Subject: User lists of "bad" naming of classes, functions etc in U++...
Posted by fudadmin on Thu, 09 Mar 2006 02:50:01 GMT
View Forum Message <> Reply to Message

User lists of "bad" naming of classes, functions etc in U++...

If you are not happy with naming of classes, functions, variables etc. in U++, start your own reply and edit it constatly.
1 reply per user!!! No discussions!!! Discuss them separately (outside this)!! Only give a suggestion(s) and reason(s)

Subject: Arijus' list of "renames"
Posted by fudadmin on Thu, 09 Mar 2006 02:51:59 GMT
View Forum Message <> Reply to Message

1. Display class -> Displaytor (because it's a functor and not physical display)
2. EditField and LineEdit -> shouldn't be both Edit+Something or Something+Edit?
3. Shouldn't it be a pattern: all classes start with a "thing" and not "action"? like DateField, TextField etc...?
4. sometimes Get() and sometimes GetData() -> shouldn't it be (always) Get+Something more precisely or all classes have only one "main" Get or GetData?
5. SetFont() -> I'd expect to find it in all Text controls...
6. ShowTabs in LineEdit is confusing esp. when searching with TabCtrl -> should be ShowTabChars...

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by Coder on Sun, 31 Dec 2006 03:20:40 GMT
View Forum Message <> Reply to Message

From my experience working on large programming teams, having the variable scope prefixed to variables name helps make code a lot easier to read, and allows for different scopes to use the same name variable (less variable name conflicts). It helps in readability in that in looking at a piece of code, you don't have to search for the declaration of a variable in order to determine if it is a local, member, inherited member, static or global variable, and also in trying to understand how a member is used and doing a search you don't accidentally come across a local variable named the same as a member. The ones used in Hungarian Notation are s_* (s_Var) for static variables, g_* (g_Var) for global variables, m_* (m_Var) for class/struct member variables. Since member variables are most often used, sometimes I've seen code where they don't use an underscore, m* (mVar), or they only use underscore _* (_Var). I've also seen some use a_* for arguement variables, this is also useful in reducing variable name conflicts. Also, in general classes/structs with no or few functions and mostly public data, don't have to use scope prefixes.

I think type prefixes are less important, but can reduce variable name conflicts in some cases, or (if there is no quick help) it allows others to quickly understand what a variable is with out looking up it's declaration.

To Werner,
That is useful but not in a way that addresses much of the above. It still requires looking things up, and the information it returns has type and global, instance, class symbol. It doesn't appear to work for local variables. Plus it adds additional information that is kind of confusing about its locality.

There are similar things you can have in Visual Studio, one of them you hover the mouse and it tells you the type and what it is a member variable (if it is a member at all).

But really nothing beats being able to look at variables and know their scope, with out having to look it up. It may also be useful if there were some how a way to color code variables based upon their scope. But you still run into one of the other issues, the variable name conflicts.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by Werner on Mon, 01 Jan 2007 17:44:55 GMT
View Forum Message <> Reply to Message

You might want to have a look at the "Assist++" section of the Ultimate++ manual, especially the paragraph "Graphical symbols used by Assist++".

Werner

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by tvanriper on Tue, 18 Sep 2007 00:43:53 GMT
View Forum Message <> Reply to Message

I'm referring to 2007.1.. I haven't been working with the dev builds, so maybe this is already addressed.

GridCtrl and ArrayCtrl are similar controls in that they provide a spreadsheet-like view into rows and columns of data.

Consequently, I think there's an advantage in making sure their functions are named similarly, as someone may elect to switch between them (in fact, I switched from an ArrayCtrl to a GridCtrl recently in my work).

