Subject: is there U++ coding style reference?
Posted by kohait00 on Mon, 26 Jul 2010 21:01:39 GMT
View Forum Message <> Reply to Message

hi people,

often i read here and there 'your package should comply with U++ coding style'.

does any coding style references exist where one could lookup things? or do we need to bring together stuff?

cheers

Subject: Re: is there U++ coding style reference?
Posted by dolik.rce on Mon, 26 Jul 2010 22:16:01 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 26 July 2010 23:01hi people,

often i read here and there 'your package should comply with U++ coding style'.

does any coding style references exist where one could lookup things? or do we need to bring together stuff?

cheers
Hi kohait,

There is no explicit document about coding style, but you can take the U++ sources as reference. Or equivalently the code formating in theide (Edit > Advanced > Format code in editor). I guess that pretty much defines the recommended style. Plus the fact that the functions should be have an uppercase on the word boundaries (including the beginning) and that the code should be clean without unnecessary comments

That's about it I think, or did I forget something?

Best regards,
Honza

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 27 Jul 2010 06:19:38 GMT
View Forum Message <> Reply to Message

thanls dolik

what about indention, tabs / spaces?

brackets on new line?

class definition style, ctor/dtor position (often seen at bottom of cpp or h file, why?)

slight programming guide on when and how to properly design class (virtual dtor etc..) i know this depends on one's own skills as well, but a checklist would help for starters.

cheers

Subject: Re: is there U++ coding style reference?
Posted by dolik.rce on Tue, 27 Jul 2010 09:06:23 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 27 July 2010 08:19thanls dolik

what about indention, tabs / spaces?
brackets on new line?

class definition style, ctor/dtor position (often seen at bottom of cpp or h file, why?)

slight programming guide on when and how to properly design class (virtual dtor etc..) i know this depends on one's own skills as well, but a checklist would help for starters.

cheers
You really want a lot of details I usually don't even think about

Prefered indentation is using tabs (better recognized in theide).

Brackets are mostly on the same line. The only exception I can think of is in function definitions.

One important thing is putting spaces around operators etc. It increases the readability a lot.

I'm not sure if there is anything specific about classes, only the fact that U++ mostly uses single header per package and one cpp for each class (or group of closely related classes). I don't know about the ctor/dtor positions, but I strongly prefer having them on top of file, because they ussually contain important info (e.g. callback bindings to widgets).

The programming style is even harder to descrribe The main point would be probably that new and delete should be hidden in implementation, not exposed used in user interface and pointers are used just to point to things. The rest are ussual things, like effective but readable code, reusable classes etc.

Any other ideas? Anybody?

Honza

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 27 Jul 2010 09:24:22 GMT
View Forum Message <> Reply to Message

maybe the best thing would be to make a 'template' class project to show things...i'll try to set sth up..

Subject: Re: is there U++ coding style reference?
Posted by mrjt on Tue, 27 Jul 2010 10:02:51 GMT
View Forum Message <> Reply to Message

The only Upp style thing that caught me out was with references. Upp style (Mireks style really) is to put the ampersand with the type:
int& parameter
but I don't worry too much about this in my own code, only when committing packages to the bazaar. Otherwise Upp matched my previous style very well (one of the reasons I liked it in fact).

I too prefer constructors at the top of the file. I also attempt to keep functions in the order in which they appear in the header for easier browsing, but this often goes astray. For clarity I usually put static variable definitions, CH_STYLE and other housekeeping stuff at the bottom.

The new/delete stuff is very important IMO, since it's what separates Upp from almost all other C++ code. The rule is NEVER use new/delete. It's very unlikely you'd need something that the Upp containers can't do but in that case you should create your own simple containter to encapsulate the allocation and deallocation of memory. Then you never have to worry about memory leaks in application code (bar some special cases with statics).

