após definir tais valores, preencha uma matriz do tamanho indicado com números lidos com scanf. Por fim, desenhe a matriz na tela

Lendo e imprimindo matrizes

```c
#include <stdio.h>
int main(void) {
  int nlinhas;
  int ncols;
  printf("Digite nlinhas e ncols: ");
  scanf("%d %d", &nlinhas, &ncols);
  // Lê a matriz
  int matriz[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++) {
        for (int c = 0; c < ncols; c++) {
          printf("Digite a posicao %d %d\n", l, c);
          scanf("%d", &matriz[l][c]);
        }
  }
  // Imprime na forma retangular
  for (int l = 0; l < nlinhas; l++) {
        for (int c = 0; c < ncols; c++) {
          printf("%3d ", matriz[l][c]);
        }
        printf("\n");
  }
  return 0;
}
```

Problema 2

Escreva um programa que inicializa uma matriz 10 × 10 com 0s em todas as posições. O usuário irá digitar o índice da linha e o índice da coluna e em seguida o valor das posições não nulas. A leitura será feita enquanto os índices forem não negativos. Após a leitura imprima a matriz na tela.

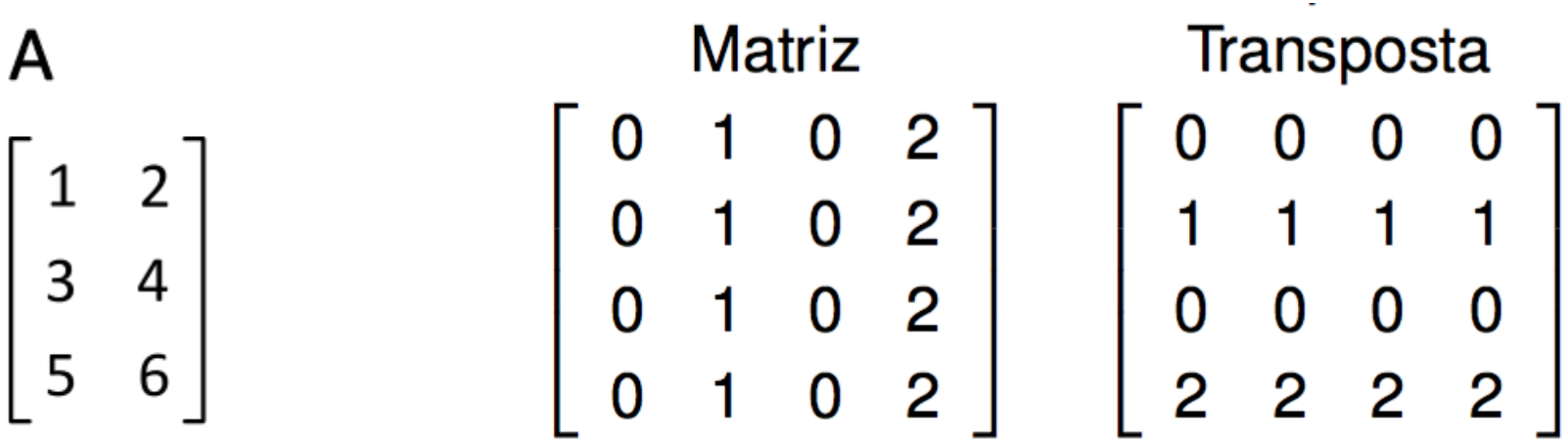



```
0 0 -1
1 2 2
9 9 1
-1 -1
```

```c
#include <stdio.h>
int main(void) {
  int nlinhas=10;
  int ncols=10;
  // printf("Digite nlinhas e ncols: ");
  // scanf("%d %d", &nlinhas, &ncols);
  // zerar a matriz
  int matriz[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++) {
    for (int c = 0; c < ncols; c++) {
        matriz[l][c] = 0;
    }
  }
  int i1=0, i2=0, v;
  while (i1>=0 && i2>=0)
  {
    printf("Digite os índices e o valor: ");
    scanf("%d %d %d",&i1, &i2, &v);
    matriz[i1][i2] = v;
  }

  // Imprime na forma retangular
  for (int l = 0; l < nlinhas; l++) {
  for (int c = 0; c < ncols; c++) {
      printf("%3d ", matriz[l][c]);
  }
  printf("\n");
  }
  return 0;
}
```

Problema 3

Escreva um programa que lê todos os elementos de uma matriz e mostra a matriz e a sua transposta na tela. O tamanho da matriz será indicado pelo usuário.



```c
#include <stdio.h>
int main(void) {
  int nlinhas;
  int ncols;
  printf("Digite nlinhas e ncols: ");
  scanf("%d %d", &nlinhas, &ncols);
  // Lê a matriz
  int matriz[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++) {
    for (int c = 0; c < ncols; c++) {
      printf("Digite a posicao %d %d\n", l, c);
      scanf("%d", &matriz[l][c]);
    }
  }
  // Imprime na forma retangular
  for (int l = 0; l < nlinhas; l++) {
    for (int c = 0; c < ncols; c++) {
      printf("%3d ", matriz[l][c]);
    }
    printf("\n");
  }
  printf("\n\n");
  // Imprime transposta na forma retangular
  for (int c = 0; c < ncols; c++) {
   for (int l = 0; l < nlinhas; l++) {
      printf("%3d ", matriz[l][c]);
    }
    printf("\n");
  }
  return 0;
}
```

Problema 4

Escreva um programa que lê 2 matrizes n × m, mostre-as na tela e mostre a soma entre as duas matrizes em seguida.

