Introdução à Programação

Programação C

Arrays e Matrizes

Prof. Roberto M. de Faria/UASC/UFCG

Conteúdo

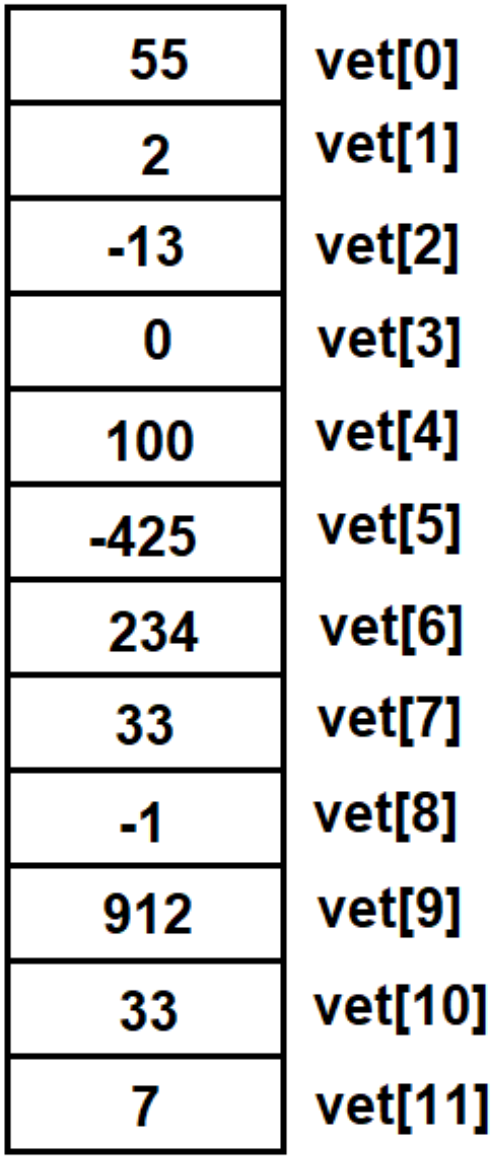
- Introdução
- Arrays
- Declaração de Arrays
- Exemplos de Uso de Arrays
- Passagem de Arrays para Funções
- Arrays de Caracteres
- Ordenação Arrays
- Busca em Arrays
- Arrays com Múltiplos Subscritos
- Passagem de Matriz para Função

Arrays

- Estruturas de itens de dados relacionados – elementos
- Mesmo ***nome***
- Mesmo ***tipo***
- Elementos individualizados por um ***índice***
- Grupo de locações consecutivas de memória
- Estrutura estática – tamanho constante ao longo de todo o programa
- Estruturas de dados dinâmicas – não estudadas aqui

Array

- Exemplo
  - Nome do array – ***vet***
  - Número de elementos (***n***) – ***12***
  - O índice entre colchetes indica a posição do elemento no array
  - O índice do primeiro elemento sempre será ***0***
  - O índice do último elemento sempre será ***n–1***
  - Elementos de array são semelhantes à variáveis



Arrays

- Declaração de arrays
  - Tipo
  - Nome
  - Número de elementos

  ```
  tipo_array nome_array[número_elementos];
  ```

- Exemplos

  ```
  int c[10];
  ```

  ```
  float meu_array[3284];
  ```

- Declaração de múltiplos arrays do mesmo tipo – formato similar para variáveis simples
- Exemplo

  ```
  int valores[100], vetor[27];
  ```

Arrays

- Omissão do tamanho – determinação a partir dos inicializadores

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

- 5 inicializadores – array com 5 elementos

Passagem de Arrays para Funções

- Passagem de argumentos do tipo array – especificação do nome do array (sem colchetes)
  - Declaração: **`tipo array[tamanho];`**
  - Protótipo da função: **`tipo funcao(tipo[])`**
  - Chamada: **`funcao(array);`**
  - Definição da função: **`tipo funcao(tipo array[]);`**
- Tamanho do array é passado para a função opcionalmente
- Arrays são passados por referência
- Nome do array é uma variável (***apontador***) com o endereço do primeiro elemento
- A função deve conhecer o espaço de armazenamento do array – número de elementos
- Modificação do conteúdo dos elementos do array original pela função

