Subject: Difficulty with Class declaration
Posted by brokndodge on Wed, 31 Mar 2010 16:25:10 GMT
View Forum Message <> Reply to Message

I am experienced with GTK-Perl and of course Bash scripting, but those languages simply are not suited for the project I am working on now. I figured since it only took two days to learn perl - I could handle C++ with the help of UPP. Class is giving me some difficulty tho. I can run everything inline, but that would make for a very difficult to maintain source tree. I am trying to break the entire project up into classes for easier maintenance.

My difficulty is in trying to call a member function of a derived class from the base class. I have reviewed various example's and read every resource I can find on classes, references and pointers. I simply do not understand references and pointers.

My first test case is a simple call to a help window. I know there is probably a better way to display a help window, there is a method to my madness here in that I am trying to understand HOW to call a member of a derived class. Most of the functionality of my project will be in these derived classes with only the parent gui being in the base class. I am getting compile errors talking about non-class types and other such with every test I run. I'm not just looking for the answer, I need to understand the why and how of it.

Here is what I'm working with:
main.h

```
#ifndef _main_main_h_
#define _main_main_h_

#include <CtrlLib/CtrlLib.h>
#include <TabBar/TabBarCtrl.h>
// included Popups.h for complete definition of
// derived class Popups
#include <main/Popups.h>


using namespace Upp;

#define LAYOUTFILE <main/main.lay>
#include <CtrlCore/lay.h>

//forward declaration of derived class
class Popups;
// UDMS is my base class
struct UDMS : TopWindow {
 private:
  Popups lilpop;
  void ProspectDetailTab(void);

 public:
  typedef UDMS CLASSNAME;
```

```
 UDMS();

};

#endif //#define _main_main_h_
```

Popups.h

```
#ifndef _main_popups_h_
#define _main_popups_h_

#include <main/main.h>

using namespace Upp;

#define LAYOUTFILE <main/popups.lay>
#include <CtrlCore/lay.h>

struct UDMS;
class Popups : UDMS
{

private:

 WithQNHelpLayout<ParentCtrl> qnhelp;

 void close();

public:
 typedef Popups CLASSNAME;
 Popups();
 //core gui's

 //prospect gui's
 void QNHelpPopup(void);

};

#endif //#define _main_popups_h_
```

main.cpp

```
#include "main/main.h"
// ommitted 99% of app code to just
//include the offending function here.

void UDMS::ProspectDetailTab(void)
{
```

```
 struct lilpop;
 Popups lilpop;

 detail.QNHelpButton.WhenAction = THISBACK(lilpop.QNHelpPopup);

};
```

Popups.cpp

```
#include "main/Popups.h"


Popups::Popups()
{ ////Title("Unified Dealer Management System").Sizeable();
 //AddFrame(menu);


 //CtrlLayout(*this, "Unified Dealer Management System");
 CtrlLayout(qnhelp);

};

// this is the member of derived class Popups i'm trying to call
void Popups::QNHelpPopup(void)
{


 CtrlLayout(qnhelp);

 qnhelp.QNHelpText.SetText(" CCI - Customer Called IN \n"
 " RMC - Returned my Call \n"
 "  NA - No Answer \n"
 "DC1,2,3,w - Phone 1,2,3,work Disconnected \n"
 "LM1,2,3,w  - Left Message on phone 1,2,3,work \n"
 "APPT - Set Appointment (open appointment popup) \n"
 "  AC - Appointment Confirmed \n"
 "  NS - Didn't Show for Appointment \n"
 "KEPT - Kept Appointment \n"
 "  NN - Enter New Note \n"
 "Find - Open Find Prospect Popup");

 qnhelp.DoneButton.WhenAction = THISBACK(close);




};

void Popups::close(void) {
```

```
 delete this;
};
```

errors from mingw when I execute with ctrl+f5

```
popups.cpp
main.cpp
In file included from C:/MyApps/main/main.h:5,
                 from C:/MyApps/main/Popups.h:4,
                 from C:\MyApps\main\popups.cpp:1:
C:/upp/bazaar/TabBar/TabBarCtrl.h: In constructor `Upp::TabBarCtrl::Item::Item()':
C:/upp/bazaar/TabBar/TabBarCtrl.h:37: warning: converting of negative value `-0x000000001' to
`Upp::dword'
In file included from C:/MyApps/main/Popups.h:4,
                 from C:\MyApps\main\popups.cpp:1:
C:/MyApps/main/main.h: At global scope:
C:/MyApps/main/main.h:19: error: field `lilpop' has incomplete type
In file included from C:/MyApps/main/main.h:5,
                 from C:\MyApps\main\main.cpp:1:
C:/upp/bazaar/TabBar/TabBarCtrl.h: In constructor `Upp::TabBarCtrl::Item::Item()':
C:/upp/bazaar/TabBar/TabBarCtrl.h:37: warning: converting of negative value `-0x000000001' to
`Upp::dword'
In file included from C:/MyApps/main/main.h:6,
                 from C:\MyApps\main\main.cpp:1:
C:/MyApps/main/Popups.h: At global scope:
C:/MyApps/main/Popups.h:13: error: expected class-name before '{' token
C:\MyApps\main\main.cpp: In member function `void UDMS::ProspectDetailTab()':
C:\MyApps\main\main.cpp:74: error: ISO C++ forbids taking the address of a bound member
function to form a po
 inter to member function.  Say `&Popups::QNHelpPopup'
main: 2 file(s) built in (0:04.54), 2274 msecs / file, duration = 4688 msecs, parallelization 100%

There were errors. (0:05.03)
```

Just like I said previously, I don't just want a solution. I need to understand the why and how. I am learning as fast as I can.

