Programming for Egineers and Scientists

C++: Part 1

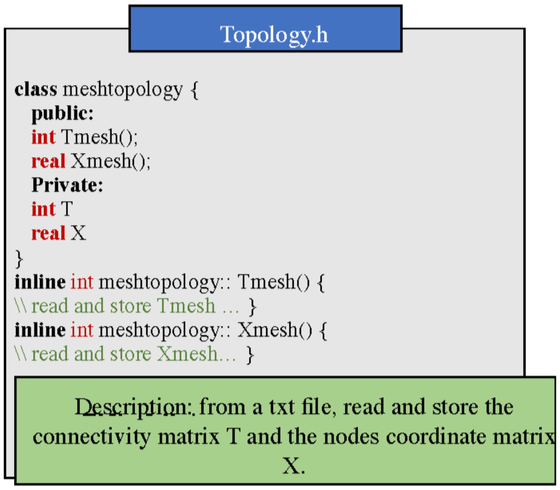
Domingo Eugenio Cattoni Correa <domingocattoni@gmail.com>

Juan Diego Iberico Leonardo <ibericoleonardo@hotmail.com>

Oriol Call Piñol <oriolcall1@gmail.com>

Introduction

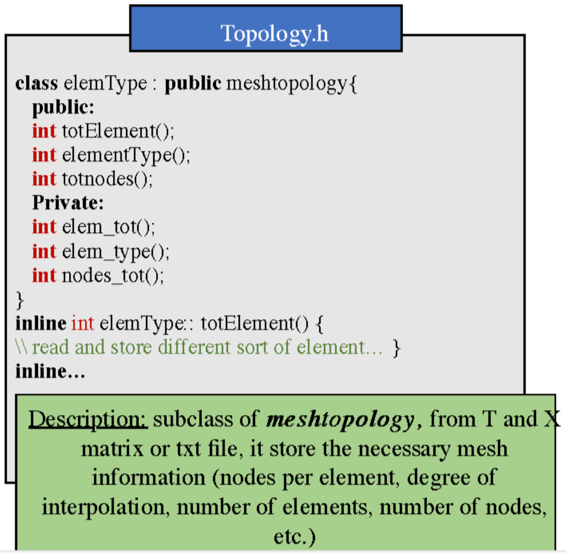
In the present document it is developed the main structure of the program. It is based in the decomposition of the necessary functions as well as structures where the variables are saved by packages. The main functions of the code we are going to be developing are the following:



Topology.h

```
class meshtopology {
   public:
   int Tmesh();
   real Xmesh();
   Private:
   int T
   real X
}
inline int meshtopology:: Tmesh() {
\\ read and store Tmesh … }
inline int meshtopology:: Xmesh() {
\\ read and store Xmesh… }
```

Description: from a txt file, read and store the connectivity matrix T and the nodes coordinate matrix X.



Topology.h

```
class elemType : public meshtopology{
   public:
   int totElement();
   int elementType();
   int totnodes();
   Private:
   int elem_tot();
   int elem_type();
   int nodes_tot();
}
inline int elemType:: totElement() {
\\ read and store different sort of element… }
inline…
```

Description: subclass of ***meshtopology***, from T and X matrix or txt file, it store the necessary mesh information (nodes per element, degree of interpolation, number of elements, number of nodes, etc.)

Boundarycondition.h

```
class Bcondition {
   public:
   real N_Bc();
   real D_Bc();
   Private:
   real NBc
   real DBc
}
inline Bcondition:: N_Bc() {
\\ read and store the edge and value where it is applied}
Inline meshtopology:: D_Bc() {
\\ read and store node and value where it is applied… }
```

Description: from a txt file, read and store the boundary conditions, for Neuman it stores in a matrix the edge (two nodes) where the value it lie and for Dirichlet the node where it is prescribed

Element.h

```
class shapeFunction{
   public:
   Real N(), Nξ(), Nη(),Nx(),Ny();
   Real Jacob(), invJacob(), detJacob();
   Private:
   Real gaussPoints(wgi,zgi);
   int elem_type();
   int nodes_tot();
}
inline shapeFunction:: N() {
\\ Define shape function accoridng to degree of int.}
inline…
```

Description: The present class has the aim to map from the reference coordinate to cartesian system.

