Exercícios de

Física Computacional

(Parte 2)

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 Outubro de 2019

2. Classes em C/C++

Exercício 24. Crie um programa que construa e teste as operações aritméticas com números complexos. Utilize variáveis do tipo double para representar os dados privados da classe i.e.

```
private:
   double realPart;
   double imaginaryPart;
```

Deverá implementar os métodos: sum (soma), subtract (subtração), multi (multiplicação), divide (divisão), setComplex (recebe dois doubles como input), printComplex (imprime o complexo na forma "a+bi");

Exercício 25. Desenvolva uma classe ***Polynomial***. Nesta classe, um polinómio é representado por um "array "de termos. Cada termo contém um coeficiente e um expoente e.g., o termo $3x^5$ é representado pelo par (3,5). Use *vector* de *pair* para representar um polinómio.

A classe deve conter:

a) Construtores de default e construtor que aceite um vector de pair i.e.

```
Polynomial(vector<pair<int,int> >);
```

b) Operadores para a adição, subtração e multiplicação de Polinómios.

```
Polynomial operator+( const Polynomial& );
Polynomial operator-( const Polynomial& );
Polynomial operator*( const Polynomial& );
```

c) Operadores de copy constructor e assignement.

d) Operadores de *Print()* para imprimir o polinomio no ecrã e *Fill()* e para criar um polinómio, com coeficientes e expoentes fornecidos durante a execução do programa pelo utilizador

Caso ache necessário, pode definir funções auxiliares, externas à classe, para operar em *vector* de *pair* e.g. ordenação, eliminação de monômios com coeficiente nulo, etc.

Exercício 26. Verifique se o seguinte programa está correcto. Compilar e executar não é necessariamente condição suficiente...*Sugestão*: *Adicione código nos métodos que permita aferir quando cada método é invocado*.

```
#include <iostream>
using namespace std;

class Base {
public:
  Base() {
    i = new int;
  }
  ~Base() {
    delete i;
  }
private:
        int* i;
};

class Derived : public Base {
```

```
public:
  Derived() {
    f = new float;
  }
  ~Derived() {
    delete f;
  }

private:
  float* f;
};

int main() {
  Base* obj = new Derived;
  delete obj;
  return 0;
}
```

Exercício 27. Construiu-se uma classe genérica ***pessoa*** que pretende possuir as características associadas às pessoas (entendidas aqui como objectos!). A declaração da classe é a seguinte: (adaptado de *Barão 2016*).

```
class pessoa {
  public:
  //constructor (nome do aluno, data de nascimento - AAAAMMDD)
   pessoa(string, unsigned int); //queremos ter default values. Como ?
   void SetName(string); //definir o nome...
   void SetBornDate(unsigned int); //agora a data de nascimento
   string GetName(); //obter o nome
   unsigned int GetBornDate();//agora a data de nascimento.
   virtual void Print(); // print do nome e data de nascimento.
  private:
   string name; //nome
   unsigned int DataN; //data de nascimento
}
```

a) Implemente o código associado aos function members da classe escrevendo ***sempre*** em cada método o código necessário que imprima o nome da classe e do método `[class::method]` de forma a sabermos quando é chamado (*recordar slides teórica !*). Compile o código e veja se não existem erros.

b) Para testar o código da classe realize um programa main onde construa um array de 5 objectos pessoa (usando valores de default da classe) através de:

```
pessoa P[5];
```

Verifique que construtor é chamado. Corrija o código e declaração da classe caso existam erros.

c) Admita agora que pretendia construir um "array" de N pointers para objectos pessoa. Construa uma função, *externa à classe*, que retorne o ponteiro para o array.

```
pessoa** DoArray(int N);
```

Preencha o ponteiro com 5 pessoas quaisquer (alunos ficcionais, a usar no exercício 28). Faça a desalocação de memória usando função apropriada.

Exercício 28. Construa agora uma classe ***alunoIST*** que derive da classe ***pessoa***. A nova classe deverá ter novos *data members (adaptado de Barão 2016)*:
- Número do aluno: *int* number
- Curso: *string* course
- funções que interajam com os novos data members.

```
class alunoIST : public pessoa {
  public:
   //constructor (numero e curso do aluno)
   alunoIST(int number, string curso);
   void SetNumber(int);
   int GetNumber();
   void Print();
   ...
  private:
   int number;
   string branch;
}
```

a) Implemente o código da nova classe.
b) Construa um array de objectos alunoIST aproveitando as mesmas 5 pessoas do Exercicio 27 (*adicionando a informação de número de aluno e curso*).
c) Construa agora um ponteiro genérico para um array de pointers de objectos *alunoIST*,

```
alunoIST ** pAluno ...;
```

contendo a informação dos 5 alunos definidos na alínea b) e criei uma função,

```
//function prototype
void Dummy(alunoIST**, const int);  //int has the number of array entries
```

que receba um pointer genérico para um array de pointers de objectos *alunoIST* e que imprima no ecrã todos os objectos (*alunoIST*) chamando a função membro *Print*(). A função que é chamada pertence a que classe (*pessoa* ou *alunoIST*)?

Exercício 29. *(adaptado de Barão 2016)* Pretende-se agora criar a classe Turma que não necessita de derivar de nenhuma das classes anteriores, antes usando os objectos da classe *alunoIST*. Uma declaração ainda que incompleta da classe seria:

```
class Turma {
     public:
       Turma(string, int n); //nome da turma, num de alunos
       ~Turma(); //destructor
     private:
       alunoIST **va; //pointer to array de pointers de objectos
       int Nalunos;
}
```

a) Complete a declaração da classe de forma a incluir os seguintes métodos:
- default constructor
- copy constructor
- copy assignment
- métodos
    - void AddAluno(alunoIST* const)
    - alunoIST* FindAluno(int numero)
    - método int GetNalunos()

b) Implemente o código da classe e em particular o método `alunoIST* FindAluno(int)` deveria ser implementado da forma mais eficaz usando a *procura dicotómica*.

c) Elabore um programa ***Turma_teste.cpp*** onde possa testar a classe e seus métodos, introduzindo vários alunos (definidos como objectos da classe alunoIST)