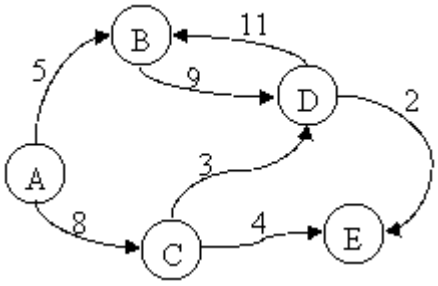
3ªaula prática - Tratamento de Exceções. *Templates* de Classes

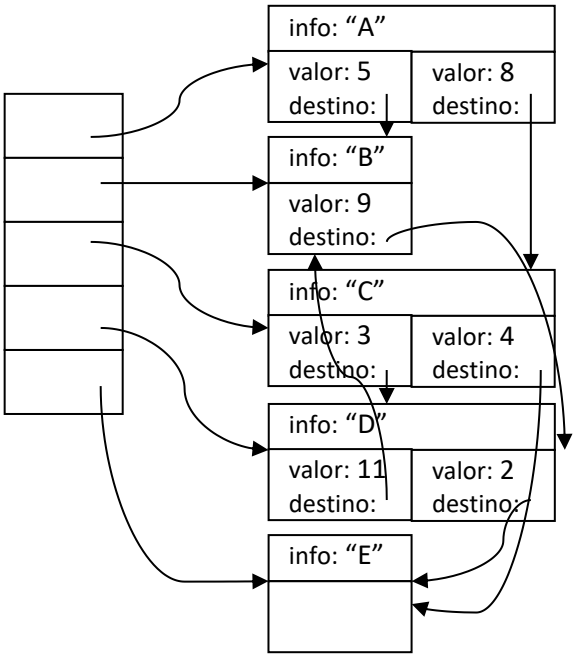
- Faça download do ficheiro ***aeda1920_fp03.zip*** da página da disciplina e descomprima-o (contém a pasta ***lib,*** a pasta ***Tests*** com os ficheiros ***grafo.h*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***)
- Note que os testes unitários deste projecto estão comentados. Retire os comentários à medida que vai implementando os testes.
- Deverá realizar esta ficha respeitando a ordem das alíneas.
- Deve fazer a implementação no ficheiro ***grafo.h***.

Enunciado

A classe **Grafo** permite representar um grafo orientado, composto por **nós** ligados por **arestas**. A informação contida nos nós e arestas do grafo pode estar associada a tipos de dados diferentes. A classe **Grafo** é uma classe genérica com dois argumentos, os **nós** e as **arestas**. Considere que todos os **nós** do grafo são diferentes.

Cada instância da classe **Grafo** contém um **vetor de apontadores para nós**. Para cada nó, existe um **vetor de arestas** (ordenadas segundo o nó de destino). A figura seguinte mostra a estrutura de dados para um exemplo.





A declaração da classe **Grafo** é a seguinte:

```
template <class N, class A>
class No{
public:
	N info;
	vector< Aresta<N,A> > arestas;
	No(N inf) {
		info = inf;
	}
};

template <class N, class A>
class Aresta {
public:
	A valor;
	No<N,A> *destino;
	Aresta(No<N,A> *dest, A val) {
		valor = val;
		destino = dest;
	}
};
```

```
template <class N, class A>
class Grafo {
      vector< No<N,A> *> nos;
  public:
    Grafo();
    ~Grafo();
    Grafo & inserirNo(const N &dados);
    Grafo & inserirAresta(const N &inicio, const N &fim, const A &val);
    Grafo & eliminarAresta(const N &inicio, const N &fim);
    A & valorAresta(const N &inicio, const N &fim);
    int numArestas(void) const;
    int numNos(void) const;
    void imprimir(std::ostream &os) const;
};
```

A implementação deve ser efetuada no ficheiro *grafo.h*.

a) Implemente:

- O construtor e o destrutor da classe *Grafo*.
- O método *numNos()* (que retorna o número de nós do grafo).
- O método *numArestas()* (que retorna o número de arestas existentes no grafo).

b) Implemente o membro-função *inserirNo(const N &dados)*, que insere um novo nó no grafo e retorna o grafo alterado (*this*). Esta função deve lançar a exceção *NoRepetido* caso esse nó já exista (ver teste unitário para esta alínea).

A exceção *NoRepetido* já está implementada.

c) Implemente o membro-função *inserirAresta(const N &inicio, const N &fim, const A &val)*, que insere uma nova aresta no grafo e retorna o grafo alterado (*this*). Esta função deve lançar a exceção apropriada caso a aresta já exista.

- Exceção *NoInexistente*: esta exceção já está implementada.
- Exceção *ArestaRepetida*:
  - Implemente esta exceção. Implemente o operador <<, que imprime no monitor os valores dos nós extremos da aresta

d) Implemente o membro-função *valorAresta*(*const N &inicio, const N &fim)*, que retorna uma referência para os dados da aresta especificada. Esta função deve lançar a exceção apropriada caso a aresta não exista no grafo (ver teste unitário para esta alínea).

- Exceção *ArestaInexistente*:
  - Implemente o operador <<, que imprime no monitor os valores dos nós extremos da aresta

e) Implemente o membro-função *eliminarAresta(const N &inicio, const N &fim)*, que elimina uma aresta do grafo e retorna o grafo alterado (*this*). Esta função deve lançar a exceção apropriada caso a aresta não exista no grafo (idêntica à da alínea anterior).

f) Implemente o membro-função *imprimir*(*std::ostream &os)*, que escreve, para um *stream* de saída, a informação do grafo. Para o exemplo indicado anteriormente, a função deve produzir:

(A [B 5] [C 8]) (B [D 9]) (C [D 3] [E 4]) (D [B 11] [E 2]) (E)

g) Utilize a função anterior para implementar o operador de saída <<.

h) Efetue a documentação dos membros-função implementados (use Doxygen).