

1.

City Of Bristol College
C Programming Documentation
Standard

1. Purpose

This working document defines a coding practice for using the C programming language to :

- i) Ensure that C code is written to a consistent style and structure
- ii) Support the production of reliable, readable and maintainable code.

2. The Coding Practice

2.1 Lexical Conventions And Code Layout

2.1.1 Blank Lines

Blank lines should be inserted to segregate distinct sections of code.

2.1.2 Block Structure

- i) The block structure of the code will be emphasised by a suitable pattern of indentation.
- ii) The indentation increment will be 3 or 4 spaces.
- iii) Braces will be indented to the same level as the control statement with which they are associated

Example :

```
if ( ..... )
{
    .....
}
```

2.1.3 Capitalisation

The conventions for using upper and lower case letters are :

- i) Keywords (reserved words) in lower case.
- ii) Macro names (introduced by **#define**) in upper case.
- iii) Majority of lower case for identifiers.

2.1.4 Length Of Identifiers

The maximum length of an identifier should not exceed 31 characters.

2.1.5 Naming

- i) Reserved words will not be used as identifiers.
- ii) The names chosen for identifiers should be meaningful but not too verbose.
- iii) When coding mathematical formulae, the identifiers should reflect the mathematical notation embodied in any supporting documents or implied by standard practice.

- iv) Identifiers should not contain two consecutive underscore characters.
- v) Single letter identifiers are not allowed unless coding mathematical formulae or loop control variables but only lower case i, j or k.

2.1.6 Comments

- i) The code should be well commented to amplify the function of the code and to provide contextual information.
- ii) Each source file and function shall be headed by a comment.
- iii) Comments should not duplicate C syntax or semantics. In the following example the comment is redundant :

```
    j++;    /*Increment j*/
```

- iv) Comments should be written in good English with normal punctuation, using both upper and lower case characters.
- v) C++ style comments introduced by double slash (//) are not to be used.
- vi) There shall be a least one space between the comment delimiter and the text of the comment.

2.1.7 Multiple Statements Per Line

Only ONE statement shall appear on each source line. When a line contains more than one statement, readers may miss the second statement.

2.1.8 Literals

If the same literal appears several times in a code segment, this literal should be implemented as a named constant and declared using capitals.

2.1.9 Gotos

The use of GOTOs are not allowed in structured programming.

2.1.10 Operators

At least one space is required before and after each operator.

2.1.11 Function Declaration

When declaring a function prototype, the function name should start with a capital letter. If the name consists of two words , each can start with a capital. The function prototype should also declare variables used in its parameters to conform with ANSI C

Example :

```
void GetDetails ( char Name, int Age );
```

OR

```
void Get_Details ( char name, int age );
```

2.1.12 Source File Layout

The following template shall be used :

```
/*  
Header Comment  
*/  
  
/*  
Preprocessor Directives  
*/  
  
/*  
User and Enumerated Type Definitions  
*/  
  
/*  
Function Prototypes  
*/  
  
/*  
Global Variables  
*/  
  
/*  
Main Function  
*/  
  
/*  
User Functions  
*/
```