Laboratório de programação

Matrizes

Universidade Estadual Vale do Acaraú – UVA

Paulo Regis Menezes Sousa

paulo_regis@uvanet.br

- Há casos em que organizar os dados em uma estrutura de **linhas** e **colunas**, como uma tabela, é mais útil.

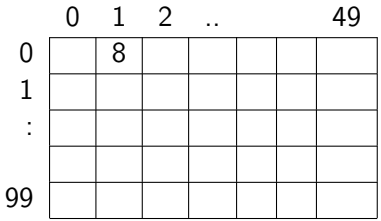
2	7	3	1
4	9	0	7
1	6	5	6

- Para isso, usamos um array com **duas dimensões**, ou seja, uma **Matrizes**.
- Em linguagem C, a declaração de uma matriz segue esta forma geral:

```
tipo nome_array[nro_linhas][nro_colunas];
```

- Para acessar determinada posição da matriz, precisamos usar dois índices: o primeiro índice especifica a linha, e o segundo, a coluna da matriz:

```
1 int mat[100][50];
2 mat[0][1] = 8;
```



- Como uma matriz possui dois índices, precisamos de dois comandos de repetição para percorrer todos os seus elementos.

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int mat[100][50];
    int i,j;
    for (i = 0; i < 100; i++) {
        for (j = 0; j < 50; j++) {
            printf("Digite o valor de mat[%d][%d]: ",i,j);
            scanf("%d",&mat[i][j]);
        }
    }
    return 0;
}
```

- A carga inicial é realizada de forma similar à de vetores com uma dimensão.

```
1  char mat[3][3]= {{'a','b','c'},{'d','e','f'},{'g','h','i'}};
```

- Na inicialização de um vetor pode-se omitir apenas um valor numérico para a dimensão mais à esquerda, sendo o respectivo valor calculado pelo compilador.

Incorreto

```
1 char mat[][]= {{'a','b','c'},{'d','e','f'},{'g','h','i'}};
```

Correto

```
1 char mat[][3]= {{'a','b','c'},{'d','e','f'},{'g','h','i'}};
```

- A passagem de matrizes para uma função é realizada indicando no cabeçalho desta pelo menos o número de colunas.
- Apenas a dimensão mais à esquerda pode ser omitida, colocando-se apenas `[ ]` ou um asterisco `*`.
- **Exemplo:**

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

Possíveis cabeçalhos de função que receberiam como parâmetro a matriz `mat`

```
1 void funcaoA(int mat[3][2]);
2 void funcaoB(int mat[][2]);
```

- A linguagem C permite que se crie um array com mais de duas dimensões de maneira fácil.

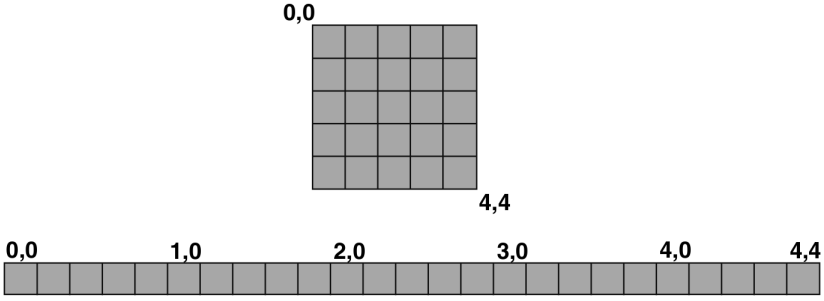
```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    //declara array de int com 1 dimensão
    int vet[5];
    //declara array de float com 2 dimensões
    float mat[5][5];
    //declara array de double com 3 dimensões
    double cub[5][5][5];
    //declara array de int com 4 dimensões
    int X[5][5][5][5];
    return 0;
}
```

- O acesso ao valor de uma posição de um array é feito utilizando um índice para cada dimensão do array.

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int cub[5][5][5];
    int i,j,k;
    //preenche o array de 3 dimensões com zeros
    for (i=0; i < 5; i++) {
        for (j=0; j < 5; j++){
            for (k=0; k < 5; k++) {
                cub[i][j][k] = 0;
            }
        }
    }
    return 0;
}
```

Estrutura interna dos arrays

Apesar de terem o comportamento de estruturas com mais de uma dimensão, os dados dos arrays, independentemente do número de dimensões que possuam, são sempre armazenados linearmente na memória. É o uso dos colchetes que cria a impressão de estarmos trabalhando com mais de uma dimensão.



Exercício 22

Faça um programa que leia uma matriz de tamanho 4×4. Imprima na tela o maior valor contido nessa matriz e a sua localização (linha e coluna).

Exercício 23

Leia uma matriz de tamanho 3×3. Em seguida, imprima a soma dos valores contidos em sua diagonal principal.

