PROGRAMMING STANDARDS

CMPS 3500

Coding Style Guidelines

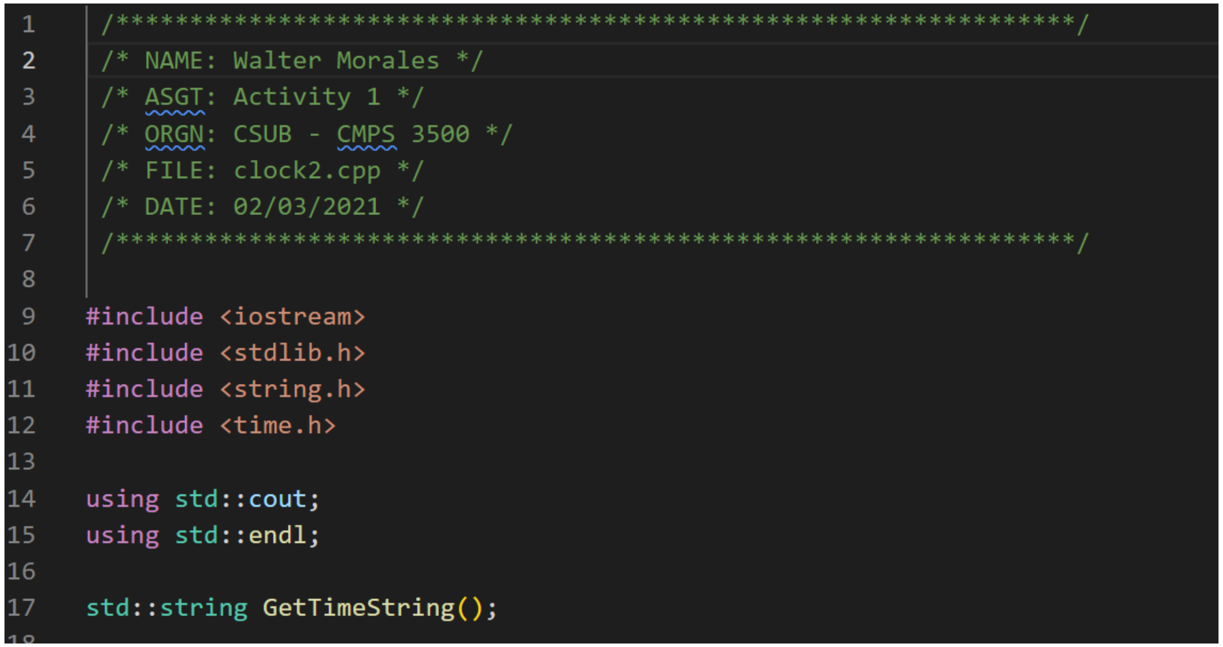
Coding Style Guidelines – CMPS 3500

If you are a student of CMPS 3500 you should follow the following coding style conventions and guidelines since your code will get evaluated using this criteria.

- Put your name on your code
- Class member variables use Snake Case Notation
- Methods and/or Functions Use Camel-Hump Notation
- Column Width
- Commenting Guidelines

Naming your file

- Put your name at the top of every source code file, include the assignment name, last update date, and file name.



```
/*************************************************************/
/* NAME: Walter Morales */
/* ASGT: Activity 1 */
/* ORGN: CSUB - CMPS 3500 */
/* FILE: clock2.cpp */
/* DATE: 02/03/2021 */
/*************************************************************/

#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <time.h>

using std::cout;
using std::endl;

std::string GetTimeString();
```

Sample header in a C++ code.

