Code Consistency Conventions

Naming Conventions

Every class name (and struct name) should begin with a capital letter and follow the camel notation convention. Ex: User, CourseChat. If a data member (variable) is crucial to the correct functioning of the class (size member in a SearchArray whose whole purpose is to search for a value for example), this data member should always be private and have a public getter. Define a public setter only if it is needed. The data members and member functions of the class should follow the naming conventions for variables and functions (discussed next).

Functions should start with a lowercase character. It follows the camel notation. Try to make the function name descriptive (more than one word) so as to keep track of them across multiple files. Ex: findPerson, calculateDistance.

Variables should start with a lowercase character. It follows the underscore notation. Keep them descriptive unless they're local variables. Comments are important here (discussed later). Ex: nb_of_people, i.

File Management

Every class should have a separate header and code file. (if supported by the language). Use inline functions only when the code is a one-three line return/set/increment statement.

Every class should be placed in separate files. Only define multiple classes in one file if the class is very small (equivalent to a struct) but required multiple times in that class only.

Comments

Every function has to have a comment as the first line of it's body (with four backslashes instead of 2), that states the purpose of this function, and if it does something complex or changes something else externally that isn't trivial, it should be stated there. Ex:

//// Takes data and creates a new user then adds the user to the global user list.
//// Returns an int that describes the outcome of the addition
//// 0: success
//// 1: user name already exists

//// 2: invalid data
//// 3: no connection
//// -1: fatal error

Every variable that is going to be used locally or for a weird purpose should have a comment next to it, identifying it. No need to put comments on self explaining variables. Ex: nb_of_users doesn't need a comment, but user_crs_itt does.

Add comments before complex loops/if statements/etc. Use your judgment.

Misc

All functions should have a prototype defined at the top of the file. Any global variable should be defined at the top with _GL at the end (Ex: connection_status_GL). Any constant variable should be defined on top with Capital letters and underscore format (Ex: TIMEOUT_WAIT_INTERVAL).

Code reuse should be prioritized. If a certain function/snippet is going to be used in multiple files, it's better to define it in an external file, include that file, and call the function from there directly. Ex: Checking for an Internet connection (note that this is already a single function, but if it were not, we would have probably had to do what was described above).

Never hard-code (i,e, write them in the code directly rather than saving them in the function and using the function in the code) numbers that are approximations or have a good chance of being changed. Ex: Time before trying to connect to server again. Time to check for a message. Font size.