Passagem de Arrays para Funções

- Passagem de elementos do array
  - Passagem por valor
  - Passagem do nome do array com subscrito para a função (e.g. **`array[3]`**)
- Protótipo da Função
  - **`void modificaArray(int b[], int tamanho);`**
  - Nomes de parâmetros – opcionais no protótipo
  - **`int b[]`** – pode ser escrito como **`int[ ]`**
  - **`int tamanho`** – pode ser escrito como **`int`**
  - **`void modificaArray(int[], int);`**

Passagem de Arrays para Funções

```
/* Passagem de arrays e elementos isolados de arrays para funções */
#include <stdio.h>
#define TAM 5
void modificaArray(int[], int);
void modificaElemento(int);
int main() {
    int a[TAM] = {0, 1, 2, 3, 4}, i;

    printf("Efeitos da passagem de arrays inteiro em chamadas por "
           "referência:\n\nOs valores do array original são:\n");
    for (i = 0; i <= TAM - 1; i++)
        printf("%3d", a[i]);
    printf("\n");
    modificaArray(a, TAM);/* passagem por referência */
    printf("Os valores do array modificado são:\n");
    for (i = 0; i <= TAM - 1; i++)
        printf("%3d", a[i]);
```

Passagem de Arrays para Funções

```
printf("\n\nEfeitos da passagem de elementos em chamadas por
valor:"
                "\n\nO valor de a[3] é %d\n", a[3]);
    modificaElemento(a[3]);
    printf("O valor de a[3] é %d\n", a[3]);
    return 0;
}
void modificaArray(int b[], int tam) {
    int j;
    for (j = 0; j <= tam - 1; j++)
        b[j] *= 2;
    return;
}
void modificaElemento(int e) {
    printf("Valor em modificaElemento é %d\n", e *= 2);
    return;
}
```

Exercícios

1)Escreva um programa em C para ler e mostrar os elementos de um array de N inteiros, com N menor ou igual a 100, utilizando para isto uma função diferente para cada ação

2)Escreva um programa em C para gerar aleatoriamente os elementos de um array de 50 inteiros, com valores entre 1 e 100, sem permitir repetições, e, em seguida, mostrar o array gerado, usando uma função diferente para cada ação

3)Estenda o programa anterior para mostrar se determinado valor está ou não presente no array, usando uma função diferente para cada ação

Exercícios

4) Escreva um programa em C para ler dois conjuntos de inteiros de 10 elementos e mostrar o conjunto união destes conjuntos, utilizando para isto uma função para cada ação

5) Estenda o programa anterior para funcionar com conjuntos de tamanho diferentes de até 100 elementos, usando uma função diferente para cada ação

6) Estenda o programa anterior para mostrar, além do conjunto união, o conjunto interseção, usando uma função diferente para cada ação

Exercícios

7) Escreva um programa em C para simular a coleta de temperaturas médias inteiras diárias, durante um determinado mês em uma localidade, com temperaturas variando entre 17 e 28^{o}C, e mostrar os dias em que a temperatura média ficou acima da média do mês, utilizando para isto uma função para cada ação

8) Estenda o programa anterior para calcular o desvio padrão e a moda das temperaturas do mês, usando uma função diferente para cada ação

Arrays de Caracteres

- Possibilidade de inicialização de arrays de caracteres a partir do uso de literais do tipo cadeia de caracteres (***string***)

  `char string[] = "cadeia";`

- Cadeia de caracteres **`primeiro`** – array estático
  - Terminação de cadeias de caracteres – ***NULL*** (***`'\0'`***)
  - **`string`** é realmente composta por 7 elementos