File Attachments

1) main.zip, downloaded 323 times

Subject: Re: Difficulty with Class declaration
Posted by cbpporter on Wed, 31 Mar 2010 19:56:59 GMT
View Forum Message <> Reply to Message

Quote:
Just like I said previously, I don't just want a solution. I need to understand the why and how. I am learning as fast as I can.

OK, I won't give you the solution then. But I hope I can help you find it.

I don't know if you are familiar with C++ build process. Every C++ (cpp) source file is compiled independently from each other and one result does not affect future results. When all files are compiled, all the results (object files) are linked together.

So "main.cpp" failed to compile. Let's see why.

On first line to have "#include "main/main.h"". When using include "" it will try to find the file relative to your folder, so I think that "#include "main.h"" would be better, especially if you want to move your packages around on disk. If you later change "main" folder to something else, you won't have to edit your sources.

So we are including "main.h", which includes <CtrlLib/CtrlLib.h> and #include <TabBar/TabBarCtrl.h>. Then you include <main/Popups.h>.

So let's see what Popups.h does. Well first in includes "main.h". But you are already including from main.h the file "popup.h", which tries to include "main.h". This is a harmless circular reference because of the include guards, but it can cause problems if previous headers do not have the necessary declarations. So your include line with "main.h" won't do anything.

Let's skip to the important part:

struct UDMS;
class Popups : UDMS

"struct UDMS" basically tell the compiler "hey, there is a structure, and it is called UDMS. I don't know anything about it, except it has a name and is a structure. Fell free to complain if someone tries to use it.". Then you try to define the class Popus, which inherits from UDMS, which at this point is not defined.

UDMS is declared in "main.h", but after you try to inherit from it in "popups.h".

So in main.h you should declare UDMS, in popups.h you should include main.h and then declare the Popups class.

I think this is getting long and confusing and I'll leave the first post at this stage. You should try to go line by line and see if everything is visible and declared before you try to use it and I await further questions.

Subject: Re: Difficulty with Class declaration
Posted by koldo on Wed, 31 Mar 2010 20:13:20 GMT
View Forum Message <> Reply to Message

Hello brokndodge

Compiling the code you have submitted I obtain other errors. Please include in a compressed file all your test base.

However I have seen that struct UDMS has a class Popups member and also class Popus is a subclass of UDMS. That sounds strange.

Subject: Re: Difficulty with Class declaration
Posted by brokndodge on Wed, 31 Mar 2010 22:57:04 GMT
View Forum Message <> Reply to Message

Quote:UDMS is declared in "main.h", but after you try to inherit from it in "popups.h".

So in main.h you should declare UDMS, in popups.h you should include main.h and then declare the Popups class.

think i got the circular dependency thing fixed. Thinking about it, main.h needs to know about Popups.h but Popups.h has no need to know main.h even exists. So first I try it your way:

in popups.h
```
#include "main.h"
// struct UDMS;
struct Popups : UDMS { //no difference in results whether
                       //": UDMS" is here or not
//stuff
};
```

and in main.h
```
// #include "popups.h"
// class Popups;
struct UDMS : TopWindow {
  struct lilpop;
  Popups lilpop;

//stuff
};
```

i get the following errors
popups.cpp
main.cpp
In file included from C:\MyApps\main\main.h:5,

```
                 from C:\MyApps\main\Popups.h:12,
                 from C:\MyApps\main\popups.cpp:1:
C:/upp/bazaar/TabBar/TabBarCtrl.h: In constructor `Upp::TabBarCtrl::Item::Item()':
C:/upp/bazaar/TabBar/TabBarCtrl.h:37: warning: converting of negative value `-0x000000001' to
`Upp::dword'
In file included from C:\MyApps\main\Popups.h:12,
                 from C:\MyApps\main\popups.cpp:1:
C:\MyApps\main\main.h: At global scope:
C:\MyApps\main\main.h:21: error: `Popups' does not name a type
In file included from C:\MyApps\main\main.h:5,
                 from C:\MyApps\main\main.cpp:1:
C:/upp/bazaar/TabBar/TabBarCtrl.h: In constructor `Upp::TabBarCtrl::Item::Item()':
C:/upp/bazaar/TabBar/TabBarCtrl.h:37: warning: converting of negative value `-0x000000001' to
`Upp::dword'
In file included from C:\MyApps\main\main.cpp:1:
C:\MyApps\main\main.h: At global scope:
C:\MyApps\main\main.h:21: error: `Popups' does not name a type
C:\MyApps\main\main.cpp: In member function `void UDMS::ProspectDetailTab()':
C:\MyApps\main\main.cpp:74: error: `lilpop' is not a member of `UDMS'
main: 2 file(s) built in (0:04.20), 2102 msecs / file, duration = 4328 msecs, parallelization 100%

There were errors. (0:04.64)
```

Then I follow my own logic and reverse that to:
in popups.h

```
//Popups.h doesn't really need to know about main.h
//so commented out this #include
//#include "main.h"

class Popups // compiler fusses about expecting a class-name
             // before "{" if i use ": UDMS" in this example
{
// stuff
};
```

and in main.h

```
//main.h does need to know about Popups.h
#include "Popups.h"
struct UDMS : TopWindow {
  struct lilpop;
  Popups lilpop;

// stuff
};
```

i get the following

```
----- CtrlLib ( GUI GCC DEBUG DEBUG_FULL BLITZ WIN32 ) (1 / 10)
```