The only caveat to this is that you want to use it for window management of non-dialog windows. For example:

```
class MyWindow : public WithLayout<TopWindow> {
  virtual void Close() { delete this; }
};

void NewWindow()
{
   new MyWindow()->Open();
}
```
though personally I would avoid this if at all possible.

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 27 Jul 2010 11:07:09 GMT
View Forum Message <> Reply to Message

this is quite a few infos, is good to concentrate them. thanks.
in fact, the coding style from u++ can already be felt somehow and i liked it as well and tried to

adopt it naturally ASAP.

especially the 'no new/delete' aspect and 'everything belong to somewhere' is great.

here comes some more stuff i tried to sum up what i learned and didnt want to forget. a class structure design. maybe this can help as well.

File Attachments

1) tempc.rar, downloaded 519 times

Subject: Re: is there U++ coding style reference?
Posted by dolik.rce on Tue, 27 Jul 2010 12:56:20 GMT
View Forum Message <> Reply to Message

Just a quick note noone mentioned so far: To keep the U++ style, the documentation should be in tpp, not in comments

Honza

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 27 Jul 2010 14:25:04 GMT
View Forum Message <> Reply to Message

your're right ofcorse, by the time i created this one i had no clue about tpp..

Subject: Re: is there U++ coding style reference?
Posted by koldo on Thu, 29 Jul 2010 08:10:05 GMT
View Forum Message <> Reply to Message

Some more advices

- Include package files using "Insert package directory file" mainly, instead of using "Insert any file".
- In the second case, tend to use relative to package directory file paths instead of absolute paths.
- If you use Bazaar package, include it to Assembly doing this: "File/Set Main Package", "Assembly" area, right button, "Edit assembly..." option, "Package nests:" field, adding the absolute path to Bazaar at the end.
- Include relative paths instead of absolute in #includes, like #include <Scatter/Scatter.h> instead of #include "/folderA/folderB/Scatter.h"

Subject: Re: is there U++ coding style reference?
Posted by dolik.rce on Thu, 29 Jul 2010 11:44:29 GMT
View Forum Message <> Reply to Message

Just remembered one more

U++ almost(?) never uses iterators, integral indices are preferred.

Subject: Re: is there U++ coding style reference?
Posted by mrjt on Thu, 29 Jul 2010 12:10:03 GMT
View Forum Message <> Reply to Message

Ah yes, that's another major plus point for me. I ****ing hate iterators.

Subject: Re: is there U++ coding style reference?
Posted by mdelfede on Fri, 30 Jul 2010 22:05:56 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 27 July 2010 00:16
.....
and that the code should be clean without unnecessary comments

The point there is to define what's "unnecessary" and how many years you'd like to remember what your code do without having to re-examine it for a week..... I'm still thinking that over-commenting is better than under-commenting.

Same discussion in as wine development. "No unnecessary comments", than just a bunch of people (4-5, I guess, among hundreds of wine contributors) can understand what the hell is happening inside gdi core code. Bah.

Max

Subject: Re: is there U++ coding style reference?
Posted by dolik.rce on Sat, 31 Jul 2010 20:07:03 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sat, 31 July 2010 00:05dolik.rce wrote on Tue, 27 July 2010 00:16
.....
and that the code should be clean without unnecessary comments

The point there is to define what's "unnecessary" and how many years you'd like to remember what your code do without having to re-examine it for a week..... I'm still thinking that over-commenting is better than under-commenting.

Same discussion in as wine development. "No unnecessary comments", than just a bunch of people (4-5, I guess, among hundreds of wine contributors) can understand what the hell is happening inside gdi core code. Bah.

Max

Hi Max!

Proper documentation does not have to be written as comments. Theide is actually very good at it. The topic++ system let's you describe each functions inputs, outputs and also how it works in "reference" section of documentation, which can be swiftly shown in code editor just by moving the mouse pointer to the squares on the left side gutter. Wider ideas and concepts should be in "implementation" section and end user manuals in "app" section.

