Coding Standards for C programming

This document contains some of the coding standards that the students have to follow , when they are writing the programs.

Indent style :

1. Indent the entire body of each function to one level of indentation.
2. Use three spaces or tab per level of indent.
3. Select the size of the indentation you prefer and uniformly apply that throughout the program.

```
// Printing on one line with two printf statements.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   printf( "Welcome " );
   printf( "to C!\n" );
} // end function main
```

← 1, 2, 3

4. If there are several levels of indentation, each level should be intended the same additional amount of space.
5. Indent the statement/s in the body of an if statement
6. If a statement is split across two or more lines, indent all subsequent lines.
7. Indent both body statements of an if … else statement

```
if ( grade >= 60 ) {
   puts( "Passed" );
} // end if
else {
   puts( "Failed" );
} // end else
```

← 4, 5, 7

8. Indent the statement/s in the body of an while/for statements

```
// processing phase
while ( counter <= 10 ) { // loop 10 times
   printf( "%s", "Enter grade: " ); // prompt for input
   scanf( "%d", &grade ); // read grade from user
   total = total + grade; // add grade to total
   counter = counter + 1; // increment counter
} // end while
```

Comments:

9. Describe the purpose of each function in the program using a comment
10. Add a comment to the right brace } of every function, to indicate the end of function.
11. Add comments to describe the purpose of the statements in the program

Example

```
// Addition program.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   int integer1; // first number to be entered by user
   int integer2; // second number to be entered by user
   int sum; // variable in which sum will be stored

   printf( "Enter first integer\n" ); // prompt
   scanf( "%d", &integer1 ); // read an integer

   printf( "Enter second integer\n" ); // prompt
   scanf( "%d", &integer2 ); // read an integer

   sum = integer1 + integer2; // assign total to sum

   printf( "Sum is %d\n", sum ); // print sum
} // end function main
```

9

11

10

Statements:

11. Separate the definitions and executable statements in a function with one blank line.

```
int integer1; // first number to be entered by user
int integer2; // second number to be entered by user
int sum; // variable in which sum will be stored

printf( "Enter first integer\n" ); // prompt
scanf( "%d", &integer1 ); // read an integer
```

12. Place a blank line before and after every if statement.

```
if ( num1 == num2 ) {
   printf( "%d is equal to %d\n", num1, num2 );
} // end if

if ( num1 != num2 ) {
   printf( "%d is not equal to %d\n", num1, num2 );
} // end if
```

White spaces:

13. Place a space after each comma(,) in printf and scanf statements.

```
printf( "Sum is %d\n", sum ); // print sum
scanf( "%d", &integer1 ); // read an integer
```

14. Place space on either side of a operator

```
sum = integer1 + integer2; // assign total to sum
```

15. Unary operators should be placed directly next to their operands with no intervening spaces.

```
passes++;
failures++;
student++;
```

Naming conventions:

16. Choose meaningful variable name.

```
int integer1; // first number to be entered by user
int integer2; // second number to be entered by user
int sum; // variable in which sum will be stored
```

17. When there are multiple words in a variable, begin each after the with a capital letter – camel case **(eg: totalCommissions)**

Programming practices:

18. Try to avoid using more than three levels of nesting in loops.