# 8ª Aula - Variáveis com Dimensionadas (Vectores e Matrizes). Variáveis Aleatórias.

## Programação

## Mestrado em Engenharia Física Tecnológica

Samuel M. Eleutério
sme@tecnico.ulisboa.pt

Departamento de Física
Instituto Superior Técnico
Universidade de Lisboa

# Variáveis Dimensionadas - Introdução

- Até aqui limitámo-nos a considerar variáveis às quais associávamos **um valor numérico**.

  Por exemplo: **i = 1;**, **x = 2.3;**, etc..
- Se nos limitássemos a estas variáveis, seria muito pouco prático guardar e manipular os dados de uma experiência em que tinhamos, por exemplo, **50 valores**.
- E se em vez de **50** tivessemos **1 milhão**?
- Obviamente, essa não é a solução adequada.
- Felizmente, existem mecanismos relativamente simples que nos permitem chegar a uma solução adequada.

# Variáveis Dimensionadas - Introdução

Se quisermos **guardar** em casa, digamos, **20** números (representados por pedrinhas) em **caixas**, como fazemos?

1. Compramos caixas (por exemplo azuis), colocamos **20 delas encostadas umas às outras** e chamamos-lhes as '**caixas azuis**';
2. Depois, na **primeira** colocamos as **pedrinhas** correspondente ao **primeiro número** que queremos representar;
3. A seguir, passamos à **segunda** e fazemos o mesmo e assim sucessivamente até à última;
4. A partir daqui podemos referir-nos a esses valores como "**o primeiro**", "**o sétimo**", "**o décimo segundo**" das '**caixas azuis**'...

Em resumo, arranjámos uma **sequência de caixas**, as '**caixas azuis**', que colocámos em algum **sítio**, **numerámo-las** e passámos a **referirmo-nos** a elas pelos seus **números**.

# Variáveis Dimensionadas - Caracterização

A descrição feita no slide anterior não é mais do que a de uma **variável dimensionada** (**long int ca[20];**) em que:

- A variável **ca** ('**caixas azuis**') é um ponteiro para a posição inicial da memória que lhe foi atribuída;
- O valor entre rectos **[20]** diz-nos que temos de reservar espaço para 20 '**long int**'s (**20 caixas**), isto é, **80 bytes** ($20 \times 4$);
- Assim, o **primeiro valor** guardado estará na posição de memória '**ca**', o segundo em '**ca + 'o comprimento de 1 long int'**', o terceiro em '**ca + 'o comprimento de 2 long int'**' e assim sucessivamente;
- Isto é, o **primeiro** avança '**0**', o **segundo** '**1**', o **terceiro** '**2**', etc., então designaremos o **primeiro** valor por '**ca[0]**', o **segundo** por '**ca[1]**', o **terceiro** por '**ca[2]**', ... até ao último '**ca[19]**'.

**Nota Muito Importante**: A numeração das variáveis dimensionadas (de dimensão '**N**') começa sempre em '**0**' e vai até '**N - 1**'.

# Variáveis Dimensionadas - Exemplo

```
x = x0;
for (i1 = 0 ; i1 <= i0 ; ++i1)
  x = r * x * (1.0 - x);
x_ref = x;
for (i1 = 0 ; i1 < imax ; ++i1)
  {
    x = r * x * (1.0 - x);
    vx[i1] = x;
    if (fabs (x - x_ref) < delta)
      break;
  }
  ++i1;
printf ("r=%.2lf , Qt: %ld - ",r,i1);
for (i2 = 0 ; i2 < i1 ; ++i2)
  printf (" %lf", vx[i2]);
printf ("\n");
```

Vamos alterar '**Prog05_08.c**', da função logística, para guardar, numa **variável dimensionada**, os **valores** das órbitas periódicas:

## Prog05_10.c

- Deixamos a função estabilizar;
- Guardamos os valores da órbita no vector **vx[i1]**
- Incrementamos **i1** de uma unidade para ser igual ao periodo da órbita encontado;
- Escrevemos '**r**' e '**i1**';
- A seguir os valores da órbita;
- Passamos à linha seguinte.

# Variáveis Dimensionadas - Notas Finais

- O programa **Prog05_11.c** é uma variante de **Prog05_10.c** em que a **escrita** no ecran passou a ser feita num **ficheiro**.
- Até aqui vimos como se trabalha com uma **variável dimensionada** com uma **única dimensão**.
- No entanto, podemos criar **variáveis dimensionadas** com muito mais **dimensões** e dos mais variados **tipos**. Exemplos:

  **float bbb[4][23][2][3];** **int k[4][2];** **double x[4][234];**

  O modo como lidamos com estas variáveis é **análogo** ao que vimos para uma só dimensão.
- Para terminar, uma chamada de atenção **muito importante**: os **limites das variáveis não são testados**.
- Quando corremos um programa podemos, com uma certa facilidade, passar esses limites e escrever (ou ler) **noutras zonas da memória**. Os resultados são **imprevisíveis** e conduzem, muitas vezes, ao termo indevido do programa.

# Variáveis Aleatórios - Introdução

- Em certos problemas é necessário utilizar sequências aleatórias, isto é, uma sucessão de números escolhidos **ao acaso**.
- Na verdade, quando se fala em "**escolhidos ao acaso**", referimo-nos a **processos deterministas** que geram sucessões **aparentemente** aleatórias.
- Estas **sequências pseudo-aleatórias** são obtidas a partir de **funções de intervalo** em zona particulares dos **parâmetros**.
- Uma vez que os computadores são **deterministas** é necessário, cada vez que se **inicia um programa**, dar um **ponto de partida diferente** à sequência aleatória (caso contrário, começariamos sempre no mesmo sítio, o que só é bom na fase de testes).
- Um modo simples de termos um **ponto sempre diferente** é usarmos o **instante em que o programa começa** para definir o **ponto de partida** da sequência.

# Variáveis Aleatórios - Funções

Basicamente, duas funções em **C** são necessárias para obtermos uma **sequência aleatória** (essas funções estão definidas em '**stdlib.h**'):

- **void srand** (**unsigned int seed**);

  Serve para definir **internamente** em que ponto se **inicia** a sequência de números aleatórios (em geral, **usa-se uma só vez**);
  Para se ter **valores diferentes**, cada vez que se corre o programa, é usual dar-lhe como argumento o retorno da função '**time**' (o instante actual) que se encontra definida em '**time.h**':

  **srand** (**time** (**NULL**));

- **int rand** (**void**);

  Não tem argumentos e retorna um número inteiro entre **0** e **RAND_MAX** (2147483647). Para um **double** em $[0, 1]$:

  **x** = ((**double**) **rand** ()) / ((**double**) **RAND_MAX**);

Ver '**Prog08_01.c**' para '**int**' e '**Prog08_02.c**' para '**double**'.