I am well aware that writing documentation this way is not as simple and fast as just typing the comments into the code. But I prefer the little extra work if it keeps the code clean.

Also, well named functions and a good structured code can make wonders

That said, I agree that non-documented code that is not understandable to anyone is useless. I just wanted to point out that there is more than one way to do it

Best regards,
Honza

Subject: Re: is there U++ coding style reference?
Posted by mdelfede on Sat, 31 Jul 2010 21:00:51 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 31 July 2010 22:07
Hi Max!

Proper documentation does not have to be written as comments. Theide is actually very good at it. The topic++ system let's you describe each functions inputs, outputs and also how it works in "reference" section of documentation, which can be swiftly shown in code editor just by moving the mouse pointer to the squares on the left side gutter. Wider ideas and concepts should be in "implementation" section and end user manuals in "app" section.

I am well aware that writing documentation this way is not as simple and fast as just typing the comments into the code. But I prefer the little extra work if it keeps the code clean.

Also, well named functions and a good structured code can make wonders

That said, I agree that non-documented code that is not understandable to anyone is useless. I just wanted to point out that there is more than one way to do it

Best regards,
Honza

Hi Honza,

well, I didn't mean that comments should be like that one :

```
// increment i by one
i++;
```

But, when you read this :

```
#endif
  Vector<Scroll> scroll;
  VectorMap<Ctrl *, MoveCtrl> move;
  VectorMap<Ctrl *, MoveCtrl> scroll_move;
  Ptr<Ctrl>      owner;
 };

 Top         *top;
```

you surely agree that :
1) You can't have the minimal clue of understanding what the hell 'top' variable means, without reading 3/4 of CtrlCore code

2) You have no hint either of what are move and scroll_move members

and so on.
Nor you'll find them in TPP topics, because they're mostly private members non accessible from external code, so no use on document them in public TPP files.

BUT, if you have to add something to CtrlCore code, fix some bug or simply try to understand the code you're quickly in a trouble.
You can't either have a clue of what Top is looking at its definition :

```
 struct Top {
#ifdef PLATFORM_WIN32
  HWND          hwnd;
```

```
  UDropTarget   *dndtgt;
#endif
#ifdef PLATFORM_X11
  Window        window;
#endif
  Vector<Scroll> scroll;
  VectorMap<Ctrl *, MoveCtrl> move;
  VectorMap<Ctrl *, MoveCtrl> scroll_move;
  Ptr<Ctrl>      owner;
 };
```

So, when I say "better overcommenting than undercommenting" I mean exactly that.
A couple of years ago I had to put my hands in CtrlCore code to add X11DHctrl (and so X11 GlCtrl), ant this costed me a lot of work just to figure out what 'Top' was used for, among other stuffs. And, the bad stuff is that NOW (after a couple of years) I completely forgot all the implications of touching at it so, If should put my hands again on it I'd have to redo all the work from beginning.
So, I guess that (IIRC, of course....) putting something like that :

```
// top : pointer to underlying native window, if the control is native, NULL otherwise
Top top;
```

would have spared me some work at the expense of a short comment line. Worse than that, before the X11DHCtrl control (so, when I had to write it...) 'top' variable was unioned with another var in order to spare some memory; on non-native controls the top was NOT null but had another (IIRC completely unrelated) meaning.

If you look ad wine core graphics code, you'll find tons of these examples, and I defy you to manage to understand what's going on without loosing some months to go through all code and without tons of trials-and-errors patching it.....

Purpose of commenting code is *not* to document it from the user's point of view, but making it quickly readable from developer's point of view, even many years later.

Max

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

Subject: Re: is there U++ coding style reference?

Posted by mrjt on Mon, 02 Aug 2010 09:21:24 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 02 August 2010 08:06good 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
That's a very good point IMO. These functions are largely redundant but what they do is make code more grammatical by adding verbs in the correct places. Upp code is both easier to read and to write because of little things like this. Personally I think it's the easiest coding style to read that I've ever seen.