    ```
    char string[] =
                {'c','a','d','e','i','a','\0'};
    ```

- Possibilidade de acesso a caracteres individuais

  `string[3]` – caractere **`'e'`**

Arrays de Caracteres

- Nome do array – endereço do array
- Leitura de cadeia de caracteres
- Uso de ***"&"*** desnecessário
  - String tem que ser criado antes da leitura, com o tamanho suficiente para a cadeia e o ***NULL*** (***'\0'***)

    ```
    scanf("%s", string);
    ```

  - Leitura de caracteres até a identificação de um espaço em branco
- Outra forma de ler cadeia – pode incluir espaços em branco

  ```
  gets(string);
  ```

- Escrita de cadeia de caracteres

  ```
  printf("%s", string);
  ```

  - Possibilidade de escrita além do array, com o formato ***"%c"*** – atenção e cuidado

Arrays de Caracteres

```
/* Tratamento de arrays de caracteres como cadeias de caracteres */
#include <stdio.h>

int main() {
    char string1[20], string2[] = "string literal";
    int i;

    printf("Entre com um string: ");
    scanf("%s", string1);
    printf("string1 é: %s\nstring2: é %s\n"
           "string1 com espaços entre caracteres é:\n", string1,
            string2);
    for (i = 0; string1[i] != '\0'; i++)
        printf("%c ", string1[i]);
    printf("\n");
    return 0;
}
```

Ordenação de Arrays

- Bubble Sort – ordenação borbulhante – (https://youtu.be/lyZQPjUT5B4)
- Várias etapas ao longo do array – os elementos de menor valor, os mais “leves”, são direcionados para o início do array
- Comparação de pares sucessivos de elementos (ordenação crescente)
  - Valores do par crescente (ou valores idênticos) – posição dos elementos no array inalterada
  - Valores do par decrescente – troca de posição dos elementos no array
- Repetição do processo de comparação

Bubble Sort

- Função para executar o Bubble Sort

```
void ordena_bolha(int array[], int tamanho) {
    int aux;

    for (int repete = 1; repete <= tamanho - 1; repete++)
        for (int indice = 0; indice <= tamanho - 2;
             indice++)
            if (array[indice] > array[indice + 1]) {
                aux = array[indice];
                array[indice] = array[indice + 1];
                array[indice + 1] = aux;
            }
    return;
}
```

Exemplo do Uso do Bubble Sort

```
// Programa para ler, ordenar e mostrar os N elementos de um
// array de inteiros
#include <stdio.h>

#define NUM_ELTOS 100

void le_elementos(int[], int);
void mostra_elementos(int[], int);
void ordena_bolha(int[], int);

int main() {
    int inteiros[NUM_ELTOS], n;

    printf("Lê, ordena e mostra os elementos de um array de tamanho"
           " N (máximo 100):\n\n");
    printf("Informe o valor número de elementos (N): ");
    scanf("%d", &n);
```

Exemplo do Uso do Bubble Sort

```
  printf("\nInforme os %d elementos do array:\n", n);
    le_elementos(inteiros, n);
    printf("\nElementos do array na ordem de leitura:\n");
    mostra_elementos(inteiros, n);
    ordena_bolha(inteiros, n);
    printf("\n\nElementos do array em ordem crescente:\n");
    mostra_elementos(inteiros, n);
    printf("\n\n");
    return 0;
}
// Lê os N elementos do array de inteiros
void le_elementos(int meu_array[], int quant_eltos) {

    for (int indice = 0; indice <= quant_eltos - 1; ++indice)
        scanf("%d", &meu_array[indice]);
    return;
}
 // Mostra os N elementos do array de inteiros
void mostra_elementos(int array_int[], int num_eltos) {
```

