Exercícios de

Física Computacional

(Parte 1)

Mestrado em Engenharia Física-Tecnológica (MEFT)

Rui Coelho

Departamento de Física do Instituto Superior Técnico

Ano Lectivo: 2019-20

Rui.alves.coelho@tecnico.ulisboa.pt

versão: 1 de Setembro de 2019

1.1 Revisão de C/C++

Exercício 1. Escreva um programa ***addnumbers.cpp*** que calcule a soma de dois números reais introduzidos pelo utilizador durante a execução do programa. O standard output deverá ser deste género:

```
--- Somar dois numeros ---
Primeiro numero : 36
Segundo numero : 53
Soma = 89
```

a) Use primeiro para leitura/escrita os "*objectos*" **cin** e **cout** da classe *istream*
b) Use agora as funções **scanf** e **printf** incluídas na biblioteca *cstdio*

Exercício 2. Escreva um programa ***pi_precision.cpp*** que determine o valor da constante π com precisão *float* e *double* a partir da função *atan()*. Deverá ter em atenção quantos algarismos significativos são impressos no écran.

Exercício 3. Escreva um programa ***check_limits.cpp*** que verifique os limites superior e inferior dos inteiros (signed, unsigned, long) e dos reais single e double precision. *Sugestão*: pesquise a biblioteca *climits e cfloat* !

Exercício 4. Cada tipo de variável ocupa um espaço diferente de memória. Escreva um programa em C++ ***bytes_check.cpp*** que calcule o número de bytes ocupado em memória pelas variáveis dos seguintes tipos: *char, short int, int, long int, unsigned int, float, double, long double*

Exercício 5. Escreva um programa em C++ ***sphere_volume.cpp*** que calcule o volume de uma esfera, com o raio dado pelo utilizador durante a execução, utilizando, em alternativa:

a) O programa main() para calcular o volume.
b) Uma função dedicada para o cálculo da volume.

Exercício 6*. Escreva um programa em C++ ***swap_first_last.cpp*** que, após introdução de um número inteiro pelo utilizador, troque o primeiro com o último dígito e.g. 6789 passe a 9786.

Exercício 7. Escreva um programa em C++ ***factorial.cpp*** que calcule o factorial de um número inteiro (n!) fornecido pelo utilizador utilizando:

a) Uma função dedicada para retornar o factorial
b) Uma função dedicada para o cálculo do factorial que use recursividade (função chamar-se a si própria).

Exercício 8. Escreva um programa ***prime_checker.cpp*** que verifique se um número inteiro introduzido pelo utilizador é um primo ou não.

Exercício 9*. Escreva um programa ***prime_decomposition.cpp*** que imprima no écran a decomposição em factores primos de um número inteiro introduzido pelo utilizador.

Exercício 10. Escreva um programa ***neper_taylor.cpp*** que calcule o valor aproximado (à 4 casa decimal) do número de Neper (2.718281...) através de um desenvolvimento em série de Taylor da função $f(x)=e^x$.

Exercício 11. Escreva um programa ***int2binary.cpp*** que converta um número inteiro em binário e imprima no écran a sua conversão em binário.

Exercício 12. Escreva um programa ***gravity2file.cpp*** que escreva num ficheiro ***trajectory.txt*** a trajectória (x,h(t)) de um objecto, lançado na vertical com velocidade v_0=20m/s, de uma altura h_0=5m. Considere a aceleração da gravidade dada por g=9.8m/s^2. O ficheiro deve ter 2 colunas i.e. tempo e altura, sendo os números representados com 4 casa decimais e com um total de 10 caracteres (*rever formatação de números*).