I've been working on a .Net project recently and in comparison I find the code very difficul to read. Partly this is inexperience with .Net but I think it's also down some poor naming conventions and language design choices.

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Mon, 02 Aug 2010 09:39:05 GMT
View Forum Message <> Reply to Message

regarding function names and variables:

Upp makes a lot usage of implicit semantic (already mentioned above) i,j,k are interger counter variables (even have abreviations for it), s is mostly a temporary string, b is bool etc..
this is not hungarian style (as to quote Linus: "hungarian is braindamaged"). in times of Assist++ and other helpers one can safely burry this idea.

this is implicit usage makes really slim synataxes possible. i.e. if(b) { //foo }, which really adds to readability.

the function names just names, they really define the action that is expected. good example are the containers, Add(), Remove(), Detach(), short, readable, general. general is also really important. this is a major advantage of upp code style. the functions exposed are not too specialised but rather imply and use typical knowledge of operation (containers again good example).
the style used for the function names is Pascal Case (according to wiki . starting with capital letter

and compunding each word again with its capital letter.

Subject: Re: is there U++ coding style reference?
Posted by koldo on Mon, 02 Aug 2010 12:59:25 GMT
View Forum Message <> Reply to Message

Hello all

Does anybody want to join this all and prepare a new chapter for U++ manual?

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Mon, 02 Aug 2010 13:34:32 GMT
View Forum Message <> Reply to Message

the esiest way woul be to provide a checklist with some examples. some checkpoints dont need any.

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 03 Aug 2010 09:24:38 GMT
View Forum Message <> Reply to Message

dolik wrote:
Quote:
Brackets are mostly on the same line

i disagree on that one . i have the impression that finding corresponding brackets is esier when they are placed on new line each. i know that Assist++ marks brackets, but it's also a visual perception of blocks, that matters, and here the new line stuff is better.

another thing:

small functions have their implementation in .h file anyway, and on same line as definition, like

Ctrl& Enabled(bool enable = true) { enabled = enable; return *this; }

rule of thumb: >3 syntactical important statements >> separate definition and implementation properly.

(syntactical important statements: real function calls and variable calculations, return values not counted, return *this is no 'important syntactical statement, only a helper.)

what about template classes?
.h / .hpp / .cpp relation and handling

did you look up my tempc class on that? any comments?

Subject: Re: is there U++ coding style reference?
Posted by mrjt on Tue, 03 Aug 2010 10:40:22 GMT
View Forum Message <> Reply to Message

Quote:i disagree on that one Smile. i have the impression that finding corresponding brackets is esier when they are placed on new line each. i know that Assist++ marks brackets, but it's also a visual perception of blocks, that matters, and here the new line stuff is better.
And I disagree with you I prefer more compact code so that I can scan it more easily (one of the reasons I dislike many verbose comments) and I don't find it makes finding pairs any harder. I feel it makes it easier to see the code srtucture personally.

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 03 Aug 2010 10:50:30 GMT
View Forum Message <> Reply to Message

compact code is a good thing, no doubt. but having braces on new line sets the commands apart from the invoker..kind of a structural helper. in any case, this might be one of the points where mirek could give an advice to have kind of a guideline..

here for comparison:

```
 if(a = 123)
 {
  int abc = 3345;
  for(int i = 0; i < 123; i++)
   abs += i;
 }

 if(a = 123) {
  int abc = 3345;
  for(int i = 0; i < 123; i++)
   abs += i;
 }
```

Subject: Re: is there U++ coding style reference?
Posted by mr_ped on Tue, 03 Aug 2010 11:05:53 GMT
View Forum Message <> Reply to Message

I prefer the second option, because I pair the "}" with "if" and that's more interesting for me than

pairing it with "{". But I can see this is matter of personal taste. (also I would write "if (a == 123) {" (notice the spaces) to fully satisfy my taste)

And assigning 123 to a in comparision without additional comment about intentionally doing so is not nice (I write in such cases code this way to make my intention obvious: "if ((a = 123) != 0) {", otherwise I consider it typo bug and fix it to "==")

Subject: Re: is there U++ coding style reference?
Posted by dolik.rce on Tue, 03 Aug 2010 11:39:48 GMT
View Forum Message <> Reply to Message

I totally agree with mrjt and his reasoning.Morover, I am using 5 levels of color coded backgrounds (with quite sharp colors), that gives much better idea about blocks than looking forbrackets.

