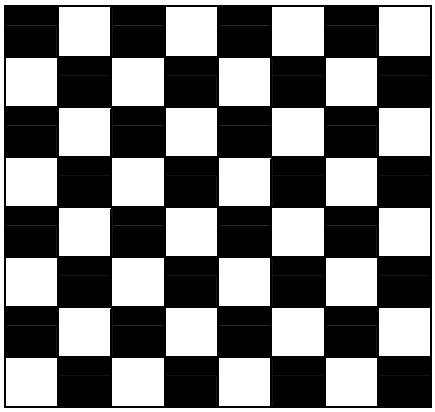
e) *Intercalação* – Receber dois inteiros positivos não nulos *i* e *j* tal que ambos são menores que 20 e correspondem aos tamanhos das duas listas de números a serem fornecidas pelo usuário e alocar todos os números de cada lista em um vetor independente. Ao fornecer estas duas listas, considera-se que o usuário irá digitá-las em ordem crescente. Em seguida, copiar os elementos dos dois vetores para um terceiro também em ordem crescente e imprimir este vetor (note que, como os dois primeiros vetores já estão ordenados, basta intercalar ordenadamente os seus elementos para construir o terceiro vetor ordenado). O resultado na tela deve ser como no exemplo a seguir:

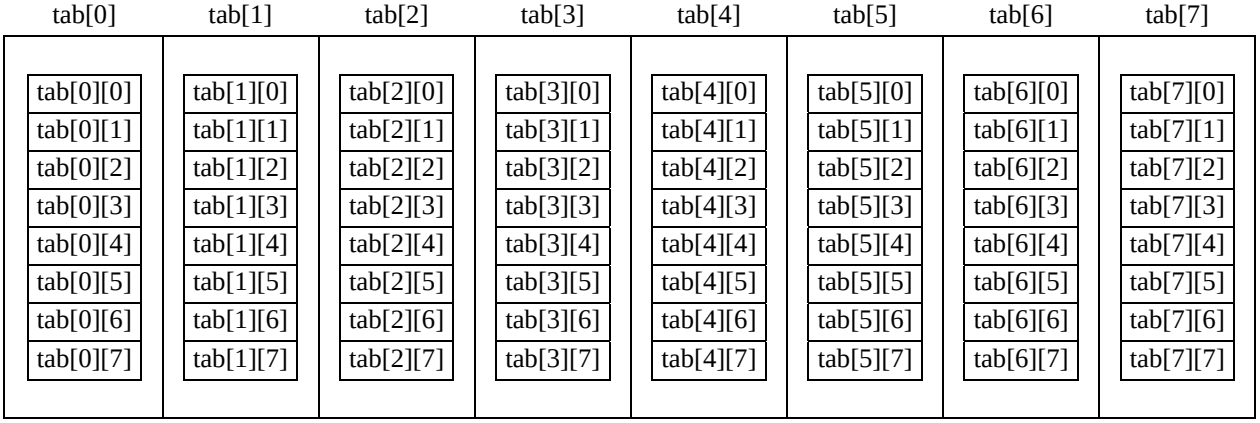
```
Digite o número de elementos do vetor A: 4
Digite o número de elementos do vetor B: 3
Digite cada elemento do vetor A........: 3 7 88 91
Digite cada elemento do vetor B........: 2 4 90
Elementos ordenados no vetor C.........: 2 3 4 7 88 90 91
```

Matriz

- Vetor de vetores de um mesmo tipo de dados.
- *Elemento da matriz* – pode ser:
  - um vetor (matriz bi-dimensional)
  - uma matriz (matriz *n*-dimensional onde $n > 2$).
- Exemplo: Uma matriz pode ser usada para representar um tabuleiro de xadrez 8 por 8:



Da seguinte forma: `char tab[8][8];`



- Exemplos de declaração:

```
char teclado[4][10];  // mapeamento das teclas de uma máquina de escrever
float despesas_mensais[12][3]; // gastos de água, luz e telefone por mês
```

- Exemplos de atribuição de dados à matriz:

```
teclado[0][0] = '1';
teclado[1][0] = 'q';
teclado[2][0] = 'a';
teclado[3][0] = 'z';

teclado[0][1] = '2';
teclado[1][1] = 'w';
teclado[2][1] = 's';
teclado[3][1] = 'x';

despesas_mensais[0][0] = 12;        // R$12,00 de água no mês de janeiro
despesas_mensais[9][1] = 40.25;     // R$40,25 de luz no mês de outubro
despesas_mensais[1][2] = 21.3 + 32; // R$54,30 de telefone no mês de fevereiro
```

- Exemplos de acesso aos dados da matriz:

```
for (int linha = 0; linha < 4; linha++) {
   for (int coluna = 0; coluna < 10; coluna++)
      printf("%c ", teclado[linha][coluna]);
   printf("\n");
}
```

- Matrizes podem ser declaradas e inicializadas como no exemplo a seguir.

```
char telefone[4][3] = {'1','2','3','4','5','6','7','8','9','*','0','#'};
```

que é o mesmo que:

```
telefone[0][0] = '1';
telefone[0][1] = '2';
telefone[0][2] = '3';
telefone[1][0] = '4';
telefone[1][1] = '5';
telefone[1][2] = '6';
telefone[2][0] = '7';
telefone[2][1] = '8';
telefone[2][2] = '9';
telefone[3][0] = '*';
telefone[3][1] = '0';
telefone[3][2] = '#';
```