GridCtrl::IsSelected and ArrayCtrl::IsSelect are just close enough to being the same that it seems a shame not to make the names match.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by unodgs on Tue, 18 Sep 2007 06:44:40 GMT
View Forum Message <> Reply to Message

tvanriper wrote on Mon, 17 September 2007 20:43
GridCtrl and ArrayCtrl are similar controls in that they provide a spreadsheet-like view into rows and columns of data.
Consequently, I think there's an advantage in making sure their functions are named similarly, as someone may elect to switch between them (in fact, I switched from an ArrayCtrl to a GridCtrl recently in my work).
GridCtrl::IsSelected and ArrayCtrl::IsSelect are just close enough to being the same that it seems a shame not to make the names match.
IsSelect is not grammaticaly correct I was trying to follow ArrayCtrl interface, but there is too much differences now and there will be more in the future (new outstanding gridctrl is comming)
But in cases like IsSelect I agree that they could be the same. The only problem is : what to choose?

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by mirek on Tue, 18 Sep 2007 12:38:46 GMT
View Forum Message <> Reply to Message

Afaik, there is no IsSelect in ArrayCtrl...

There is IsSelection and IsSelected.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by mr_ped on Thu, 20 Sep 2007 13:16:39 GMT
View Forum Message <> Reply to Message

Util.h:

```
class Exc : public String {
public:
 Exc(); // throw according to GetLastError()
 Exc(const String& desc) : String(desc) {}

// void Show() const;
};

class AbortExc : public Exc {
public:
 AbortExc();
};
```

There's no word "Exception" anywhere, not even in comments.
Makes it hard to find trough text search if you are just checking around if the UPP has some predefined Exception class.
I don't ask as much for renaming the class, as for the comment added.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by mirek on Fri, 21 Sep 2007 12:56:17 GMT
View Forum Message <> Reply to Message

OK, now there is

Mirek

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by darkwyrm on Sun, 09 Mar 2008 13:19:56 GMT
View Forum Message <> Reply to Message

1. Option -> Checkbox
2. OptionTree -> CheckboxTree
3. Switch -> RadioButton

It's not a matter of opinion on this one, but of matching a cross-API de facto standard (like analog clocks .

gtk+: checkbutton / radiobutton
win32: checkbox / radiobutton
beos: BCheckBox / BRadioButton
wxwidgets: wxCheckBox / wxRadioButton
Qt: QCheckBox / QRadioButton

I would not have found them had I been actually looking for them.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by unodgs on Sun, 09 Mar 2008 17:22:58 GMT
View Forum Message <> Reply to Message

darkwyrm wrote on Sun, 09 March 2008 09:191. Option -> Checkbox
2. OptionTree -> CheckboxTree
3. Switch -> RadioButton

No no no I must say option means more to me than checkbox. The same with Switch. I know this is very subjective, but I vote for no changes here.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by ag_newb on Tue, 17 Nov 2009 09:00:45 GMT
View Forum Message <> Reply to Message

Boost C++ naming system should be adopted.
reason: delimiting by underscore is safer than capitalization, IMHO.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by andrei_natanael on Tue, 17 Nov 2009 14:41:17 GMT
View Forum Message <> Reply to Message

ag_newb wrote on Tue, 17 November 2009 11:00Boost C++ naming system should be adopted.
reason: delimiting by underscore is safer than capitalization, IMHO.
Have you an argument for this?

```
void DoSomething(const String& paramOne) {}

void do_something(const string& param_one) {}
```

IMO, using second version(with underscores) takes more time to write because when you write _ you have to press Shift+_ and that means 2 key press more than camel case naming.

Using camelCase notation you identify clearly which is a method of a class or a public variable. For example, you could notate all member function using notation LikeThis and variables using notation likeThis. If a variable is an instance of a class which redefine operator () is clear that accessing it like instance.variableName() means calling operator () from variableName and not calling function variableName().

```
class X
{
  class Y { public: void operator() {} };
public:
  Y variableName;
  void CallMe() {}
};

class x
{
  class y { public: void operator() {} };
public:
  y variable_name;
  void call_me() {}
};

int main()
{
```

```
  X camelCaseNotation;
  x underscore_notation;
  camelCaseNotation.variableName(); // means calling operator () from variableName
  camelCaseNotation.CallMe(); // means calling a member function

  underscore_notation.variable_name(); // looks like we are calling a member function
  underscore_notation.call_me(); // ok, we are calling a member function
  return 0;
}
```

IMO camelCase make you easier understanding the context.