----- TabBar (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (2 / 10)
----- CtrlCore (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (3 / 10)
----- Draw (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (4 / 10)
----- plugin/bmp (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (5 / 10)
----- RichText (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (6 / 10)
----- Core (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (7 / 10)
----- plugin/z (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (8 / 10)
----- plugin/png (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (9 / 10)
----- main (GUI MAIN GCC DEBUG DEBUG_FULL BLITZ WIN32) (10 / 10)
popups.cpp
main.cpp
In file included from C:\MyApps\main\main.h:5,
                 from C:\MyApps\main\main.cpp:1:
C:/upp/bazaar/TabBar/TabBarCtrl.h: In constructor `Upp::TabBarCtrl::Item::Item()':
C:/upp/bazaar/TabBar/TabBarCtrl.h:37: warning: converting of negative value `-0x000000001' to `Upp::dword'
C:\MyApps\main\main.cpp: In member function `void UDMS::ProspectDetailTab()':
C:\MyApps\main\main.cpp:74: error: ISO C++ forbids taking the address of a bound member function to form a po
 inter to member function. Say `&Popups::QNHelpPopup'
main: 2 file(s) built in (0:05.19), 2596 msecs / file, duration = 5359 msecs, parallelization 100%

There were errors. (0:05.68)

now if i'm reading that correctly, it seems to be saying that i am simply calling Popups::QNHelpPopup incorrectly. The thing is I think it wants me to use a reference to Popups::QNHelpPopup rather than to call it directly, the whole reference thing is where I get completely lost.

Quote:However I have seen that struct UDMS has a class Popups member and also class Popups is a subclass of UDMS. That sounds strange.

I think that is where my confusion lies. I don't understand the difference or which one I should be doing. I don't fully understand most of the Perl code I write either, but it works and I know that in this instance to get this result I can do this. Seems that with c++ I do need to fully understand it. Class member or subclass seems like splitting hairs to me. So maybe that is where the problem lies.

ps. Attached unmodified main.zip to my first post. thats the complete project nest from before i started tinkering with it based on cbpporter's guidance. attached new main.zip to this post that includes new changes.

File Attachments

1) main_2.zip, downloaded 320 times

Subject: Re: Difficulty with Class declaration
Posted by Sender Ghost on Thu, 01 Apr 2010 01:58:12 GMT
View Forum Message <> Reply to Message

brokndodge wrote on Thu, 01 April 2010 00:57
I think that is where my confusion lies. I don't understand the difference or which one I should be doing. I don't fully understand most of the Perl code I write either, but it works and I know that in this instance to get this result I can do this. Seems that with c++ I do need to fully understand it. Class member or subclass seems like splitting hairs to me. So maybe that is where the problem lies.

Hello, Brokn.

To understand new things which you didn't understand before you need to construct new meaning.
First of you can use your previous experience to do analogues. For example, word of "class" is a programming thing that have concrete meaning, but this word is new for you. I suggest to use analogue word, such as "box". In the one big box you can place some small boxes. Next step is to expand box meaning from cube form to use different forms, e.g. flexible form.
The compiler can understand inner (private), outer (public) and protected inner (protected) places of box (class). The struct by default is public for compiler, but class is private - this is a difference between class and struct in C++.
Now, when you associate your previous meaning with current you can imagine how some box (class) can be used to construct another boxes (classes/structures). For example, you have following class declarations (just templates/forms from which compiler can construct new objects):

```
class Shape { };
class Square : Shape { };
class Circle : Shape { };
class CircleWithSquare : Circle, Square { };
```

In your imagination you can "draw" following images (or hologram):

Where ordered arrows forms dependence graph (note: some associative structure in another meaning) between classes. Now, you can understand paths to access dependence classes from one to another, e.g. CircleWithSquare object (instantiated class) can access Square and Circle class structures from which it constructed, as well as Shape.

Next step is improving of your thinking, his abilities to understand (new and/or previous) things. Programming as a whole is just a part of it, but not only.

Hope, I explained to you some meaning of your "how" and "why" questions.

File Attachments

1) Shapes.png, downloaded 944 times

Subject: Re: Difficulty with Class declaration
Posted by brokndodge on Thu, 01 Apr 2010 04:19:09 GMT
View Forum Message <> Reply to Message

So if I understand you correctly, UDMS can't access Popups, but Popups CAN access UDMS. Did I get that right?

So in order to get access to Popups, it needs to inherit from a class that UDMS inherits from. Possibly Upp? Or is my thinking twisted. Your analogy would indicate that BallWithSquare can access any class above it, but that Ball CANNOT access Square directly. So I have a menu function in main.cpp that needs access to Popups. I would actually have to define

```
class MenuOnMenuBar : Popups, UDMS {
    void menu ();
    Popups pop;
};
```

in order for menu to call any function in Popups. But then how does UDMS access MenuOnMenuBar? I'm missing something.

The normal hierarchy I am accustomed to is that main sets up the basic structure of my app. Then every other function comes under main. With this description of classes, my lowest level function would be in my top level class and everything would derive from it. That's completely backwards.

Maybe I shouldn't be using class at all. Can I have just one class UDMS and define everything in that one class? That means just one header file for the entire project.

