# Coding standards

The purpose of the coding standard is to maximize the readability of the code. The code per se will not be documented other than comments in the coding diary. Therefore it is imperative that the code is unambiguous and easy to read.

Readability has the following advantages:

* The coding process gets faster as programmers can understand each other’s code
* Programmers will spend less time rereading their own code that they have written earlier
* The testing and debugging process is simplified
* It is easier to bring new programmers onto the project
* The code is easier to audit

## Variables

Variable names should be chosen with care to truly reflect their purpose. Always chose a long name that explains the variables purpose rather than a short name that is easy to type (e.g. Int i). All names should be in English (meaning English and not Swinglish) except: database column, database table and xml tag names. The reason for the exception is that these names are set in TRFS 2016:47 in the appendices. Of course the same goes for structs, enums, classes etc.

Limit the scope of your variables if unsure declare a variable as private until tests prove it needs a larger scope

## Fixed values

Fixed values should be made into constants again with meaningful names. For example (const int WorkdaysPerWeek = 5.) A fixed number of options should be an enum.

## Comments

Do not comment code. If you feel the need to comment then the code is not readable. Rewrite it so that is it. The only comments in the code are to be references in the coding diary. The coding diary should not either be used to explain code but to inform the reader about important architectural decisions mad in the code and what impact they may have on other parts of the system.

## Functions

Function are to be given meaningful names just as variables. Function names need to be consistent for example if you have a function getDatabaseValue for retrieving a value from a database do not name the next function readDatabaseRow to retrieve a whole row. Make a decision is retrieving database data getting or reading.

Do not retrieve values from a function or sub routing with “out” or referenced parameters it is confusing.

When calling a function you should ideally only have one parameter going in. Multiple parameters make the function harder to understand. If you need multiple parameters create a structure that has a meaningful name the reader can identify with more easily.

## Formatting and organization

Always take the time to indent your code properly.

When a function calls another function position the functions it calls beneath and as close to the calling function this makes the code faster to read.

Make functions small and limit a function to doing one thing. It is difficult to give a meaningful name to a function that does many things. It is also difficult to understand a function that contains a mass of text.

## Classes & Structs

Classes should expose functionality and hide their variables. Put functionality not structures in classes. Data structures format data and do not have functionality organize variables and data in structures. Limit the scope of your classes so that they do one logical thing.

Just as with functions a class should manipulate one thing and have a single purpose. If you have difficulty keeping your class small then its doing too many things. Break it down into more clases.