Exercício: Fazer um programa em linguagem C para cada um dos seguintes problemas:

a) *Multiplicação de matrizes* – Receber as dimensões e elementos de duas matrizes A e B (assuma que o usuário não fornecerá matrizes de dimensões maiores que 100 por 100). Em seguida, armazenar em outra matriz C o resultado da multiplicação entre elas e imprimir esta matriz. Lembre que a multiplicação entre duas matrizes $A_{x,y}$ e $B_{z,t}$ só é possível quando $y = z$.

```
#include <stdio.h>

int main(int argc, char *argv[])
{
   int matriz_a[100][100], matriz_b[100][100], matriz_c[100][100];
   int lin_a, lin_b, lin_c, col_a, col_b, col_c, total_somas, soma;
   int passo1, passo2, passo3;

   printf("\nDigite as dimensoes da matriz A: ");
   scanf("%d %d", &lin_a, &col_a);
   for (passo1 = 0; passo1 < lin_a; passo1++) {
      for (passo2 = 0; passo2 < col_a; passo2++) {
         printf("A[%d][%d] = ", passo1, passo2);
         scanf("%d", &matriz_a[passo1][passo2]);
      }
   }
   printf("\nDigite as dimensoes da matriz B: ");
   scanf("%d %d", &lin_b, &col_b);
   if (col_a == lin_b) {
      for (passo1 = 0; passo1 < lin_b; passo1++) {
         for (passo2 = 0; passo2 < col_b; passo2++) {
            printf("B[%d][%d] = ", passo1, passo2);
            scanf("%d", &matriz_b[passo1][passo2]);
         }
      }
      printf("\nA x B = \n");
      lin_c = lin_a;
      col_c = col_b;
      total_somas = col_a;
      for (passo1 = 0; passo1 < lin_c; passo1++) {
         for (passo2 = 0; passo2 < col_c; passo2++) {
            soma = 0;
            for (passo3 = 0; passo3 < total_somas; passo3++)
               soma += matriz_a[passo1][passo3] * matriz_b[passo3][passo2];
            matriz_c[passo1][passo2] = soma;
            printf("%d ", matriz_c[passo1][passo2]);
         }
         printf("\n");
      }
   }
   else
      printf("\nNão existe a multiplicação destas matrizes");

   return 0;
}
```

b) *Matrizes esparsas* – Uma matriz que possui mais zeros do que outros valores é chamada de matriz esparsa. Normalmente, uma matriz esparsa é representada de uma forma diferente de matrizes "normais". Em matrizes não esparsas, armazenamos todos os valores, sejam eles zeros ou não. Em matrizes esparsas, registramos apenas os locais onde há um valor diferente de zero. Para fazer isso, usamos uma outra matriz que armazena, em cada linha, a tripla (*x*, *y*, *valor*), como no exemplo a seguir:

Matriz esparsa					
Índices	**1**	**2**	**3**	**4**	**5**
1	0	3	0	0	0
2	0	0	0	0	0
3	0	-1	0	0	0
4	7	0	0	0	0

Nova representação			
Índices	**1 (linha)**	**2 (coluna)**	**3 (valor)**
1	1	2	3
2	3	2	-1
3	4	1	7

Faça um programa que gere, a partir de um matriz esparsa de tamanho e valores fornecidos pelo usuário, a representação mais curta dessa matriz, conforme mostrado acima. Faça o programa mostrar essa nova matriz. Se logo no início, quando o usuário informar as dimensões da matriz e estas forem maiores do que 100 por 100, interrompa o procedimento e apresente uma saída com o número "0" (zero). O programa deve usar duas matrizes: uma para armazenar a representação original e a outra para a nova representação. A seguir tem-se as entradas fornecidas para o exemplo anterior e o resultado a ser apresentado na tela:

```
4 5  ──────────► Entrada: Dimensão (linha e coluna) da matriz esparsa
0 3 0 0 0 ─────► Entrada: Valores da 1ª linha da matriz esparsa
0 0 0 0 0 ─────► Entrada: Valores da 2ª linha da matriz esparsa
0 –1 0 0 0 ────► Entrada: Valores da 3ª linha da matriz esparsa
7 0 0 0 0 ─────► Entrada: Valores da 4ª linha da matriz esparsa
1 2 3 ─────────► Saída:   Valores da 1ª linha da matriz reduzida
3 2 –1 ────────► Saída:   Valores da 2ª linha da matriz reduzida
4 1 7 ─────────► Saída:   Valores da 3ª linha da matriz reduzida
```

c) *Jogo da velha* – Implemente o jogo da velha para 2 jogadores humanos. O jogador deve digitar as coordenadas do "O" (ou do "X") a ser colocado no tabuleiro. O programa deve verificar a cada jogada, da maneira menos braçal possível, se algum jogador conseguiu vencer ou se o jogo acabou em empate. O tabuleiro deve ser impresso novamente a cada nova jogada. Não permita que o jogador tente jogar em uma posição já ocupada do tabuleiro.