My original blueprint calls for

```
class UDMS : TopWindow{}; //main class, creates topwindow, menubar and
                          //TabBar, calls Popups::Login then passes the rest
                          //of the work off to other modules.
                          // I had some difficulty with my initial attempt
                          // to call Sales from UDMS so I put together a
                          // simple popup in an effort to learn how all
                          // of this will work together.

class Popups : UDMS {}; // every class will share
                       // many common popups
class Sales : UDMS, Popups{};
class Inventory : Sales, Popups{};
class Finance : Sales, Inventory, Popups {};

class Service : UDMS, Popups{};
class Mechanics : Service, Popups{};
class Parts : Service, Popups{};
class Cashier : Mechanics, Service, Popups{};

class Reception : Sales, Service, Popups {};
```

```
class Accounting : Sales, Service, Popups {};
class Admin : Sales, Service, Reception, Accounting, Popups {};
```

With the reasoning described in your analogy I have the entire sketch backwards. I understand that since UDMS needs to call each of these modules, UDMS would have to derive from each of them. So defining each module in a class is just not the way to handle this project: or is there a way to achieve my goal thru pointers?

I'm going to go back to the tutorials and try to find a better way to do this.

Thanks each and every one of you for the information you gave me.

Subject: Re: Difficulty with Class declaration
Posted by cbpporter on Thu, 01 Apr 2010 06:39:59 GMT
View Forum Message <> Reply to Message

brokndodge wrote on Thu, 01 April 2010 01:57
in popups.h

```
#include "main.h"
// struct UDMS;
struct Popups : UDMS { //no difference in results whether
                       //": UDMS" is here or not
//stuff
};
```

and in main.h

```
// #include "popups.h"
// class Popups;
struct UDMS : TopWindow {
  struct lilpop;
  Popups lilpop;

//stuff
};
```

Well you are getting slightly closer. Main.h defines UDMS and Popups inherits form it, so this is ok. But main.h is wrong.

```
struct UDMS : TopWindow {
  struct lilpop;
  Popups lilpop;
```

First you have a "struct lilpop;". Struct used like this is used to forward declare a type, in you case the type will be "lolpop". Then you declare a variable of type Popups and also name as the type

the struct from a line ago. But the big problem is that here is you include order:

main.h: Defines UDMS, uses Popups
popups.h: Defines Popups, uses UDMS

This time you have a harmful/uncompilable circular reference. UDMS can not have a Popups member, though it can have a pointer to it.

Quote:
Seems that with c++ I do need to fully understand it. Class member or subclass seems like splitting hairs to me.

No, they are at least in principle profoundly different concepts.
Seeing this:
Quote:
```
class UDMS : TopWindow{}; //main class, creates topwindow, menubar and
                    //TabBar, calls Popups::Login then passes the rest
                    //of the work off to other modules.
                    // I had some difficulty with my initial attempt
                    // to call Sales from UDMS so I put together a
                    // simple popup in an effort to learn how all
                    // of this will work together.

class Popups : UDMS {}; // every class will share
                   // many common popups
class Sales : UDMS, Popups{};
```

I think you are expecting classes to behave differently than in C++. Maybe Perl. Maybe Perls has no traditional inheritance and you simulate it with members like one does in C. In "class UDMS : TopWindow{}" UDMS will inherit everything from TopWindow. You do not need to add a TopWindow member to UDMS.

Also "class Sales : UDMS, Popups{};" is wrong because you are inheriting from UDMS twice: once for UDMS and once for Popups, which already inherits from UDMS. You are using multiple inheritance, a quite tricky and rarely needed feature of C++.

Inheritance is a "is a kind of" relationship. Square is a kind of Shape. You do not want to have a Shape and call members from Square while in Shape. You want a Square.

Membership is a "has a" kind of relationship. Like said above, a Square is a kind of Shape. I does not have a Shape inside (as in adding it as a member; on the implementation side it does have one but it is added by the compiler). It has different stuff inside. It may have two points and a color. If you think that Shape is the common ground and it should have a color, then Shape will have a color, Box will be a kind of Shape and it will not directly have its own color, but it will inherit it from Shape, and it will have two points for the corners which Shape does not have.

Subject: Re: Difficulty with Class declaration
Posted by Sender Ghost on Thu, 01 Apr 2010 10:50:41 GMT
View Forum Message <> Reply to Message

Yes, I explained some part of the picture, of course. This is a starting point from/to where you can define your understanding. To do self-development you can use own methodologies or more general, such as practice by example. For example, you can create Doxygen diagrams of existing C++ code or use Source Navigator NG applictaion to see cross-references, e.g. LLVM API Documentation class hierarchy, libstdc++ Source Documentation class hierarchy.

Your blueprint forms following class diagram:

So, cbpporter is right, you using multiple inheritance.

The code have another layout. For THISBACK macros callbacks you need to define functions inside class/struct with CLASSNAME typedef from where it called.

As one of variants, you can implement all of Bar generation functions inside UDMS class then link ParentCtrls events with functions inside UDMS constructor.

File Attachments
1) UDMS.png, downloaded 987 times

Subject: Re: Difficulty with Class declaration
Posted by brokndodge on Sun, 04 Apr 2010 15:43:31 GMT
View Forum Message <> Reply to Message

IT COMPILES!!! I've spent a lot of time over the last several days reading every article Google yielded on c++ member functions and pointers. I still don't understand references and pointers, but I looked at me code last night and the solution just came to me. That's the way I learn I guess. I just need some time to absorb what I have read.