Exemplo do Uso do Bubble Sort

```
 // Mostra os N elementos do array de inteiros
void mostra_elementos(int array_int[], int num_eltos) {
    for (int indice = 0; indice <= num_eltos - 1; ++indice)
        printf("%d  ", array_int[indice]);
    return;
}
// Ordena um array de inteiros na ordem crescente
void ordena_bolha(int array[], int tamanho) {
    int aux;
    for (int repete = 1; repete <= tamanho - 1; repete++)
        for (int indice = 0; indice <= tamanho - 2;
             indice++)
            if (array[indice] > array[indice + 1]) {
                aux = array[indice];
                array[indice] = array[indice + 1];
                array[indice + 1] = aux;
            }
    return;
}
```

Pesquisa de Valores em Arrays

- Pesquisa sequencial ou linear (https://youtu.be/-PuqKbu9K3U)
  - Simplicidade
  - Comparação de cada elemento do array com o valor de procura (valor chave)
  - Utilidade em arrays pequenos ou desordenados
- Pesquisa binária (https://youtu.be/iP897Z5Nerk)
  - Só para arrays ordenados
  - Comparação do elemento do meio com a chave (o procurado)
  - Se ***iguais*** – elemento encontrado
  - Se ***chave < elemento do meio*** – busca na primeira metade do array
  - Se ***chave > elemento meio*** – busca na segunda metade do array
  - Repetição do processo

Pesquisa Linear ou Sequencial

- Função para executar a pesquisa linear ou sequencial – retorna a posição do elemento no array, se encontrado, ou -1, caso não encontrado

```
int pesquisa_linear(int procurado, int array[], int tamanho) {
    int indice;

    for (indice = 0; indice <= tamanho - 1; ++indice)
        if (procurado == array[indice])
            return indice;
    return -1;
}
```

Pesquisa Binária

- Função para executar a pesquisa binária – retorna a posição do elemento no array, se encontrado, ou -1, caso não encontrado

```
int pesquisa_binaria(int procurado, int array[], int tamanho) {
    int inicio, meio, fim;

    inicio = 0;
    fim = tamanho - 1;
    while (inicio <= fim) {
        meio = (inicio + fim) / 2;
        if (procurado == array[meio])
            return meio;
        if (procurado < array[meio])
            fim = meio - 1;
        else
            inicio = meio + 1;
    }
    return -1;
}
```

Exercícios

9) A média de um grupo de valores é soma destes valores dividida pelo números de valores – faça um programa para gerar N inteiros entre 1 e 99, e determinar sua média (N no máximo será 1000)

10) A mediana de um grupo de valores é o elemento do meio quando estes estão ordenados, porém, quando se tem um número de valores par, a mediana é a média dos dois elementos do meio – acrescente uma nova funcionalidade ao programa anterior para que seja encontrada também a mediana dos N valores

