Introdução à Programação #14

Arrays

Até agora apenas falámos de **variáveis escalares**, ou seja, variáveis com valores individuais. Vamos então falar de uma nova **estrutura de dados**: as *arrays*.

Arrays, também conhecidas por "tabelas" ou "vetores", são estruturas de dados que, ao contrário de variáveis escalares, nos permitem armazenar mais do que um valor.

Estas são como matrizes (ou tabelas) de dados em que cada dado está localizado numa determinada posição que pode ser acedida através de "coordenadas". Existem dois tipos: as unidimensionais e as multidimensionais.

Arrays Unidimensionais

Arrays unidimensionais podem ser comparadas a tabelas com uma única coluna e várias linhas. São o tipo mais simples de *arrays*.

Sintaxe

```
1.  tipo nome[tamanho];
```

Onde:

- **tipo** → O tipo de dados que a *array* vai conter;
- **nome** → O nome da *array*, tal como se faz com uma variável;
- **tamanho** → O número máximo de elementos que a *array* irá conter.

Exemplo

```
1.  int idades[10]; //Array de 10 elementos
2.
3.  idades[0] = 14; //Atribuição Correta
4.  idades[4] = 12; //Atribuição Correta
5.  idades[7] = 15; //Atribuição Correta
6.  idades[10] = 20; //Atribuição Incorreta (tamanho máximo da array ultrapassado)
```

No exemplo anterior, a última declaração está errada porque o índice máximo da *array* foi ultrapassado, ou seja, a *array* apenas pode conter 10 elementos mas, como a contagem inicia sempre no zero, "idades[10]" refere-se ao décimo primeiro elemento.

Podemos também, à semelhança do que acontece com as variáveis, atribuir valores às *arrays* no momento em que as declaramos da seguinte forma:

```
1.  int idades[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Arrays Multidimensionais

Arrays multidimensionais são todas aquelas que contenham mais do que uma dimensão, ou seja, se comparamos a uma tabela, têm mais do que uma coluna.

Sintaxe

```
tipo nome[linhas][colunas];
```

Exemplo

```
/*
 * Declaração de uma array 6x5
 */
float notas[6][5];

notas[0][0] = 18.7; //1ª Linha, 1ª Coluna
notas[0][1] = 15.4; //1ª Linha, 2ª Coluna
notas[3][2] = 19.6; //4ª Linha, 3ª Coluna
notas[5][4] = 17.5; //6ª Linha, 5ª Coluna
notas[6][0] = 20.0; //Excedeu o máximo de linhas (6)
notas[5][5] = 17.4; //Excedeu o máximo de colunas (5)
```

No exemplo anterior é possível visualizar a criação de uma tabela *array* 6 linhas por 5 colunas. Traduzindo esta *array* para uma tabela, ficaríamos com o seguinte:

Que, inserindo os dados anteriormente utilizados no exemplo, ficaria preenchida da seguinte forma:

18.7	15.4			
		19.6		
				17.4

A sintaxe que aplicámos anteriormente para atribuir valores a *arrays* quando as inicializamos também pode ser utilizada neste contexto. Veja o seguinte exemplo:

```
int idades[2][4] = {
    {1, 2, 3, 4},
    {0, 1, 2, 3}};
```

http://pplware.sapo.pt/tutoriais/vamos-programar-introducao-a-programacao-14