Anyway, I was reading thru the UDMS class, getting ready to just put everything into one huge file under one class, when I got to UDMS::ProspectDetailTab. It dawned on me that I wouldn't be able to point to Popups::member. I would need to point to a UDMS::member, then call Popups::member from there.

Came up with this solution:
main.h:
#ifndef main_main_h
#define main_main_h

#incluce <CtrlLib/CtrlLib.h>

using namespace upp;

#include "Popups.h" //has to come after "using namespace upp"

```
struct UDMS : TopWindow
{
private:
   // a bunch of declarations

   void CallToPopupsClass(); // new wrapper function
public:
   typedef UDMS CLASSNAME;
   UDMS();
   Popups pop;
};
#endif
```

main.cpp

```
#include "main.h"
// a lot of code that will eventually be moved
// to another function

void CallToPopupsClass(); // new wrapper function
{   pop.QNHelpPopup();
   return;
};

void ProspectDetailTab();
{   detail.QNHelpButton.WhenAction = THISBACK(CallToPopupsClass);
};
```

Thanks, cbpporter, for your insight. I've removed all references to UDMS from Popups. The circular dependency was preventing me from getting to the real problem: what I was trying to do just can't directly be done in c++. I'm working on figuring out a more dynamic way to call Popups class. Haven't gotten that far yet. Probably end up with a more dynamic inline function for my wrapper. I don't want 30 wrappers to Popups::members in main.cpp.

I think I understand what pointers and references are, but I'm having trouble with the when, where and why.

Next issue: try as I might, I'm having trouble with \n in a String. The newline sequence seems to be completely ignored. It's not showing up in the code, but it's also not resulting in a newline. I also tried crating my own popup window for this task with the same result.

```
String helpTxt= " CCI - Customer Called IN \n"
 " RMC - Returned my Call \n"
 "  NA - No Answer \n"
 "DC1,2,3,w - Phone 1,2,3,work Disconnected \n"
 "LM1,2,3,w  - Left Message on phone 1,2,3,work \n"
 "APPT - Set Appointment \(open appointment popup\) \n"
 "  AC - Appointment Confirmed \n"
```

```
"  NS - Didn't Show for Appointment \n"
"KEPT - Kept Appointment \n"
"  NN - Enter New Note \n"
"Find - Open Find Prospect Popup";

//qnhelp.DoneButton.WhenAction = THISBACK(CallToStub); */

PromptOK(helpTxt);
```

The above code results in:
Quote:
CCI - Customer Called IN RMC - Returned my Call NA - No Answer DC1,2,3,w - Phone 1,2,3,work Disconnected LM1,2,3,w - Left Message on phone 1,2,3,work APPT - Set Appointment (open appointment popup) AC - Appointment Confirmed NS - Didn't Show for Appointment KEPT - Kept Appointment NN - Enter New Note Find - Open Find Prospect Popup

Should look like:
Quote:
CCI - Customer Called IN
RMC - Returned my Call
NA - No Answer DC1,2,3,w -
Phone 1,2,3,work Disconnected
LM1,2,3,w - Left Message on phone 1,2,3,work
APPT - Set Appointment (open appointment popup)
AC - Appointment Confirmed
NS - Didn't Show for Appointment
KEPT - Kept Appointment
NN - Enter New Note
Find - Open Find Prospect Popup

Where di I go wrong?

Subject: Re: Difficulty with Class declaration
Posted by dolik.rce on Sun, 04 Apr 2010 16:01:05 GMT
View Forum Message <> Reply to Message

brokndodge wrote on Sun, 04 April 2010 17:43Next issue: try as I might, I'm having trouble with \n in a String. The newline sequence seems to be completely ignored. It's not showing up in the code, but it's also not resulting in a newline. I also tried crating my own popup window for this task with the same result.

```
String helpTxt= " CCI - Customer Called IN \n"
 " RMC - Returned my Call \n"
 "  NA - No Answer \n"
 "DC1,2,3,w - Phone 1,2,3,work Disconnected \n"
 "LM1,2,3,w  - Left Message on phone 1,2,3,work \n"
```

```
"APPT - Set Appointment \(open appointment popup\) \n"
"  AC - Appointment Confirmed \n"
"  NS - Didn't Show for Appointment \n"
"KEPT - Kept Appointment \n"
"  NN - Enter New Note \n"
"Find - Open Find Prospect Popup";

//qnhelp.DoneButton.WhenAction = THISBACK(CallToStub); */

PromptOK(helpTxt);
```

The above code results in:
Quote:
CCI - Customer Called IN RMC - Returned my Call NA - No Answer DC1,2,3,w - Phone 1,2,3,work Disconnected LM1,2,3,w - Left Message on phone 1,2,3,work APPT - Set Appointment (open appointment popup) AC - Appointment Confirmed NS - Didn't Show for Appointment KEPT - Kept Appointment NN - Enter New Note Find - Open Find Prospect Popup

Hi!
That is not a bug, that is a feature PromptOK expects its argument to be QTF formated. You can find more info on the formating in documentation. If you want to show just a plain text, you can use
PromptOK(DeQtf(helpTxt));
Best regards,
Honza

Subject: Re: Difficulty with Class declaration
Posted by brokndodge on Mon, 05 Apr 2010 01:29:35 GMT
View Forum Message <> Reply to Message

Quote:That is not a bug, that is a feature PromptOK expects its argument to be QTF formated. You can find more info on the formating in documentation.