Exercício 13. Pretende-se verificar se é boa ideia usar iteradores em ciclos ***for*** que sejam números reais. Para tal, num programa ***real_iterators_check.cpp*** queremos calcular a soma dos números 0.1+0.2+0.3+.....+5.5 com o auxílio do seguinte código:

```
...
double sum=0;
for (double x=0; x != 5.5; x += 0.1) {
       sum += x;
}
```

Caso o programa não faça o que é pretendido, *descubra a razão*.
Sugestão: na biblioteca <math.h> existe uma função chamada *nextafter(a,b)* que determina o real que é representado a seguir a x=a seguindo em direcção a x=b e.g. *nextafter(2.0,5.0)* retorna o real que é representado a seguir a x=2 por valores maiores que 2.

1.2 Referências, ponteiros e alocação dinâmica

Exercício 14. Estenda o programa ***factorial.cpp*** que calculava o factorial de um número inserido pelo utilizador através de funções dedicadas (*declaração e implementação separadas*) mas retornando o valor:

a) Por referência i.e. `void factorial (int & fact, int n)`

b) Por ponteiro i.e. `void factorial (int * fact, int n)`

Exercício 15. Pretende-se escrever um programa ***test_summation.cpp*** que implemente uma função *SUM* que retorne por referência a soma de todos os números inteiro de 1 a N sendo N introduzido opcionalmente pelo utilizador. Caso o utilizador não seja chamado a introduzir nenhum valor de N, o programa deve retornar o valor -1.

Nota: A função *SUM* deve estar implementada em ficheiro separado e com *header* file próprio.

Exercício 16. No programa que se segue fazem-se *calls* às funções *fintv* e *fdoublev* que retornam ponteiros para arrays de inteiros e double respectivamente, cuja dimensão é dada no argumento das funções. (*Barão 2016*)

```
int main() {
int *a = fintv(100);
double *b = fdoublev(100);
}
```

As funções devem ser implementadas autonomamente em ficheiros separados ***fintv.cpp*** e ***fdoublev.cpp***. Uma implementação possível da função *fintv* poderia ser a seguinte:

```
int* fintv(int n) {
int v[n];
return v;
}
```

a) Verifique se o exemplo de código está funcional e em caso negativo, corriga-o e complete com a função que falta. ***Para a verificação, crie arrays de 10 componentes com valor igual ao índice do array***.
b) Com base nas funções anteriores, realize novas funções *fintv* e *fdoublev* que permitam a criação de tensores de até dimensão 3. Coloque funcional o seguinte programa *main*.

```
int main() {
// retornar uma matriz de inteiros de dimensão 100x50 inicializados a 1
int ***a = fintv(100,50); //declaração correcta, quantos elementos na 3ª dimensão?
// retornar um tensor de double de dimensão 100x50x2 inicializados a 5.
double ***b = fdoublev(100, 50, 2);
}
```

c) Realize as funções que façam o printout para o ecrã dos valores dos tensores

```
void print(int***, ...);
void print(double***, ...);
```

d) No final do programa main apague a memória alocada.

Exercício 17. *Justifique* através de um programa ***const_on_pointers.cpp*** qual a declaração correcta da seguinte função que nos deve permitir retornar por ponteiro um array (com N elementos) com a ordem dos seus elementos invertida.

a) `void flip(const double* const v, int N);`
b) `void flip(double* const v, int N);`

Exercício 18. Escreva um programa ***random100.cpp*** que gere 100 números aleatórios compreendidos entre 50 e 100. Deve existir uma função dedicada ***randgen*** que retorne apenas um número aleatório e o número de amostras já realizadas.

```
void randgen(double* num, ...);
```

Exercício 19. Realize um código C++ no qual se definam funções que realizem as seguintes tarefas:

a) calcular e retornar o traço da matriz

```
double Trace(int** mx, int n);
```

b) retorne um array com os elementos da linha-i da matriz m x n

$$M = \begin{bmatrix} 1 & 2 & 7 \\ -2 & 6 & -3 \\ 3 & 9 & 3 \end{bmatrix}$$

```
int* Mrow(int i, int** mx, int m, int n);
```

c) retorne um array com o resultado da multiplicação de uma matriz M(n x m) por um vector coluna de V (m) elementos. Utilize a matriz M da alínea b) e o vector abaixo.

