Rules of programming

Summary:

This document presents the rules retained by the development team of Aster (EDA) for the writing of the source code of Code_Aster.

The compliance with these rules has two main aims:

- to ensure a good portability of the code,
- to ensure a good legibility (and thus maintainability) of the source text.

The control of the compliance with the rules of programming is carried out during the tender of the developments (during the execution of the orders `Hg made` and `Hg submit`) by the tool `aslint` (deposit `devtools` who can also be used in an autonomous way).
The errors (messages of the type E or C like *error* or *convention*) emitted by aslint must imperatively be corrected by the developer.
Alarms (messages of the type W like *warning*) must be removed as much as possible for the improvement of the code and do not have to be only accepted if it is not possible to make differently.

This document provides elements complementary to the messages of the type *warning* emitted by `aslint` to explain the objective of it.
One also described there conventions or advices which cannot be checked automatically.

1 Automatic checks

The whole of the checks carried out by `aslint` are indexed:
- in-house EDF: http://aster-services.der.edf.fr/mercurial/messages_index.html
- on Internet: https://bitbucket.org/code_aster/codeaster-src/wiki/AslintMessagesIdentifiers

Each message is identified by a followed letter by four digits.

The letter determines the gravity of the messages:
- E means *error* : it is an error which one cannot be unaware of;
- C means *convention* : it is an error by convention. For example, an alarm of the compiler which one considers too serious in code_aster.
- W means *warning* : it is usually possible to remove them by respecting good practices;
- I means *information* : it is simply about information on the analysis of the source.

The first two figures are used to indicate the language concerned and the type of messages.

One does not detail here the various messages raised by `aslint`.
It is imperative to correct the errors E. One does not have the choice!
It is imperative to correct the errors by Convention.
It is strongly recommended to eliminate a maximum of Warnings to improve the code.

2 Complementary rules

2.1 Modules FORTRAN

During compilation, it is necessary to compile the modules in the order imposed by their interdependences, and then, to compile the routines which use these modules.

To identify quickly and in an automatic way the modules, the following naming is thus forced:
- `xxxx_type.F90` for the modules which define types derived (a type by module) and the elementary methods for creation from the objects of this type;
- `xxxx_module.F90` for the “methods” on the types.

All the derived types must be documented (like are the structures of Jeveux data).

2.2 Constants

One avoids the abundant use of `#define`.
Parameters should rather be defined (`parameter`) gathered thématiquement in a module.

3 To circumvent the law

During the tender of developments (with `Hg submit`), it happens that one wants to pass in addition to the emission of certain error messages (by convention, errors C) for one ***good reason***.
It is possible to make so that the errors are degraded into simple warnings.

For that, one writes in the file `~/.hgrc`, in the section `[aster]` :

```
check.submit = warn
```

It will then be necessary to explain to the integrator *good reason* who justifies this skirting.