Guião da Ficha de Trabalho 4

Arrays Multidimensionais em C

A forma mais simples de um array multidimensional é o **array bidimensional**. Um array bidimensional é basicamente uma lista de *arrays* unidimensionais. Um *array* bidimensional é declarado da seguinte forma:

```
tipo nomeArray[i][j];
```

Exemplo deste tipo de *array* são as matrizes. Por exemplo, uma matriz **m** de 3 linhas e 4 colunas com elementos do tipo *float* pode ser declarado da seguinte forma:

```
float m[3][4];
```

m é um array de duas dimensões que pode albergar 12 elementos:

	Coluna 0	Coluna 1	Coluna 2	Coluna 3
Linha 0	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Linha 1	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Linha 2	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Formas equivalentes de inicializar um array bidimensional **c** com 2 linhas e 3 colunas:

```
int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[2][3] = {1, 3, 0, -1, 5, 9};
```

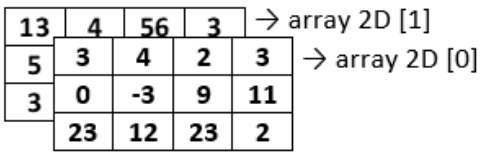
Analogamente, um ***array* tridimensional** é inicializado da mesma forma. Por exemplo, **y** pode ter 24 elementos (2 x 4 x 3):

```
float y[2][4][3];
```

Exemplo de inicialização de um array tridimensional:

```
int test[2][3][4] = { { {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },
                      { {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} } };
```

Representação conceptual:



Exercícios em sala de aula

1) Criar um *array* que armazene valores de temperatura de duas cidades durante uma semana fornecidos pelo utilizador

```
Cidade 1, Dia 1: 10
Cidade 1, Dia 2: 11
Cidade 1, Dia 3: 13
Cidade 1, Dia 4: 14
Cidade 1, Dia 5: 15
Cidade 1, Dia 6: 14
Cidade 1, Dia 7: 12
Cidade 2, Dia 1: 21
Cidade 2, Dia 2: 22
Cidade 2, Dia 3: 25
Cidade 2, Dia 4: 26
Cidade 2, Dia 5: 24
Cidade 2, Dia 6: 23
Cidade 2, Dia 7: 23

Valores de temperatura registados:

Cidade 1, Dia 1 = 10
Cidade 1, Dia 2 = 11
Cidade 1, Dia 3 = 13
Cidade 1, Dia 4 = 14
Cidade 1, Dia 5 = 15
Cidade 1, Dia 6 = 14
Cidade 1, Dia 7 = 12
Cidade 2, Dia 1 = 21
Cidade 2, Dia 2 = 22
Cidade 2, Dia 3 = 25
Cidade 2, Dia 4 = 26
Cidade 2, Dia 5 = 24
Cidade 2, Dia 6 = 23
Cidade 2, Dia 7 = 23
```

2) Dadas duas matrizes de dimensões 2 x 2, pelo utilizador, apresentar a matriz soma

```
1a matriz
Introduza elemento da posicao 1 1: 1
Introduza elemento da posicao 1 2: 1
Introduza elemento da posicao 2 1: 2
Introduza elemento da posicao 2 2: 2
2a matriz
Introduza elemento da posicao 1 1: 2
Introduza elemento da posicao 1 2: 2
Introduza elemento da posicao 2 1: 1
Introduza elemento da posicao 2 2: 1

Matriz Soma:
3.0     3.0
3.0     3.0
```

3) Escrever um algoritmo usando as funções *inserir()*, *multiplicar()* e *mostrar()* que permita inserir duas matrizes e obter a sua multiplicação

```
Introduza numero de linhas e de colunas da primeira matriz: 2 2
Introduza numero de linhas e de colunas da segunda matriz: 2 2

Introduza os elementos da matriz 1:
Posicao 1 1: 1
Posicao 1 2: 2
Posicao 2 1: 3
Posicao 2 2: 4

Introduza os elementos da matriz 2:
Posicao 1 1: 4
Posicao 1 2: 3
Posicao 2 1: 2
Posicao 2 2: 1

Matriz final:
8  5

20  13
```

Ficheiros em C

Um ficheiro representa uma **sequência de bytes**, independentemente do tipo de ficheiro. A linguagem C contempla um conjunto de funções inclusas na biblioteca "`stdio.h`" que permitem manipular ficheiros.

