Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Mon, 02 Aug 2010 07:06:24 GMT
View Forum Message <> Reply to Message

hey mdelfede,

your point id pretty clear. i myself tend to 'overcomment' things as well. but the problem often is, that the commments are quite cryptic anyway. so even I myself, after several months have to really think about 'what did I mean with the comment' .

so the problem is not only to know 'where' to put some commentary hints, but also the 'what' to put there and how detailed. too little and you could have spared it, too much and it makes you think about just as much as if there were no comments.

so the clue on all that is to figure out 'the grain of information' which is needed later.

concerning upp code itself, i admit, i was (and still am) a beginner in a lot of programming topics, but i really learned MUCH from perceptable Upp coding style.though the code is mostely ot commented at all, it can be read qquite well. the quite offensive promotion of 'smart and agressive use of C++' is not underestimated and is a truth. but on the other hand ofcorse, i am very thankful for the forum, becasue there are fields where comments would just save some truble.

to comlete some coding style again:

often, one does not need to know every variable and its usage. the 'big picture' is often the best starting point for own development. these 'presumptions' taken by the programmer and author of the class / code are the missing stone, i.e. in Map Containers in upp, the approach is to simply have the desired container with an Index<> of the keys aligned to the same size and corresponding in indices. this makes further searches a lot easier because one can recognize more in the code. nevertheless, a prgrammer will not spare the work to dig the code a bit, if wanting to provide some new functionality or patch. but the step to go there can vary in extent, this is for sure, with the amount of quality information available on the subject don't underestimate simply visual reading of code.

another thing i personally really like about upp is the tendency to use really short but intuitive variables and even members. this makes the code itself less 'typographic' and enables focusing on the algorithm instead of literal text . especially things like '_myMember' or 'm_myOtherWeiredMember' make life unnesseccarily hard (with the help of Assist++ (STRG+Space) one esealy can find out if it's a member anyway).

good is also, that there is kind of a standard implicit positive action in many API functions, to be used as triggers or parameters or so, and they have negative counterparts that simply rely on the positive ones.

```
void Enable(bool e = true) {/*your code*/
void Disable() { return Enable(false); }
bool IsEnabled() const { /*your code*/ }
```

is really practical.

that was my 2 cents