Thanks for the link, Honza. Seems like an odd markup lanquage, but after studying the link you provided I got just enough information to do what I wanted to do.

Subject: Re: Difficulty with Class declaration
Posted by mr_ped on Tue, 06 Apr 2010 08:30:09 GMT
View Forum Message <> Reply to Message

about pointers and references...
If you are coming from "higher" language, it should be new for you, because many modern languages hide pointers internally and let you work with references only, so no wonder it does confuse you.

Don't worry, the concept itself is very simple, and U++ will actually support you in *not* using pointers extensively (which is often source of many memory leaks or damaged memory for inexperienced C++ programmers), so learning the concept and learning the U++ way of thinking (everything belongs somewhere) will get you back to productivity quite fast (faster then learning the "good old C").

It's about memory. Let's imagine a street with houses, each house having 1 room capable to store 32bits for example, they are on the street "memory" and each house has different number (going from 0 up).
Now let's have two variable declarations:
int IntegerValue;
int* IntegerPointer;

Both will reserve one house to hold it's value, let's say IntegerValue has house 100, IntegerPointer has house 101.

IntegerValue = 13;
This will tell compiler you want to store value 13 into house 100, but you don't need to remember it was house 100 and to do "memory[100] = 13;", you can use the "IntegerValue" name as an alias, and let compiler to figure it out for you.

IntegerPointer = &IntegerValue;
Notice the left side, there's no asterisk or ampersand, so it's pure value assignment, like in previous case. On the right side there's ampersand. This tells compiler you don't want the "13" value of IntegerValue, but you want it's true address (pointer), so the result will be 100; House 101 holds value 100 now.

int X = *IntegerPointer;
And the asterisk will dereference the IntegerPointer value (100) and use it as an address, so the X will be filled up with value from house 100, and that one holds 13.

int Y = IntegerPointer; would put 100 into Y (and compiler would warn you you are casting pointer to integer value, so it's probably not what you want).

int** pointer = &IntegerPointer; will put value 101 into "pointer", which is pointer to a pointer to an integer (thus int** data type)

"int*" is data type "pointer to int".

also keep in mind the [] (C arrays) works as dereferencing operator too, so:
int A = IntegerPointer[0];   //A == 13 (value from house 100+0)
int B = IntegerPointer[1];
//WRONG, but will work, we are going out of bounds of original IntegerPointer usage, as it was pointing to single value, not an array ... and in house 100+1 there's value 100, because there's the IntegerPointer itself stored, but that's just coincidence, there could have been anything.

int properarray[10]; //properarray is same type as int*, but the compiler will reserve 10 houses in

single range, like houses from 120 to 129. properarray then holds value 120, and using [] operator you can fetch up particular member of array.

uhm... this is probably the shortest possible introduction to C pointers, make sure you fully understand each statement, then we can move to new/delete operators, object instances and why U++ does help you to not bother with them too often (which makes U++ code looks quite like Java, because you don't have to bother with all this pointers stuff most of the time).

I completely omit C/C++ references (ampersands in type definitions and functions calls), because they are simple to explain once you understand pointers. Also I did omit function pointers and other syntax sugar, which is not needed to understand the concept, because in the end, the concept is simple:
- values are numbers
- pointers are addresses to numbers (and address itself is implemented as a simple number too, so for CPU there's no difference between number/pointer, it's just 13 and 100, but compiler+syntax does allow you to use them in different context and warn you when you use it plainly wrong)

update - also note after:
int* IntegerPointer;
we have reserved space in memory, but it's value is not initialized, so int x = *IntegerPointer; is mistake and you will either crash for referencing protected memory, or get bogus value.
This is one of the reasons why the references in C/C++ exist, basically they are identical to pointers, but they can't be uninitialized.
Also the good programming practice is to initialize pointers immediately, like int* IntegerPointer = NULL;
In case you then use it before setting it up to it's true value, you will get crash with access to adress 0, which is better then hunting random bogus numbers and occasional random crashes.

Subject: Re: Difficulty with Class declaration
Posted by brokndodge on Tue, 06 Apr 2010 14:20:51 GMT
View Forum Message <> Reply to Message

Quote:I completely omit C/C++ references (ampersands in type definitions and functions calls), because they are simple to explain once you understand pointers. Also I did omit function pointers and other syntax sugar, which is not needed to understand the concept, because in the end, the concept is simple:
- values are numbers
- pointers are addresses to numbers (and address itself is implemented as a simple number too, so for CPU there's no difference between number/pointer, it's just 13 and 100, but compiler+syntax does allow you to use them in different context and warn you when you use it plainly wrong)

Your short introduction is actually a little more clear than the various tutorials and howto's I've read in the last week. The "why" still alludes me tho. I currently have two class's: UDMS and Popups. UDMS::non_static_members needs to call Popups::non_static_members, but doesn't need to know anything about them or get anything from them. I want

Popups::non_static_members to hand some work off to MyAppSQL::non_static_members. Sometimes Popups::non_static_members might need to get some information back, but most of the time it will just pass information along.

I haven't implemented the new class yet because I still can't get UDMS to make a direct call to Popups. Right now I have a work around to call one Popups::non_static_member from UDMS, but it's ugly and will result in either a switch case or 30 additional functions in UDMS just to call various members of Popups. Then I have 8 more modules to write that will all need to be called from UDMS.

