Coding Standards

Edward Hall

University of Nottingham

Coding standards

Coding standards are basic rules for programming that are stipulated between programmers. Coding standards

- Promote code readability, portability and extensibility.
- Facilitate code maintenance.

Some standards dictate how programs should be laid (where to put comments, how to use new lines, etc.). Other rules refer to naming conventions for variables, classes and functions. Other rules outlaw various programming practices that, albeit legal in the language, are considered dangerous.

Throughout this module we follow many of the coding standards outlined in Pitt-Francis & Whiteley (Section 6.6).

Coding standards

Coding standards

1. Code within functions, `if`, `while` and other blocks is indented. Curly braces are always used and appear on a line of their own.
2. Lines of codes that are too long are split across multiple lines. How much is too long? 80 columns seems to be a standard width.
3. Names for variables are meaningful, for instance `width` and `height` rather than `w` and `h`. We occasionally break this rule when dealing with mathematical software for which short names are unambiguous. For instance, it may be acceptable to use `x` in the method `Solve` of a `LinearSystem`.
4. Variables are declared close to where they are used, rather than at the beginning of a function. This is a convention adopted mostly by `C++` programmers, so you may find it confusing if you programmed in `Fortran` before. For instance

```
double x, residual;
for (int i=0; i<10; i++)
{
  x = x - ( exp(x) - 1 )/ exp(x);
  residual = exp(x) - 1;
}
```

Coding standards

Coding standards

5. Locally declared variables have underscores, for instance `grand_total`.
6. When pointers are declared, the `*` is written adjacent to the variable type, without spaces in between. When pointers are dereferenced the `*` is written adjacent to the variable name, without spaces in between. Hence

   ```
   double* p_solution = new double;
   *p_solution = 1.0;
   ```

   This helps disambiguating the meaning of `*`. This convention rules out the possibility to declare multiple pointers on one line, such as

   ```
   double *p_solution, *p_residual;   // Valid C++ code, not allowed by our coding standard
   double* p_solution;                // Valid C++ code, allowed by our coding standard
   double* p_residual;
   ```

   The same convention is used for `&` and reference variables.

Coding standards

Coding standards

7. Pointers begin with the letter “p”, for instance `p_solution` (for a locally declared variable) or `pSolution`.
8. Function names are in camel-case and the first word is a verb (to stress that they *do something*). For instance `ComputeArea()`, or `InitialiseSolver()`.
9. Names of arguments to functions and class methods are also in camel-case, but they begin with a lowercase letter, such as `dimension`, `pSolution` and `pResidual` in this code

```
void CalculateResidual(const int dimension,
                       const double* pSolution, double* pResidual) const
```

10. Private or protected members of a class have camel-case names and their first letter is “m” for “my”. Hence if we read `mSize` and `mpSolution` in a class method, we immediately distinguish them from method variables. See another example below.

```
void SetLength(const double& length)
{
  mLength = length;
}
```

Coding standards

Coding standards

11. Class names are also camel-case and begin with an uppercase letter, for example `Vector` and `ComplexNumber`.
12. Codes contain descriptive comments