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by koldo on Tue, 17 Nov 2009 15:48:19 GMT
View Forum Message <> Reply to Message

I support andrei_natanael.

Best regards
Koldo

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by mirek on Wed, 18 Nov 2009 07:41:21 GMT
View Forum Message <> Reply to Message

andrei_natanael wrote on Tue, 17 November 2009 09:41ag_newb wrote on Tue, 17 November 2009 11:00Boost C++ naming system should be adopted.
reason: delimiting by underscore is safer than capitalization, IMHO.
Have you an argument for this?

```
void DoSomething(const String& paramOne) {}

void do_something(const string& param_one) {}
```

IMO, using second version(with underscores) takes more time to write because when you write _ you have to press Shift+_ and that means 2 key press more than camel case naming.

I would not bother 2 more key presses.

But what I do not like about undescores is that it makes the line wider - in quite a lot more cases, it will break statements into more lines. That I believe reduces code readability.

However, I have removed 'sticky' flag of this post. I guess at this stage, any massive renaming is

impossible. And as it seems the topic is quite subjective, current naming might not be the best, but is definitely usable...

Mirek

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by andrei_natanael on Wed, 18 Nov 2009 11:05:58 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 18 November 2009 09:41
I would not bother 2 more key presses.

Then why we have << operator in TopWindow, THISBACK macro, etc.?
Regarding this, i like more to write win.Add(widget) than win << widget, and callback(this, &Window::Something) instead of THISBACK(Something) even if you type more because even if someone is a newcomer to U++ he figure out that win.Add(widget) will add the widget to win window.
Now that i'm talking about that i may summarize the things which i don't like at U++.
We say about U++ that it's a modern framework using advanced C++, though i don't see many design patterns used here, the platform abstraction is not so well implemented(IMO Chameleon it's not so good, dirty code inside) and there are a lot of macros (some have a good reason why them exists).

```
GUI_APP_MAIN // U++ initialization is hidden by this macro
{
  TopWindow w;
  w.Run();
}
```

It could be written:

```
int main(int argc, char *argv[]) // it is possible to write main() for GUI applications in Windows too
{
  Application app(argc, argv); U++ initialization is hidden in Application class
  TopWindow w;
  return app.Run(w);
}
```

Don't get me wrong, i like U++ that's why i'm using it but here are some bits that i don't agree with because they could be better.

Back to naming conventions if you're using STL with camelCase you may make the difference between you functions, algorithms, etc. and those provided by STL and say: Hey this Sort is provided by U++ not STL (because different naming conventions). However Mirek if you are supposed to re-create U++ which naming convention would you use?

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by mirek on Wed, 18 Nov 2009 11:48:10 GMT
View Forum Message <> Reply to Message

Quote:
GUI_APP_MAIN // U++ initialization is hidden by this macro

Quote:
int main(int argc, char *argv[]) // it is possible to write

Possible, but not really standard. And GUI_APP_MAIN does more than just Application app(argc, argv).

I am not really happy abou these macros too, but they really DO encapsulate a lot of platform specific stuff.

Quote:
Back to naming conventions if you're using STL with camelCase you may make the difference between you functions, algorithms, etc. and those provided by STL and say: Hey this Sort is provided by U++ not STL (because different naming conventions). However Mirek if you are supposed to re-create U++ which naming convention would you use?

Next time, I might consider camelCase, but I would most likely used InitCaps anyway....

Mirek

Subject: Re: User lists of "bad" naming of classes, functions etc in U++...
Posted by tojocky on Wed, 18 Nov 2009 12:56:34 GMT
View Forum Message <> Reply to Message

Interesting debate!

I thing that must to be a rules standard.

For me, I like macros!

With respect, Ion Lupascu (tojocky)