Variables Use Snake Case Notation

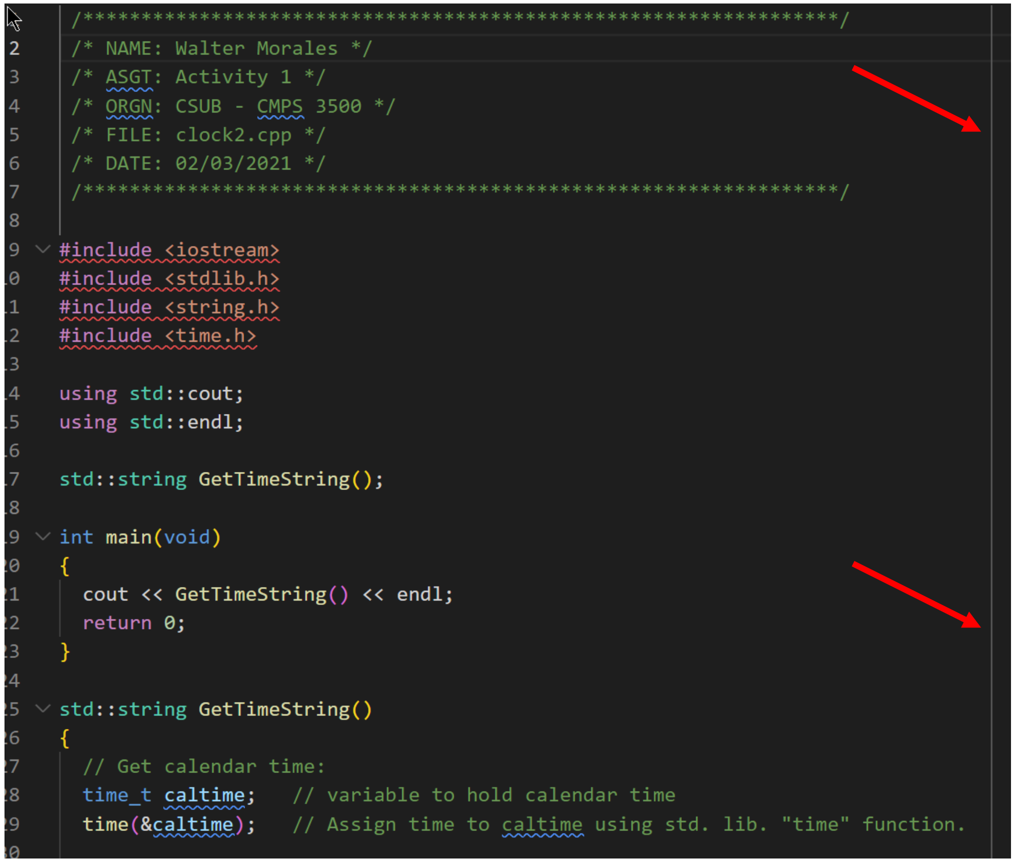
- Variables should begin with a lowercase letter. Even better if lower case variables are used exclusively. Variables with multiple word components should be connected with underscore character.
- Samples:
  - `a, i, j`
  - `one, result, var23`
  - `m_verbose, initial_temperature, best_so_far`

Methods and/or Functions Use Camel-Hump Notation

- **Camel-hump notation** uses a capital letter for each word component in the class name except for the first letter. By always beginning with a small letter, class functions are easily distinguishable from class names, which also use camel-hump notation.
- Samples:
  - `getFirstElement()`
  - `addIvPFunction()`
  - `solveIPP()`

Column Width

- A column of code should be no greater than 80-85 characters wide. Limiting the column width is motivated by wanting to see more than one column on the screen and avoiding wrap-around in a text editor.



```
/*****************************************************************/
/* NAME: Walter Morales */
/* ASGT: Activity 1 */
/* ORGN: CSUB - CMPS 3500 */
/* FILE: clock2.cpp */
/* DATE: 02/03/2021 */
/*****************************************************************/

#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <time.h>

using std::cout;
using std::endl;

std::string GetTimeString();

int main(void)
{
  cout << GetTimeString() << endl;
  return 0;
}

std::string GetTimeString()
{
  // Get calendar time:
  time_t caltime;    // variable to hold calendar time
  time(&caltime);    // Assign time to caltime using std. lib. "time" function.
```

Commenting Guidelines

- Commenting code is subjective but doing it in some way is almost universally accepted as a good practice. Each programming language has a different way of commenting in the source code.
- The basics tenets of commenting your code are simple:
  - Make them brief
  - Keep them relevant
  - Use them liberally, but not to excess