Exercício 24

Crie uma função que receba por parâmetro uma matriz de inteiros e a inicialize da seguinte forma:

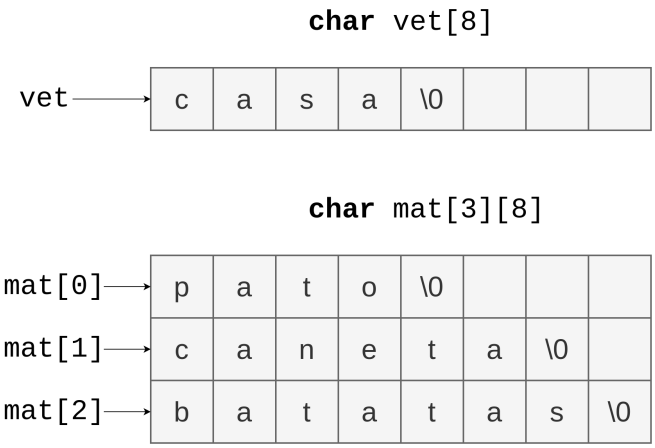
$$\text{mat}[i][j] = 2i + 7j - 2 \text{ se } i < j;$$
$$\text{mat}[i][j] = 3i^2 - 1 \text{ se } i = j;$$
$$\text{mat}[i][j] = 4i^3 - 5j2 + 1 \text{ se } i > j.$$

- Observe os programas abaixo

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

int main() {
    char vet[8];

    scanf("%s", vet);

    printf("%s\n", vet);

    return 0;
}
```

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

int main() {
    int i;
    char mat[3][8];

    for (i=0; i<3; i++)
        scanf("%s", mat[i]);

    for (i=0; i<3; i++)
        printf("%s\n", mat[i
            ]);

    return 0;
}
```

- A imagem abaixo fornece uma visualização de como as strings são armazenadas.



- A título de exemplo, vamos observar como poderíamos implementar o famoso Jogo da Velha representando tabuleiro do jogo com uma matriz de caracteres.

x		o
	x	
		o

- A declaração deverá ser realizada do seguinte modo:

```
1       char velha[3][3];
```

- A carga inicial é realizada de forma similar à de vetores com uma dimensão.

```
1 char velha[3][3]= {{' ',' ',' '},{' ',' ',' '},{' ',' ',' '}};
```

- Na inicialização de um vetor pode-se omitir apenas um valor numérico para a dimensão mais à esquerda, sendo o respectivo valor calculado pelo compilador.

Incorreto

```
1 char velha [][]= {{' ',' ',' '},{' ', ' ', ' '},{' ',' ',' '}};
```

Correto

```
1 char velha [][3]= {{' ',' ',' '},{' ',' ',' '},{' ',' ',' '}};
```

Exercício 25

Escreva um programa que coloque o tabuleiro do jogo da velha nesse estado, depois de ter sido iniciado com espaços durante a declaração do mesmo.

x		o
	x	
		o

Em seguida mostre o tabuleiro na tela.

```c
#include <stdio.h>
#define DIM 3
#define VAZIO ' '

void mostrar(char s[DIM][DIM]) {
    int i,j;
    printf("   0   1   2\n");
    for (i=0; i < DIM; i++) {
        printf("%d ",i);
        for (j=0; j < DIM; j++)
            printf(" %c %c",s[i][j], j == DIM-1 ? ' ' : '|');

        if (i != DIM-1)
            printf("\n  ---+---+---") ;
        printf("\n");
    }
}
```

```c
int main() {
    char velha[DIM][DIM] = {' ',' ',' ',' ', ' ', ' ',' ',' ',' '};
    int x, y, numeroDeJogadas = 0;
    char simbolo = 'O'; // Simbolo inicial
    while (1) {
        system("clear"); // apaga o console
        mostrar(velha);
        printf("\nJogador: %c\nPosição (x y): ", simbolo);
        scanf("%d %d",&x,&y);
        if (velha[x][y] == VAZIO) {
            velha[x][y] = simbolo;
            simbolo = (simbolo == 'O') ? 'X' : 'O';
            numeroDeJogadas++;
        }
        else
            printf("Posição já ocupada\nJogue Novamente!!!\n");
        if (numeroDeJogadas == DIM*DIM) break; /* Finalizar o jogo */
    }
    mostrar(velha);
    return 0;
}
```

- Em linguagem C podemos passar argumentos através da linha de comando para um programa quando ele inicia.
- A função `main` recebe parâmetros passados via linha de comando como vemos a seguir:

```
1                int main(int argc, char *argv[])
```

- Onde:
  - `argc` é um valor inteiro que indica a quantidade de argumentos que foram passados ao chamar o programa.
  - `argv` é um vetor de **char** que contém os argumentos, um para cada *string* passada na linha de comando.
- O primeiro elemento `argv[0]` armazena o nome do programa que foi chamado no *prompt*, sendo assim, `argc` é pelo menos igual a 1, pois no mínimo existirá um argumento.

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

int main(int argc, char* argv[]){
    int i;

    printf("programa: %s\n",argv[0]);

    if(argc == 1)
        printf(" info: nenhum parâmetro.\n");

    for(i=1; i<argc; i++)
        printf(" parametro [%d]: %s\n", i, argv[i]);

    return 0;
}
```

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

int main(int argc, char* argv[]){
    int i, soma;

    printf("programa: %s\n",argv[0]);

    if(argc == 1)
        printf(" info: nenhum parâmetro.\n");

    for(i=1, soma=0; i<argc; i++)
        soma = soma + atoi(argv[i]);

    printf(" soma = %d\n",soma);

    return 0;
}
```

Continue a implementação do jogo da velha...

Exercício 26

Altere a função `main` para que você possa passar como parâmetro para o programa pelo console o caractere que deseja usar no jogo (O ou X). Caso o usuário não passe parâmetro mantenha o caractere inicial como O. No caso de um caractere inválido ser passado envie uma mensagem informando o erro e encerre o programa.

Exercício 27

Crie uma função que receba como parâmetro a matriz do jogo e retorne 0 caso não haja um vencedor, 1 caso o primeiro jogador seja o vencedor e 2 caso seja o segundo. Acrescente esta função ao jogo da velha de forma que ele termine quando um jogador completar uma linha, coluna ou diagonal e diga quem venceu.