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$
$$= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

```c
#include <stdio.h>
int main(void) {
  int nlinhas;
  int ncols;
  printf("Digite nlinhas e ncols: ");
  scanf("%d %d", &nlinhas, &ncols);
  // Lê a matriz
  printf("######## Matriz 1 ##########\n");
  int matriz1[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++) {
    for (int c = 0; c < ncols; c++) {
      // printf("Digite a posicao %d %d\n", l, c);
      scanf("%d", &matriz1[l][c]);
    }
  }
  printf("######## Matriz 2 ##########\n");
  int matriz2[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++) {
    for (int c = 0; c < ncols; c++) {
      // printf("Digite a posicao %d %d\n", l, c);
      scanf("%d", &matriz2[l][c]);
    }
  }

  // Imprime na forma retangular
  for (int l = 0; l < nlinhas; l++) {
    for (int c = 0; c < ncols; c++) {
      printf("%3d ", matriz1[l][c] + matriz2[l][c]);
    }
    printf("\n");
  }

  return 0;
}
```

Problema 5

Escreva um programa que lê 2 matrizes n × n (mesmo número de linhas e colunas), mostre-as na tela e mostre **o produto** entre as duas matrizes em seguida.
Dica, o produto entre duas matrizes pode ser calculado com 3 laços for aninhados (um dentro do outro). Faça no papel primeiro.

$$\mathbf{AB} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} \alpha & \beta & \gamma \\ \lambda & \mu & \nu \\ \rho & \sigma & \tau \end{pmatrix} = \begin{pmatrix} a\alpha + b\lambda + c\rho & a\beta + b\mu + c\sigma & a\gamma + b\nu + c\tau \\ p\alpha + q\lambda + r\rho & p\beta + q\mu + r\sigma & p\gamma + q\nu + r\tau \\ u\alpha + v\lambda + w\rho & u\beta + v\mu + w\sigma & u\gamma + v\nu + w\tau \end{pmatrix},$$

```c
#include <stdio.h>
int main(void)
{
  int nlinhas;
  int ncols;
  printf("Digite nlinhas e ncols (matriz quadrada): ");
  scanf("%d %d", &nlinhas, &ncols);
  // Lê a matriz
  printf("######## Matriz 1 ##########\n");
  int matriz1[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++)
  {
    for (int c = 0; c < ncols; c++)
    {
      printf("Digite a posicao %d %d\n", l, c);
      scanf("%d", &matriz1[l][c]);
    }
  }
  printf("######## Matriz 2 ##########\n");
  int matriz2[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++)
  {
    for (int c = 0; c < ncols; c++)
    {
      printf("Digite a posicao %d %d\n", l, c);
      scanf("%d", &matriz2[l][c]);
    }
  }
  int matriz3[nlinhas][ncols];
  for (int l = 0; l < nlinhas; l++)
    for (int c = 0; c < ncols; c++)
      matriz3[l][c] = 0;

  // produto
  for (int l = 0; l < nlinhas; l++)
  {
    for (int c = 0; c < ncols; c++)
    {
      for (int n = 0; n < ncols; n++)
        matriz3[l][c] += matriz1[l][n] * matriz2[n][c];
    }
    printf("\n");
```

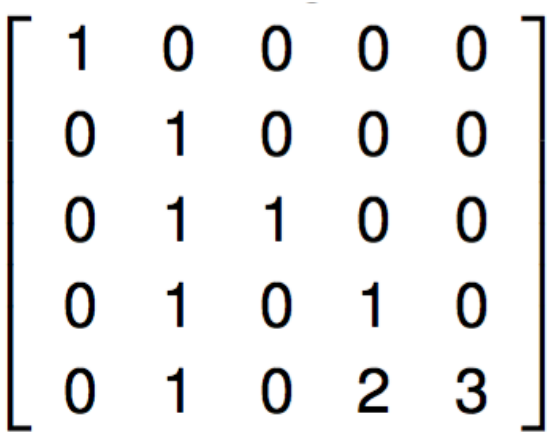
Problema 6

Escreva um programa lê uma matriz e depois verifica se esta é uma matriz triangular inferior.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}$$

Problema 6

Escreva um programa lê uma matriz e depois verifica se esta é uma matriz triangular inferior.



```
// [0][0] = 1 [0][1] = 0 [0][2] =
0
// [1][0] = 2 [1][1] = 1 [1][2] =
0
// [2][0] = 3 [2][1] = 4 [2][2] =
9

int triang_inf = 1;
for (int c = 1; c < ncols; c++)
  for (int l = c-1; l >= 0; l--)
       if (M[l][c] != 0)
            triang _inf = 0;

if (triang_inf)
  printf("eh triangular
inferior");
```