Quando se trabalha com ficheiros, é necessário declarar **um apontador para o ficheiro**, pois é o que permite a comunicação entre o ficheiro e o programa:

```
FILE *f;
```

Para se poder trabalhar com o ficheiro, é necessário abri-lo usando a função `fopen()`:

```
f = fopen("pathEnomeFicheiro", "modo");
```

Por exemplo, para abrir o ficheiro `fic.txt` para escrita usa-se a sintaxe:

```
fopen("C:\Users\fic.txt", "w");
```

Na tabela a seguir apresenta-se os diversos **modos** existentes para **manipular ficheiros**:

Modo	Significado	O que acontece ao ficheiro
r	Abre o ficheiro para leitura.	Se o ficheiro não existe, fopen() retorna NULL.
rb	Abre o ficheiro para leitura em modo binário.	Se o ficheiro não existe, fopen() retorna NULL.
w	Abre o ficheiro para escrita.	Se o ficheiro não existe, é criado um. Se já existe, é reescrito.
wb	Abre o ficheiro para escrita em modo binário.	Se o ficheiro não existe, é criado um. Se já existe, é reescrito.
a	Abre o ficheiro para adicionar conteúdo no fim.	Se o ficheiro não existe, é criado um.

ab	Abre o ficheiro para adicionar conteúdo no fim, em modo binário.	Se o ficheiro não existe, é criado um.
r+	Abre o ficheiro para leitura e escrita.	Se o ficheiro não existe, fopen() retorna NULL.
rb+	Abre o ficheiro para leitura e escrita em modo binário.	Se o ficheiro não existe, fopen() retorna NULL.
w+	Abre o ficheiro para leitura e escrita.	Se o ficheiro não existe, é criado um. Se já existe, é reescrito.
wb+	Abre o ficheiro para leitura e escrita em modo binário.	Se o ficheiro não existe, é criado um. Se já existe, é reescrito.
a+	Abre o ficheiro para leitura e para adicionar conteúdo no fim.	Se o ficheiro não existe, é criado um.
ab+	Abre o ficheiro para leitura e para adicionar conteúdo no fim, modo binário.	Se o ficheiro não existe, é criado um.

Após feitas todas as manipulações ao ficheiro, deve-se fechar o mesmo usando a função `fclose()` como se apresenta a seguir:

```
fclose(f);
```

Para ler a partir de um ficheiro de texto e para escrever no mesmo, são usadas as funções `fscanf()` e `fprinf()`, respetivamente.

Exercícios em sala de aula

4) Corrigir o programa para que este permita guardar o valor que o utilizador escolher num ficheiro de texto

```
#include <stdio.h>

int main() {
            int num;
            fptr = fopen("programa.txt","w");


            if(fptr == NULL) {
                printf("Erro!");
                exit(1);
            }
            printf("Introduza um numero: ");
            scanf("%d",&numero);
            fprintf(f,"%f",num);
            fclose(fptr)
            return 0;
}
```

5) Escrever um algoritmo para ler o ficheiro e apresentar o valor escolhido pelo utilizador no exercício anterior

Outras funções relevantes para a manipulação de ficheiros são `fwrite()` e `fread()`, para ler e escrever num ficheiro no caso de ficheiros binários. Sintaxe de aplicação:

```
fwrite(endereçoVariável, tamanhoVariável, númeroVariável, apontadorFicheiro);
fread(endereçoVariável, tamanhoVariável, númeroVariável, apontadorFicheiro);
```

Tanto a função `fwrite()` como a função `fread()` precisam de quatro argumentos: o endereço dos dados a serem escritos em disco, o tamanho desses dados, o número desse tipo de dados e o apontador para o ficheiro onde se quer escrever.

Outra função auxiliar é a função `fseek()`, que procura o cursor para um dado registo no ficheiro

```
fseek(FILE * apontador, long int deslocamento, int pontoDEpartida)
```

O primeiro parâmetro é um apontador para o ficheiro. O segundo parâmetro é a posição do registo a procurar e o terceiro parâmetro especifica a localização onde começa a procurar (`SEEK_SET` se começar do início do ficheiro, `SEEK_END` se começar do fim do ficheiro e `SEEK_CUR` se começar a posição atual do cursor no ficheiro).

Tarefa única para avaliação na aula seguinte (1.5 valores): ler o conteúdo de um ficheiro, alterá-lo e voltar a guardar

Copie o conteúdo seguinte para um ficheiro `tabuleiro.txt` e guarde.

```
3 4
X . . X
. O . X
. . O .
```

De seguida crie um algoritmo que

1. abra o ficheiro
2. apresente o seu conteúdo ao utilizador
3. permita que este altere o seu conteúdo (substituir **.** por **X** ou **O**)
4. grave as alterações no ficheiro inicial
5. apresente o conteúdo alterado ao utilizador.

Pode optar por usar um menu à semelhança do que fez na tarefa do NIB.