Our Current Coding Standard
Version: 0.1
Revised: 2018-11-23

```
/*  ----------------------------------------------------------------<Header>-
    Name:       fahr2cels.c
    Title:      Fahrenheit to Celsius temperature converter.

    Group:      TI-81
    Student:    Petrenko O.I.
    Written:    2018-11-22
    Revised:    2018-11-23

    Description: Write a program that requests the user to enter a Fahrenheit
                 temperature. The program should read the temperature as a type
                 double number and pass it as an argument to a user-supplied
                 function called Temperatures() . This function should calculate
                 the Celsius equivalent and the Kelvin equivalent and display all
                 three temperatures with a precision of two places to the right
                 of the decimal. It should identify each value with the
                 temperature scale it represents. Here is the formula for
                 converting Fahrenheit to Celsius:
                 Celsius = 5.0 / 9.0 * (Fahrenheit - 32.0)
 ----------------------------------------------------------------</Header>-*/

#include <stdio.h>
#include <stdlib.h>
void prn_temp(float cels, float fahr);
int main(void) {
    float celsius, fahrenheit;
    printf("\nEnter temperature in celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (1.8) * celsius + 32;
    prn_temp(celsius, fahrenheit);
    return 0;
}

/*  ----------------------------------------------------------------------[<]-
    Function: prn_temp
    Synopsis: Prints temperature in F and C.
    ----------------------------------------------------------------------[>]-*/
void prn_temp(float celsius, float fahrenheit) {
    printf("\n%f deg celsius is %f fahrenheit\n", celsius, fahrenheit);
}
```

C Coding Standard: Formatting

Brace Placement

Of the three major brace placement strategies one is recommended:

```
   if (condition) {      while (condition) {
      ...                   ...
   }                     }
```

```
do {
    v = x + y;
    w = y + 2;
} while (1);
```

When Braces are Needed

All if, while and do statements must either have braces or be on a single line.

Always Uses Braces Form

All if, while and do statements require braces even if there is only a single statement within the braces. For example:

```
if (1 == somevalue) {
   somevalue = 2;
}
```

Justification

It ensures that when someone adds a line of code later there are already braces and they don't forget. It provides a more consistent look. This doesn't affect execution speed. It's easy to do.

One Line Form

```
if (1 == somevalue) somevalue = 2;
```

Justification

It provides safety when adding new lines while maintainng a compact readable form.

If Then Else Formatting

Common approach is:

```
while (1) {
    if (condition) {
        somevalue = 2;
    } else if (condition) {
        w = y + 2;
    } else {
        v = x + y;
    }
}
```

If you have else if statements then it is usually a good idea to always have an else block for finding unhandled cases. Maybe put a log message in the else even if there is no corrective action taken.

switch Formatting

- Falling through a case statement into the next case statement shall be permitted as long as a comment is included.
- The default case should always be present and trigger an error if it should not be reached, yet is reached.
- If you need to create variables put all the code in a block.

Example

```
   switch (...) {
      case 1:
         ...
      /* comments */

      case 2: {
         int v;
         ...
```

```
      }
      break;

      default:
   }
```

Pointer Variables
place the * close to the variable name not pointer type
Example

```
  char *name= NULL;

  char *name, address;
```

Variable Names

use all lower case letters

use '_' as the word separator.

Example:

```
    message
    time_of_error
```

Global Variables

Global variables should be prepended with a 'g_'.
Global variables should be avoided whenever possible.

Example:

```
    int g_log;
```

Global Constants
Global constants should be all caps with '_' separators.

Justification
It's tradition for global constants to named this way. You must be careful to not conflict with other global #defines and enum labels.

Example:

```
    const int A_GLOBAL_CONSTANT= 5;
```