I tried to create just one class with each group of members in seperate .cpp files, but I didn't get the declaration right or something. I think my lack of understanding pointers is where the problem lies. I figured out the THISBACK macro is using a pointer "this". MingW doesn't want me to use "the address of a bound member function to form a pointer to a member function". The c++ shortcut << yields the same result, so it seems to mean the same thing as THISBACK.

So, it's the how and why of pointers that I'm missing. Or, maybe I shouldn't be using a pointer at all to call members of Popups. That is where my wrapper comes in. No pointer! void UDMS::CallToPopupsMember(){
Popups pop;
pop.Member();
};No pointer! But how do I do:void UDMS::ProspectDetail()
{detail.Button.WhenAction = THISBACK(CallToPopupsMember);
};without the wrapper function? THISBACK is a pointer. Right? If I just do Popups pop; detail.Button.WhenAction = THISBACK(pop.Member());MingW throws a fit. The next set of windows I'm working on are on hold until I get this figured out. I'm writing the schema.sch when I get tired of looking for the solution. Then I come back to this problem for a lil while.

Best I can come up with right now is a switch case. Hand a String[Array] off to UDMS::CallToPopups that includes the Popups::Member I want to call plus the arguments for that member. Then parse that String with a switch to figure out which Popups::Member needs called and pass the arguments to it. Seems ugly to me when I could just make a direct call.

I refuse to build my app with GTK-Perl. I don't want to hand code all those windows and I don't like Glade! So ugly will have to do for the first alpha release. I figure, if I can get the biggest part of the project coded, I will learn a lot along the way and can fix the ugliness before release candidate 1.

Thank you for your response Mr_Ped. Any further insight you could provide would be much appreciated. I know that I have tackled a pretty large project for my first c++ program. But, I believe it is the best language for this project and the project is the primary focus. I'll pick up the rest along the way.

PS. Ultimate++ is excellent and the community here is phenomenal!!! I understand more and more of the example code every day. I keep going back over them each day to gleam the next piece of insight.

Sorry, can't afford to study your problem in details, so I will response in dry theory and short introductions to general C/C++ concepts. (I hope other members will fill the gap with your project like they did up till now).

WHY pointers - excellent question and I'm feeling ashamed I didn't even think about explaining that.

Let's get back to stone age of programming to see why pointers did born.
There are generally two memory pools available for PC program, that's the STACK and HEAP. (and GPU memory nowadays, but it's the same stuff as highly specialized heap)

STACK is local to executed code, addressed by "sp" register in CPU. Take for example this piece of ugly code:

```
int ExplainStack1( int arg1 ) {
  int my_x = 13;
  my_x += arg1;
  return my_x;
}

//we start to execute here!
int result = ExplainStack1( 7 );
```

from this code snippet it's impossible to tell where result is stored, so ignore that one.
After execution starts, you want to call function ExplainStack1 with number 7 as parameter.
There's empty stack at the start, you PUSH there the number 7 (stack is LIFO queue, so you push it at top of stack), then you call the function (the return address for CPU is pushed on stack as well).
Inside function you reserve additional stack space for "my_x" variable (like pushing unknown value on top of stack).
So the stack looks like this:
top) my_x //<- here points the CPU "sp"
+1) return address
+2) 7
You fetch value of stack+2 (the 7), add it to top of stack (get resulting 20), put that into stack on position where result is expected by the original caller (let's say it's top+2 replacing the "7" for simplicity reasons).
You POP the local function mess from stack (the my_x and return address), and jump to return address...
Stack looks like this now:
top) 20 //the return address and my_x were above
The main code stores the 20 into "result".

As you can see, there's no absolute address used, everything is relative to top of stack.

This is nice, and some programming languages don't have heap and pointers stuff, just stack and they live well with it. C/C++ is more low level thing, so it's not enough.
Back in x86 days stack was usually limited to 64kB, so doing:

```
void foo() {
  int my_data[100000];
}
```

would not work, exceeding the limits of stack space. Even nowadays the stack is much smaller then heap.
But you have 640kB of RAM, right? (at least Bill told so) And you want to use it, you want just 100k of data. That's what heap is for, you reserve junk of 100k values out of it easily.

Stack is generally for returning addresses between functions calling and little amount of local variables to make things neat and tidy for CPU and execute it fast.

But that means you have to reserve/release big chunks of memory from OS's heap by some means, and that needs absolute addressing, because the OS can't be sure it will have free block of memory on the same relative spot to your code, like always.
That's where pointers come into play:

```
void foo() {
  int *mydata = new int[100000];   //reserve heap memory
  //on local stack only the pointer is allocated (32 or 64bits),
  //the 100k bunch of ints is reserver in heap.

  mydata[32768] = 10;    //work with the heap memory trough pointer

  delete [] mydata;  //release the heap memory by it's pointer
  //if you will not, it will be lost, because pointer to it is on local stack
  //which will be lost after function return => memory leaking
}
```

As you can see, pointers are not popular, because if you use them for allocating heap memory (and that's the most classic usage of them), and you lost the pointer, you can't release the memory later -> it is lost. (OS will recover it after your application does exit, but that's considered bad practice anyway)

... fast forward to modern times and U++ ...
the U++ way of avoiding pointers (and thus common mistakes of dereferencing uninitialized pointer, or not release resources) is to put as much as possible somewhere where it belongs.
I.e. if your widget needs int counter, it's member of it's class. That widget is used by main window, so whole widget is member of main window. Main window is allocated on local stack of main like: MainWindowClass window; (which allocated all the stuff inside in one big chunk on stack, like also that widget with it's counter).
So having int counters[100000]; as class member in widgets would hurt the stack, right? Do you need pointer then?
Depends what you are doing, if you are doing high performance stuff with static aligned memory pools, you may be better with pointer + manual management of resources.

