# Coding Style

## General Recommendations

The main goal of the recommendations is to improve the readability, understanding and maintainability of the source code. Violation to this recommendation is allowed only if it enhances the readability of the source code.

## Header Files

Every .cpp file should have an associated .h file. Files should be saved in the same directory. An exception to this can be the main.cpp file, which does not have to have an associated .h file and other small files containing unit testing can be excluded from having an associated .h file.

All .h files must be included in the ResourcePath.hpp file. This allows to keep a record of all includes in one place. Every .cpp file will only need to have

*#include “ResourcePath.hpp”.* This will then be able to lookup any .h file that it needs.

## Include Statements

Include statements should be sorted and grouped. Sorted by their hierarchical position in the system with low level files included first. Leave an empty line between groups of include statements.

Include statements should be located at the top of a file only.

Example:

*#include<iostream>*

*#include<fstream>*

*#include<cmath>*

*#include<iomanip>*

## The #define Guard

All header files should have the #define guard to avoid multiple inclusions of declarations.

Formatting for the #define guard should be <*PROJECT>\_<PATH>\_<FILE>\_H\_*

## Naming Conventions

1. Names representing types must be in mixed case starting with upper case. Example: *Name, FileName*
2. Variable names must be in mixed case starting with lower case.

Example: *name, filename*

1. Constants must be all upper case using underscore to separate words.

Example: *MAX\_NAME, PI*

1. Names representing methods should be verbs and written in mixed case starting with lower case.

Example: getName( ), computeArea( ), setName( )

1. Names of classes should begin with an uppercase letter.

## Variables

1. Variables must be initialized where they are declared.
2. Variables must not have dual meaning.
3. Variable names must be meaningful.
4. Use of global variables is not permitted; in C++ there is no need for global variables.
5. Class variables must always be declared as private members.

## Functions

Functions should be kept as small as possible. When a function reaches roughly 40 lines look at separating it into smaller parts.

However there is no hard limit to the size of a function, it is unavoidable at times for a function to grow in size this can be acceptable if there is no alternative method of breaking it into smaller parts.

## Tabs, Spacing and Indentation

Tabs must be set to four characters.

The bracing style must follow the style of where the opening brace begins on the same line of the statement and ends on a new line.

Example;

If (a < b) {

a = b;

}

Classes must follow the same format for bracing, the opening brace begins on the same line of the class and the closing brace is on a new line.

The labels for *private:* and *public:* are not indented.

Example;

Class MyClass {

public:

MyClass( ); // Constructor.

~MyClass( ); // Destructor.

void myMethod ( );

private:

int year;

};

## Commenting Your Work

Comments should be used to explain what a section of code does if it is not clear.

It is preferable that if the code is too complex to understand that it should be re-written.

Single line “//” commenting is to be used even for multi line commenting.

Comments must have a space after “//” and start with upper case and end with a period.

Block commenting “/\* \*/” should be used primarily for commenting out sections of code for debugging purposes.