# Coding Standard

## Origin of coding standard

These standards are heavily influenced by the standards used by Epic in the Unreal Engine. This is a well established code base with a published coding standard. The standard has been tried and tested across many development studios and projects, and has proven to be a clear and workable standard on a large codebase of several million lines of code.

<https://docs.unrealengine.com/latest/INT/Programming/Development/CodingStandard/index.html>

## Naming conventions

Descriptive names need to be used on variable names. The names need to be clear, concise and useful. Do not abbreviate too much.

|  |
| --- |
| // Bad: t = s + l + b;  // Good: TotalRecords = NewRecords + OldRecords – DeletedRecords; |

To make declarations of variables clear they should be declared one at a time.

Functions that have a return variable type of bool need to phrased as a question. Boolean variables should be prefixed with a “b”, such as “bSuccess”.

|  |
| --- |
| bool ValidRecord(VersionRec VersionToCheck) {...} // what is true  bool IsValidRecord(VersionRec VersionToCheck) {...} // Clear what we are checking |

Procedures should be named with a verb followed by an object. Do not use ambiguous verbs.

If a variable is passed by reference to a function, then the variable should be prefixed with “Out”. This indicates that the argument may change.

## General guidelines

Write self-documenting code:

|  |
| --- |
| // Bad: t = s + l + b;  // Good: TotalRecords = NewRecords + OldRecords – DeletedRecords; |

Write useful comments:

|  |
| --- |
| // Bad: // increment records ++TotalRecords;  // Good: // We know that there is another record ++ TotalRecords; |

Bad code should be re-written, not commented:

|  |
| --- |
| // Bad: // total number of records is sum of small and large records less the number of records that are both t = s + l + b;  // Good: TotalRecords = NewRecords + OldRecords – DeletedRecords; |

Do not contradict the code:

|  |
| --- |
| // Bad: // never increment records! ++Records;  // Good: // We know that there is another record ++Records; |

## Code Formatting

### Braces { }

Braces should begin on a new line. All control structures should use them. Even a one line statement should have braces.

|  |
| --- |
| **if**(bDoesRecordExists)  {  UseRecord();  }  **else**  {  CreateRecord();  } |

### Indentation

Tabs should be used for indenting code. Spaces can be used when aligning multi-line statements.

### Pointers

These should be in the format type\*<space>VarName. This is so that they can be quickly found in source files.

|  |
| --- |
| //Use this:  Record\* Rec  //Not These  Record \*Rec Record\* Rec |

## General Issues

* Initialize variables as close as possible to where they are used. This minimizes the distances in the code which makes it more readable.
* Functions should be broken down in to sub-functions where possible. This helps to make the code more readable and reusable.
* Functions declarations should not have a space between the function name and the first parenthesis.
* Compiler warnings from the code should be fixed where possible
* A blank line should be at the end of all .cpp and .h files
* Class members should be declared as private unless they need to be used as an interfacing function.
* Constant variable constraints should be set where possible to insure that values that are not expected to change remain as they are.
* Debug code should be descriptive and shown in appropriate context. “i=5” is not helpful, “For loop in CreateRecord Started: i = 5” is far more helpful when debugging.
* Reduce complex expressions into more logical easy to understand sub-expressions

## Standard Header

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Filename: databaseinterface.cpp

\* Group Name: TeamNixon

\* Subject: CSCI222

\* Assignment: File Archiver

\* File Description: This module handles all interactions with the database

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/