If you have dynamic app which may need 10k, 30k or 100k of values from some external file, use NTL container instead.
That will make only the container management to occupy the stack space, and the container code will handle the dynamic heap memory allocation and releasing (trough properly written&called destructors of containers).
Like member of class is: Vector<int> counters;
Then in the member function you can do:
counters.Reserve(100000);
counters[32768] = 20;

As you see, no direct pointers and direct explicit memory management.

You just have to design your code and classes in a way that everything belongs somewhere and it's lifetime span is well defined by existence of parent object.
Like the "my_x" is released at the end of ExplainStack1 function implicitly, so C++ destructors and proper usage of classes will make this naturally happening for you.

Pointers are of course very handy whenever you need to address something in absolute way, no matter if you did allocate that resource by yourself, or not. (like accessing video ram of text mode at 0xb800:0000 from C by directly writing byte values there in DOS days)

... I hope this makes some sense, why pointers exists, and why they are source of many coding mistakes and more then a decade was spent to create libraries which help you to NOT use pointers (like STD, NTL and whole U++).

Now yet another Java vs C++ difference to answer your WHY:
void foo(int buffer[100000]) { ... function code ... }

In Java the function does get pointer to buffer which is automagically allocated in heap memory, so the stack is not burdened by push/pop of 100k data and killing the cache on CPU.
This is all hidden in Java, and as programmer you don't have to think about that, you put 100k buffer into function definition, and the java machine will put there just pointer (reference) to actual data.
In C++ it will do what you ask from it, i.e. kill the stack, cache and eventually OS too.
(well, the syntax I did use would actually lead to pointer even in C++ IMHO, but if it would be huge struct, it would do the ugly thing anyway, so I hope you get the idea)
So in C++ you can define the function as:
void foo(int *buffer) { ... function code ... }
And on the stack only pointer will be stored, which you can dereference later for actual values in buffer.

This naturally leads us to C++, classes and references:
1) void foo(WindowClass window) { ... }
2) void foo(WindowClass * window) { ... }
3) void foo(WindowClass & window) { ... }

1) upon calling with some w1 object will CLONE it on the stack space (if WindowClass is memory huge, you are asking for trouble), and give the function the clone (done by copy constructor) to

play with.
2) you have to call it with &w1 (address of w1 object), it will dereference it to have access to actual w1 (not it's copy!).
You can also call it with "NULL", which will make you crash upon dereferencing it.
3) (references - like Java magic) you do call it with foo(w1), but only pointer is passed down on stack. In function you don't have to dereference it as pointer, you work with it like with instance, i.e. window.bar(); What you do with it will affect the upper w1, it's not a copy of it. You can't pass NULL, only valid "WindowClass" instance, which is checked during compilation time.

I.e. references are nice to use when you want to affect actual value of passed argument, or you want to save the stack space and the passed object would be too huge. But you don't want pointers because you don't want to test for NULLs, etc.

Pointers are nice when you have to be fully dynamic, like maybe you get that object because user did want it, or maybe you get just NULL, because you should work without it.

Subject: Re: Difficulty with Class declaration
Posted by mr_ped on Tue, 06 Apr 2010 15:45:04 GMT
View Forum Message <> Reply to Message

Quote:
```
void UDMS::CallToPopupsMember() {
  Popups pop;
  pop.Member();
};
```

This will allocate local fresh new "pop" every time you call the function, then it will call "Member()" upon this fresh new instance, then it will get destroyed.

If you want persistent instance of Popups, it should be either member variable of UDMS definition, thus visible for every member function of UDMS, or you can use "static Popups pop;" which will make it visible only inside CallToPopupsMember() function, but it will be created only after first time call, later the already instantiated object will be used every next time.

If the "pop" should live outside of UDMS, you may consider to have only Popups *poppointer; as UDMS member variable, set to NULL by default, and set to something else by the owner of "pop" when the "pop" exists.
Then the function would work with pointer

```
void UDMS::CallToPopupsMember() {
  if ( poppointer == NULL ) return; //nobody gave me pop yet
  poppointer->Member();  //call Member() upon the given pop
};

void UDMS::SetPop( Popups *_newpop = NULL ) { poppointer = _newpop; }
```

And the owner of pop will call the UDMS::SetPop with either it's address or with NULL before it does destroy the pop.
(this is quite complicated design pattern with pointers usage, which can be most of the time avoided, but I'm putting it here to show you different approach)

Subject: Re: Difficulty with Class declaration
Posted by mr_ped on Tue, 06 Apr 2010 15:54:25 GMT
View Forum Message <> Reply to Message

And last thing... I'm definitely not explaining full C/C++ pointer syntax in these articles, like why new int[] leads to delete [], etc... you will have to dig into proper syntax in language reference. I'm just explaining the basic concept behind it, with that I'm sure you will understand all the syntax sugar from language reference soon, just keep on mind you will have to dig into it one more time.

Also I personally dislike stuff like function pointers (used for callbacks mostly), because I have to check language reference every time I have to use them (to know how to write it properly for compiler). This is another neat thing about U++, the macro "THISBACK" will save you in 99% of cases, so you don't have to bother with this rare syntax stuff.
But of course you should get trough it at least once to have idea what's going on in the background of THISBACK.