$$V = \begin{bmatrix} 2 \\ 3 \\ -2 \end{bmatrix}$$

d) retornar o produto de 2 matrizes quando de dimensões compatíveis. Efectue a multiplicação NxM com M dada na alínea b) e N dada por

$$N = \begin{bmatrix} 2 & 3 & 1 \\ 3 & 0 & -2 \end{bmatrix}$$

1.3 Biblioteca STL

Exercício 20. Escreva um programa ***avgcircle.cpp*** onde implemente uma função *rand2vec* com declaração dada abaixo que retorna um vector de N-números complexos X_i com i=0,...,N-1, distribuídos aleatoriamente numa circunferência de raio=1. Calcule a média desse vector, entendida como o número complexo X_M=(<Re(X)>, <Im(X)>)

```
vector<complex<double> > rand2vec(int N);
```

Experimente para diferentes valores de N. O que pode concluir do resultado à medida que N aumenta ? Comente o resultado.

Exercício 21. Realize uma função array2vec e o respectivo programa main(), cujo objectivo é transferir um array de números inteiros para uma estrutura STL vector. (*Barão 2016*)

```
vector<int> array2vec(int, int*);
```

a) aplique a função aos arrays seguintes: **a**=(1, 10, 5, 6, 9, 3) e **b**=(2, 5, 5, 7, 3)
b) elabore numa função *array2vecs*, utilizando a biblioteca *<algorithm>*, a seriação dos valores de cada vector, quer na ordem crescente, quer na ordem decrescente. Retorne o vector ordenado para os arrays **a** e **b**.

```
vector<int> array2vecs(int n, int* a, int order); //order=-1 or 1
```

c) elabore uma função *array2vecmax*, que determine o valor máximo existente em cada um dos arrays **a** e **b** (converta-os para vector no interior da função e use os métodos apropriados da classe vector).

```
int array2vecmax(int n, int* a);
```

d) elabore uma função *array2vecfind*, que localize a posição do valor 7 em cada um dos arrays (converta-os para vector no interior da função e use os métodos apropriados da classe vector)

```
int array2vecfind(int n, int* a, int value);
```

e) realize a desalocação de memória que tenha utilizado antes do programa terminar.

Exercício 22. Neste exercício pretende-se explorar os containers vector e map da biblioteca STL (*Barão 2016*).

a) O programa main (incompleto) que se segue implementa um vector que conterá elementos da tabela periódica. Cada elemento é descrito por uma estrutura de dados ATOM.

```
int main() {
// fill with 1st 4 elements
ATOM hydrogen;
hydrogen.A = 1;
hydrogen.Z = 1;
hydrogen.name = "Hydrogen";
//allocate and fill vector
vector<ATOM> vperiodic(4); //4 elems allocated
...
```

```
// print the contents of every element of the vector
...
return 0;
}
```

Complete o programa *main* de forma a executar as accões descritas no programa, nomeadamente:

- Escrever uma estrutura ATOM num ficheiro header ***atom.h*** que contenha os dados de um elemento da tabela periódica.
- Incluir num vector os primeiros 4 elementos da tabela periódica.
- Imprimir no ecrã os detalhes de cada elemento contido no vector

b) Escreva agora um novo programa *main* que faça a gestão dos 6 elementos com um *map*, realizando as seguintes acções:

```
int main() {
// criar mapa
map< string, ATOM > mperiodic;
// preencher mapa com os 1os 6 elementos
...
// imprimir no ecrã todas as entradas do mapa
...
}
```

Exercício 23. Neste exercício pretende-se explorar o container vector da biblioteca STL na manipulação de matrizes. Replique o Exercício 19 definindo matrizes como *vector<vector<int> >*. *Exemplo*: para definir uma matriz 3x5 iremos definir *vector<vector<int> >* M(3).