Aula 11

Objectivo: Templates (continuação)

Problema 1

Continuando o problema 1 da aula anterior crie uma nova classe genérica: `ARRAY_ORDENAVEL`, que acrescenta a possibilidade de ordenação do "array" (logo só é aplicável a elementos que implícita ou explicitamente estabeleçam uma relação de ordem). Esta classe assume que a relação de ordem entre os seus elementos é feita recorrendo a uma função global (que tem de existir para cada instanciação de T):

```
int maior_do_que(T elem1,T elem2);
```

Esta nova classe deve ser descendente da classe genérica `ARRAY` e para além da interface desta deve incluir os seguintes serviços:

```
template <class T>
class ARRAY_ORDENAVEL // (...)
{
public:
// (...)

  void define_proc_ordenacao(void (*proc_ord)(ARRAY_ORDENAVEL &arr));
  void ordena(void);
  T maximo(void);
// (...)
};
```

Assim sem prejuízo de a classe estar à partida ligada a um algoritmo de ordenação especifico (por exemplo o de selecção), esta permite que este seja redefinido fazendo uso da possibilidade que existe em `C++` de utilizar variáveis (e atributos) do tipo ponteiro para funções.
(NOTA: A declaração de uma variável ponteiro para uma função `void` com uma argumento inteiro é feita da seguinte forma:
`void (*p_func)(int arg));`)

1.1. Experimente a classe para vários tipos de elementos (`int` e `char*`), e para vários algoritmos de ordenação.

1.2. Implemente e experimente o método `maximo`, que deve devolver o valor

máximo existente no "array".

Problema 2

Continuando o problema 2 da aula anterior, estude a possibilidade de também criar uma classe `LISTA_ORDENAVEL` (ou seja uma lista que pode não estar ordenada, mas à qual é aplicável um método de ordenação, de forma similar ao problema anterior).
(NOTA: Chama-se a atenção para o facto de a interface da classe `LISTA` não permitir o uso de índices, sendo esta apenas percorrível sequencialmente)