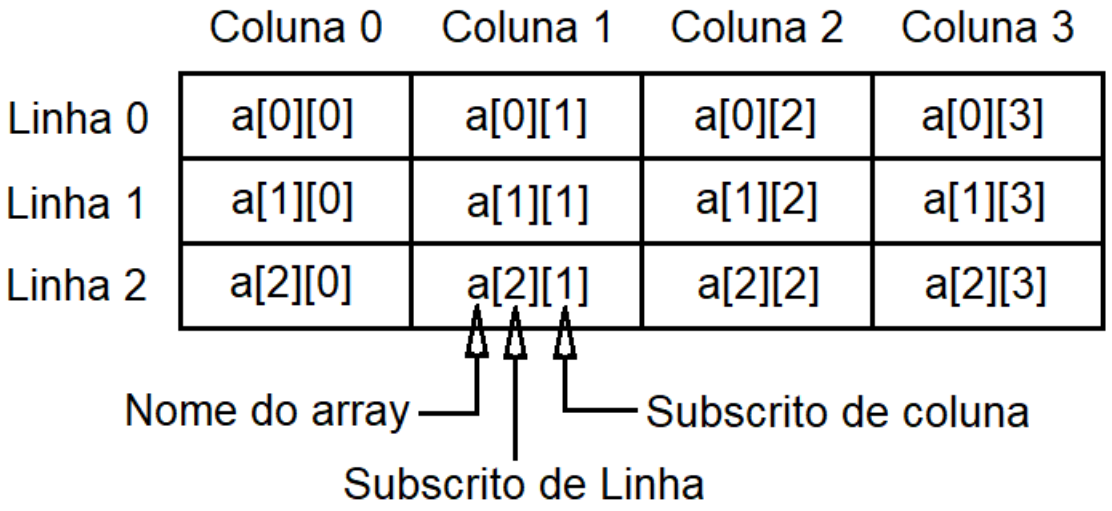
Exercícios

11) A moda de um grupo de valores é o valor que ocorre mais vezes no grupo – acrescente uma nova funcionalidade ao programa anterior para que seja encontrada também a moda dos N valores

12) Faça um programa para gerar um conjunto de N inteiros entre 1 e 999 e verificar se M está ou não presente no conjunto (N no máximo será 100). Se M estiver presente, mostre em que posição. Use pesquisa linear e pesquisa binária para a verificação.

Arrays com Múltiplos Subscritos

- Um array na linguagem C pode ter várias dimensões (vários subscritos) – até três possibilita uma visão espacial
- Matrizes são tabelas com duas dimensões – organizadas em linhas e colunas (***array m x n***)
- Matrizes – especificação da linha e, em seguida, da coluna (sempre nessa ordem)



Arrays com Múltiplos Subscritos

- Declaração e inicialização de matrizes

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

1	2
3	4

  - Agrupamento de inicializadores – linhas entre chaves
  - Se os inicializadores forem em menor número – elementos não especificados ajustados para zero