Element.h

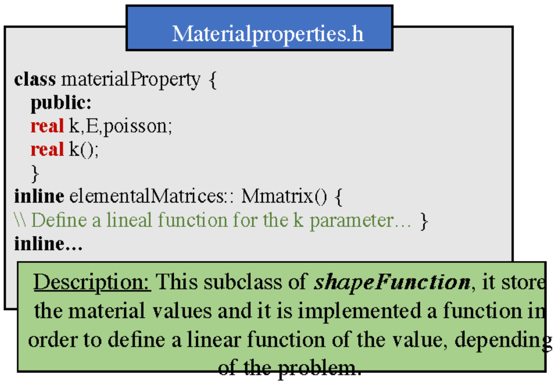
```
class elementalMatrices {
   public:
   real Kmatrix(), Mmatrix(), Cmatrix();
   }
inline elementalMatrices:: Mmatrix() {
\\ Compute the mass matrix  … }
inline…
```

Description: This subclass of ***shapeFunction***, it compute in different functions the elementals matrices such as stiffness, convective or mass matrix.

Element.h

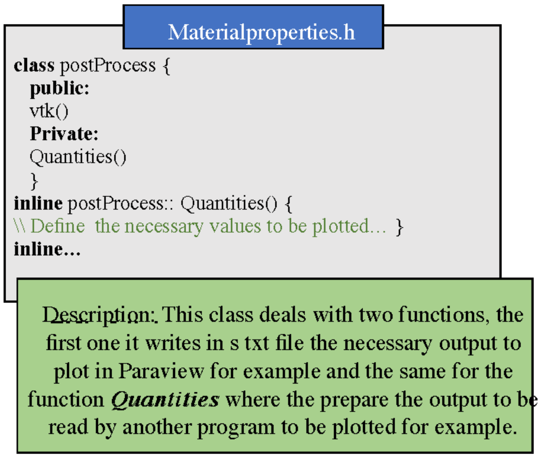
```
class elementalForce{
   public:
   Real fVector();
   }
inline elementalForce:: N() {
\\ Compute the R.H.S of the system.}
inline…
```

Description: The present subclass of ***shapeFunction*** it compute the R.H.S of the system through the function **fVecotr** defined in the subclass **elementalForce**.



```
class materialProperty {
   public:
   real k,E,poisson;
   real k();
   }
inline elementalMatrices:: Mmatrix() {
\\ Define a lineal function for the k parameter… }
inline…
```

Description: This subclass of ***shapeFunction***, it store the material values and it is implemented a function in order to define a linear function of the value, depending of the problem.



```
class postProcess {
   public:
   vtk()
   Private:
   Quantities()
   }
inline postProcess:: Quantities() {
\\ Define  the necessary values to be plotted… }
inline…
```

Description: This class deals with two functions, the first one it writes in s txt file the necessary output to plot in Paraview for example and the same for the function ***Quantities*** where the prepare the output to be read by another program to be plotted for example.

As we can see, all the different functions have their specific task. We also need to point out that all of them belong to different classes such as the ones we are going to explain right below.

Structure

The structure for this different functions is organized by classes and subclasses.

The class “meshTopology” has a subclass called “elementType” and they both are connected to the header file “Topology.h”.

The class “bCondition” is associated to the header file “Boundarycondition.h”.

The class “shapefunction” has two subclasses called “elementalMatrices” and “elementalForce” respectively. This two classes and their mother (shapeFunction) are linked to the header “Element.h”.

The last two structures that will form our code are the class “materialProperty” which is related to the header “Materialproperties.h” and “postProcess”, which is connected to the header “Postprocess.h”.

The structure above will be enough to carry out the different tasks stated by the problem.

Diagram:

The first element is the header, the second is the class and the the last part are subclasses, only for some of them.

Topology.h → meshTopology → elementType
Boundarycondition.h → bCondition
Element.h → shapeFunction → (elementalMatrices, elementalForce)
Materialproperties.h → materialProperty
Postprocess.h → postProcess

Conclusion

In addition to everything said before, this assignment has been an introduction to the main assignment about coding a whole Finite Element code using the programming language C++.
This first approach was useful to start understanding how the different structures work in this language. The main difficulties found during this homework were trying to comprehend how these were organized and which was the best or most suitable way to construct with them.