Aula 2

Objectivos: Definir e construir classes com encapsulamento em *C++*.
Construtores e destrutores de classes.

Problema A1

Pretende-se fazer um programa simples para construir e gerir uma lista de pessoas.

A1.1. Construa a classe `PESSOA` com a informação do nome, número de BI e a idade. A classe deve ter, para além de um construtor apropriado, pelo menos, um método que escreva na saída a informação da pessoa.
A estrutura da classe pode ser a seguinte:

```
class PESSOA
{
public:

  // construtor usando como parâmetros toda a informação necessária
  // escrever a informação da pessoa

private:

  char *p_nome;
  char *p_num_bi;
  int p_idade;
};
```

(**Nota:** A parte mais importante de uma classe é a sua interface pública. Por esta razão os nomes dos métodos públicos – e também a restante assinatura: nomes e tipos de argumentos caso existam – devem ser simples e fáceis de compreender. As abreviaturas poderão ser utilizadas se, no contexto da classe, o seu significado for perfeitamente claro.)

A1.2. Faça um programa (`main`) que exemplifique o uso da classe.

A1.3. Faça com que o armazenamento do nome e número de BI seja feito recorrendo a alocação dinâmica de memória, e acrescente um destrutor apropriado à classe.

A1.4. Construa uma classe que permita armazenar uma lista de pessoas. A estrutura da classe pode ser a seguinte:

```
#define NumeroMaxPESSOAS 1000

class LISTA_PESSOAS
{
public:

  // construtor
  // adicionar PESSOA
  // escrever a lista de pessoas

private:

  int p_num; // número de pessoas guardadas
  PESSOA *p_array[NumeroMaxPESSOAS]; // array de ponteiros para PESSOA
};
```

(**Nota:** Deve-se ser consistente na construção de programas por forma a simplificar a sua compreensão. Como exemplo, nas estruturas de classes apresentadas, o nome das classes é sempre em maiúsculas, o nome dos métodos públicos sempre em minúsculas, as palavras são separadas por "_" e os atributos privados começam sempre por "`p_`".)

A1.5. Faça um programa de gestão simples de uma base de dados de pessoas, que permita adicionar novas entradas à base de dados ou mostrar o seu conteúdo (sugestão: use um menu).

A1.6. Acrescente o serviço de remoção de uma PESSOA da colecção.

A1.7. Modifique a implementação da classe `LISTA_PESSOAS` mudando a representação interna de um *array* para uma lista ligada simples de PESSOA.

(**Nota:** Em *C++* os atributos de uma classe nunca devem ser públicos, por forma a impedir que o seu valor seja definido externamente à classe. Desta forma a classe pode controlar completamente a definição do seu estado, simplificando o seu uso e aumentando a sua modularidade.)

Problema A2

Faça um programa com funcionalidades semelhantes às do anterior, para gerir uma colecção de quadros de um museu. Assuma que cada quadro é identificado por um nome, um pintor, e o ano em que foi feito.

A2.1. Acrescente a possibilidade de inserir informação sobre cada pintor, incluindo a data de nascimento, o estilo de pintura em que se insere, e uma lista das suas obras.

Problema A3

A3.1. Construa uma classe que descreva a informação sobre alunos da Universidade de Aveiro.

A3.2. Construa uma classe que represente uma disciplina, de um qualquer curso da Universidade. Esta classe deve estar associada a uma lista de alunos nela inscritos.

A3.3. Faça um programa que permita para uma dada disciplina, a inscrição e remoção de alunos, a atribuição de uma nota, a apresentação de uma nota de um qualquer aluno (inscrito), e a apresentação de uma pauta final da cadeira (assumindo que apenas se está interessado na nota final de cada aluno).

Problema B1

B1.1. Construa uma classe `CComputer`.

B1.2. A classe `CComputer` deve ter a seguinte estrutura:

```
class CComputer
{
public:
  void Display();  // imprimir a informação acerca do computador
  CComputer (/*argumentos*/);  //construtor
  CComputer (); // construtor (por defeito)
  ~CComputer (); // destrutor

private:
  unsigned  frequencia_do_barramento;
  unsigned  frequencia_do_processador;
  unsigned long  tamanho_do_HD;
  unsigned  tamanho_de_RAM;
};
```

B1.3. Construa a função main que pode usar alguns objectos do tipo `CComputer`, por exemplo:

```
int main(int argc, char* argv[])
{
  CComputer c1(100, 800, 15, 128);
  CComputer c2(133, 1000, 30, 256);
  CComputer c3(100, 900, 20, 512);

  c1.Display(); c2.Display(); c3.Display();

  return 0;
}
```

Problema B2

B2.1. Construa uma classe `CStaticStack` que representa uma pilha de computadores de tamanho fixo.

B2.2. A classe `CStaticStack` deve ter a seguinte estrutura:

```
typedef CComputer stack_item;
class CStaticStack  // pilha de computadores
{
public:
  CComputer Pop ();
  void Push (stack_item);
  Ccomputer LookAtTop ();

  //construtor
  //destrutor

private:
  unsigned m_uSize;    // tamanho da pilha
  stack_item* m_pMem;  // memória reservada para a pilha
  stack_item* m_pTop;  // topo da pilha
};
```

B2.3. Faça um programa que exemplifique o uso da classe `CStaticStack`.

Problema B3

B3.1. Construa uma pilha de caracteres `CStack` que elimine as restrições de tamanho. Uma solução possível é fazer com que cada nodo da pilha para além de guardar os dados (um caracter) guarde um ponteiro para o nodo seguinte.

```
typedef char StackItemType;
struct StackNode;
class CStack
{
public:
  // construtor e destrutor
  // operações sobre pilha

private:
  StackNode* m_pTop; // ponteiro para o primeiro nodo da pilha
};
-------------------------
struct StackNode // um nodo da pilha
{
  StackItemType item; // os dados a guardar na pilha
  StackNode* next;    // ponteiro para o nodo seguinte
};
```

B3.2. Utilize a pilha criada para verificar se uma linha de caracteres (fornecida através da linha de comando `Project -> Settings -> Debug -> Program Arguments`) contém os parêntesis equilibrados. A linha deve ser analisada caracter a caracter de esquerda para a direita. Os parêntesis estão equilibrados se:

- Cada parêntesis ")" corresponde a um parêntesis "(" encontrado previamente na linha;
- Quando atingir o fim da linha é encontrado um par para cada parêntesis "(".

Exemplos:

1. Os parêntesis na linha "`abc ( defg (ijk) (l (mn) ) op ) q`" estão equilibrados.
2. Os parêntesis na linha "`abc (def) ) g ( ijk (l)`" não estão equilibrados.