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

1	0
3	4

- Referência de elementos
  - Especificação da linha e, em seguida, da coluna

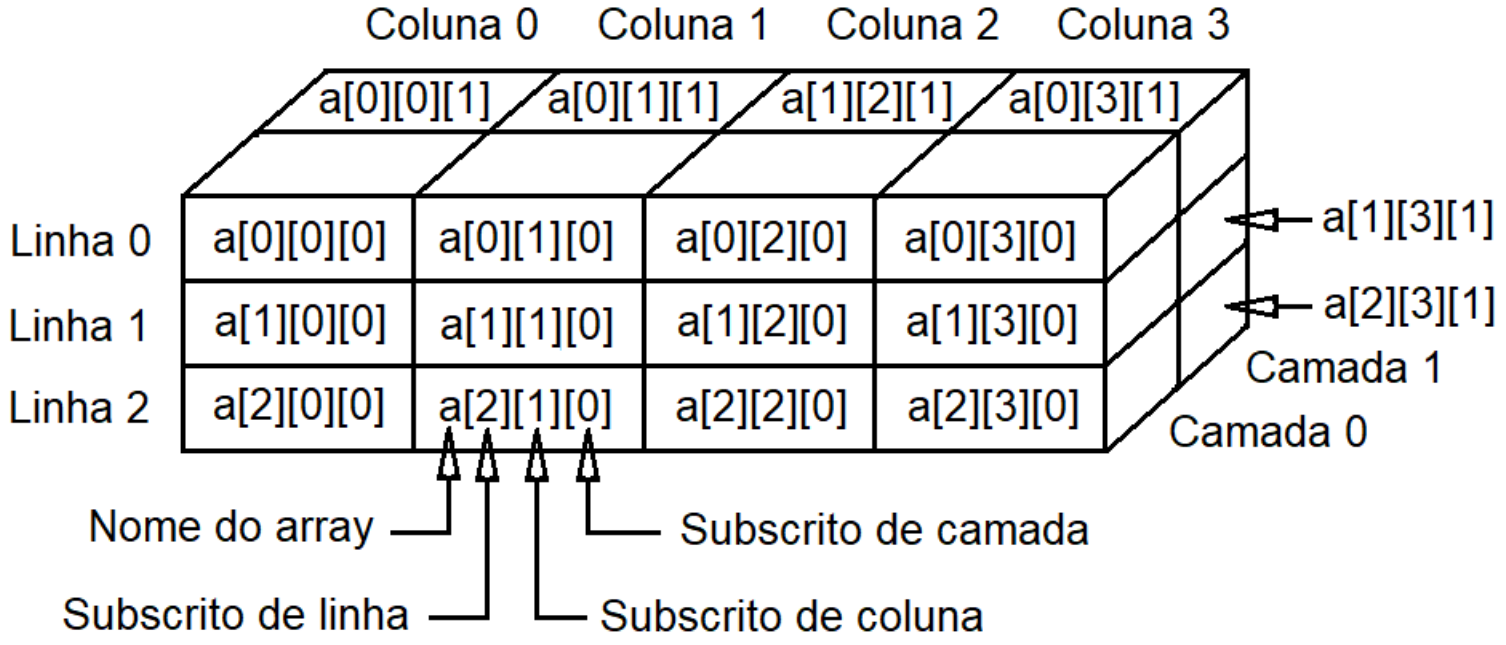
```
printf("%d", m[0][1]);
```

Arrays com Múltiplos Subscritos

- Declaração de arrays com três dimensões – três subscritos

```
int a[3][4][2];
```

- Referência de elementos de array com três subscritos
  - **`a[linha][coluna][camada]`**
  - Especificação da linha, da coluna e da camada (sempre nessa ordem)



Passagem de Matriz para Função

- Uma matriz é passada por referência (endereço) para uma função – o nome da matriz contém seu endereço
- Conhecendo o endereço da matriz, a função pode modificá-la independentemente de onde tenha sido criada
- Protótipo

```
tipo nome_função(tipo[][num_cols]);
```

- Chamada

```
nome_função(nome_matriz);
```

- Definição da matriz do como parâmetro da função

```
tipo nome_função(tipo nome_matriz[][num_cols]) {
    . . .
}
```

Exemplo de Uso de Matriz

```
/* Controle de notas de alunos, usando uma matriz */
#include <stdio.h>
#include <locale.h>
#define ALUNOS 3
#define EXAMES 4

int minimo(const int[][EXAMES], int, int);
int maximo(const int[][EXAMES], int, int);
double media(const int[], int);
void imprimeArray(const int[][EXAMES], int, int);

int main() {
   int aluno;
   const int notas[ALUNOS][EXAMES] = {{77, 68, 86, 73}, {96, 87, 89, 78},
                                      {70, 90, 86, 81}};
   setlocale(LC_ALL, "");
   printf("O array é:\n");
   imprimeArray(notas, ALUNOS, EXAMES);
```

Exemplo de Uso de Matriz

```
   printf("\n\nNota menor: %d\nNota maior: %d\n",
          minimo(notas, ALUNOS, EXAMES), maximo(notas, ALUNOS, EXAMES));
   for (aluno = 0; aluno <= ALUNOS - 1; aluno++)
      printf("\Nota média do aluno %d %.2f\n", aluno,
             media(notas[aluno], EXAMES));
   return 0;
}
/* Determinação da menor nota */
int minimo(const int notas[][EXAMES],int pupilos, int testes) {
   int i, j, menorNota = 100;

   for (i = 0; i <= pupilos - 1; i++)
      for (j = 0; j <= testes - 1; j++)
         if (notas[i][j] < menorNota)
            menorNota = notas[i][j];
   return menorNota;
}
```

Exemplo de Uso de Matriz

```
/* Determinação da nota máxima */
int maximo(const int notas[][EXAMES], int pupilos, int testes) {
   int i, j, maiorNota = 0;

   for (i = 0; i <= pupilos - 1; i++)
      for (j = 0; j <= testes - 1; j++)
         if (notas[i][j] > maiorNota)
            maiorNota = notas[i][j];
   return maiorNota;
}
/* Determinação da nota média para um aluno */
double media(const int notas[], int testes) {
   int i, total = 0;

   for (i = 0; i <= testes - 1; i++)
      total += notas[i];
   return (double)total / testes;
}
```

