**Practical work no. 1 - example**

**Specification**

We shall define a class named Graph representing a *directed graph*.

We need some auxiliary clases:

***Graph::Edge\_id***

an edge identifier, for referring to an edge from outside the Graph class (can be thoght as an edge index or as a pointer to the internal structure representing an edge).

***Graph::Edge\_incident\_iter***

an edge iterator, for iterating through the edges incident to a given vertex. Convertible to Graph::Edge\_id

The class Graph will provide the following methods:

***explicit Graph(int n)***

Constructs a graph with *n* vertices and without arcs.

***int count\_vertex() const***

Returns the number of vertices

***int insert\_new\_vertex()***

Inserts a new vertex and returns its number.

***Vertex remove\_vertex(int v)***

Removes the vertex given *v*

***Graph::Edge\_id insert\_new\_edge(int v1, int v2)***

Inserts an arc from vertex *v1* towards vertext *v2*. **Precondition:** there must be no arc from *v1* to *v2*.

The rest of the member functions, alogn with the public member functions of the helper classes Edge\_id and others go here.

**Implementation**

The implementation uses two auxiliary structs:

struct Edge { // represents an arc of the graph

int source; // index of source vertex

int target; // index of target vertex

struct Edge\* next\_source; // next edge from the same source

struct Edge\* prev\_source; // previous edge from the same source

struct Edge\* next\_target; // next edge to the same target

struct Edge\* prev\_target; // previous edge to the same target

int label; // label attached to the graph (length or so)

};

struct Vertex { // represents a vertex

struct Edge\* first\_in; // pointer to the first inbound arc; null if none

struct Edge\* first\_out; // pointer to the first outbound arc; null if none

};

Each edge is member of two double-linked lists, a list of all edges that are outbound from the same vertex, and a list of all edges inbound to the same vertex. Each vertex has a pointer to one of the edges in each list.

Class Graph will have the following data members:

***int n***

represents the number of vertices

***std::vector vertices***

the vertices

Edge\_id and the others shall be documented here as above. Helper (private) functions may be documented here.