As I said before, the only exception to have bracket on separate line is the first bracket in function definition. I don't like that too much as well, but there I can see the reason - if you set the function name apart from the rest of text, it makes it easier to find the function when "scanning" through long file.

But I would never do this with if, while, etc. Actually I often make the block on the same line, if it contains just a short code:
if(condition){int a = 213;}
for(int i = 0; i < 123; i++){abs += i;}But that is my personal style, definitely not official U++

Honza

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 03 Aug 2010 13:58:22 GMT
View Forum Message <> Reply to Message

Quote:
And assigning 123 to a in comparision without additional comment about intentionally doing so is not nice

that was a fast shot, i meant == ofcorse, where we are at the root of the problem .

some people even do stuff like

if(123 == a)
....
and

if(2 <= GetCount())

just to avoid this. i can see the good intention here but, honestly, if you found sth like that in code, you probably like i did, would have to twist your brain to grasp it.

this braces topic is, as we can see, really a question of taste and IMHO not too much affecting the overall code style. thats why i left it out in the code style section that i have uploaded some minutes ago.

please review, correct, add what ever is missing.
thanks

Subject: Re: is there U++ coding style reference?
Posted by mr_ped on Tue, 03 Aug 2010 15:28:01 GMT
View Forum Message <> Reply to Message

By reading book "Code complete" I learned to always write:
if (smaller <= bigger) ...
even if that leads to things like
if (10 <= i && i <= 13) ... //i is as expected inside 10..13 range

So 2 <= GetCount() looks reasonably to me.

Anyway, single "=" in if does produce warning anyway, until you use the more complex way like in my post (that's why I'm doing it, so I can detect mistakes from warnings).

But we are going off-topic here.

edit: (joke) also you don't need functional ">" on keyboard when you always use smaller <= greater..

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Tue, 03 Aug 2010 15:34:19 GMT
View Forum Message <> Reply to Message

this was a simple example though...there were more complex.
generally smaller < bigger is right. but in terms of semantical thinking it's sometimes not so good to follow it.

in upper case, semantical meaning was 'if count of a vector is at most 2, then...' where the focus or starting point is the count (and not the number 2), which should go first, because one thinks of count first..

offtopic, you're right

Subject: Re: is there U++ coding style reference?
Posted by mirek on Thu, 05 Aug 2010 10:50:55 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 03 August 2010 06:50compact code is a good thing, no doubt. but having braces on new line sets the commands apart from the invoker..kind of a structural helper. in any case, this might be one of the points where mirek could give an advice to have kind of a guideline..

here for comparison:

```
 if(a = 123)
 {
  int abc = 3345;
  for(int i = 0; i < 123; i++)
   abs += i;
 }

 if(a = 123) {
  int abc = 3345;
  for(int i = 0; i < 123; i++)
   abs += i;
 }
```

I certainly use more compact style.

Anyway, w.r.t. to formatting style guidelines, in the very very old past I was irritated when somebody submitted something with different formatting.

Later I have found that it in fact helps me to distinguish my code from other people just by seeing the formatting style

After this, I just do not care. Just produce a good code. If I feel too uncomfortable, I will reformat. By doing this I will also mark for me the code as "checked"

Subject: Re: is there U++ coding style reference?
Posted by kohait00 on Thu, 05 Aug 2010 10:53:49 GMT
View Forum Message <> Reply to Message

good idea