Exemplo de Uso de Matriz

```
/* Impressão do array */
void imprimeArray(const int notas[][EXAMES], int pupilos, int testes) {
   int i, j;

   printf("                Notas\n");
   printf("          [0]  [1]  [2]  [3]");
   for (i = 0; i <= pupilos - 1; i++) {
      printf("\naluno[%d] ", i);
      for (j = 0; j <= testes - 1; j++)
         printf("%-5d", notas[i][j]);
   }
}
```

Exercícios

1) Faça um programa em C que leia e mostre os elementos de uma matriz de inteiros MxN, com M e N menores ou iguais a 100

2) Faça um programa em C preencha e mostre uma matriz identidade de ordem 10

3) Faça um programa em C que leia os elementos de uma matriz de números reais 5x7 e mostre a matriz lida e sua transposta

4) Faça um programa em C que leia os elementos de duas matrizes de números reais MxN e mostre a soma destas matrizes, com M e N menores ou iguais a 10

5) Faça um programa em C que leia os elementos de duas matrizes de números reais MxN e mostre o produto destas matrizes, com M e N menores ou iguais a 10

Exercícios

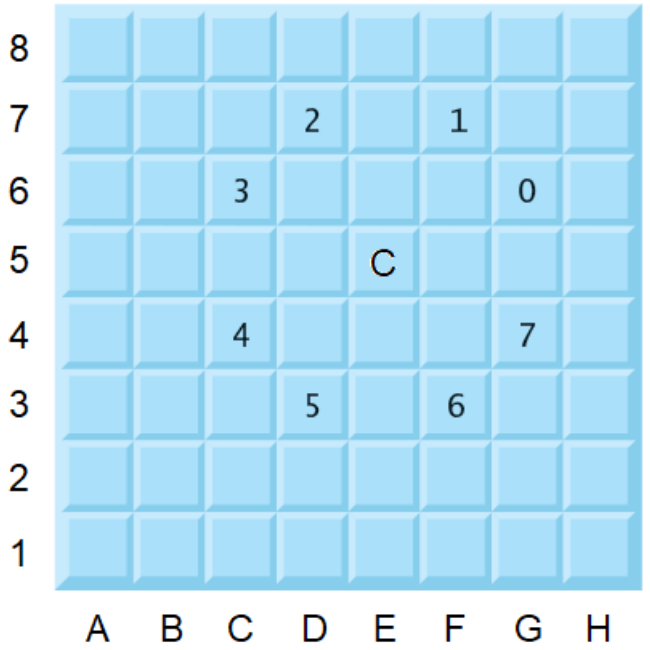
6) Faça um programa em C para receber as jogadas de um Jogo da Velha e mostrar o tabuleiro do jogo atualizado a cada jogada, posicionando os **O**'s e os **X**'s

Sugestão para o tabuleiro:

```
     |     |
  1  |  2  |  3
-----+-----+-----
  4  |  5  |  6
-----+-----+-----
  7  |  8  |  9
     |     |
```

- A numeração corresponde às posições a serem referenciadas nas jogadas
- O programa deve informar automaticamente quando houver vitória de um jogador ou houver empate.
- Use uma matriz para representar o tabuleiro na memória

Exercícios



7) ***Movimento do cavalo.*** A peça do jogo de xadrez chamada cavalo se movimenta em forma de L (por duas casas em uma direção e depois por uma casa em uma direção perpendicular). Assim, por um quadrado no meio de um tabuleiro vazio, o cavalo pode fazer oito movimentos diferentes (numerados de 0 a 7), como mostra a figura acima.

Exercícios

Faça um programa que receba, utilizando a notação algébrica do xadrez (letra da coluna seguida do número da linha), a posição de um cavalo no tabuleiro e mostre na tela o tabuleiro e as posições para as quais este cavalo pode se movimentar, utilizando asteriscos para assinalar estas posições. Use uma matriz para representar o tabuleiro e os possíveis movimentos válidos.