Subject: Usage of DUMPC and other macros by using flags
Posted by 281264 on Fri, 15 Oct 2010 13:21:09 GMT
View Forum Message <> Reply to Message

Hi,

Is there an easy way to control (whether you want them to be executed depending upon you are debugging or not) the execution of, for instance, DUMPC() by using the flags defined in Main Package Configuration dialog?

I have done the following:

1.- I created a flag in the Main Package Configuration:

field Flags: .FLAGDEBUG (the point is for its application in this specific package, is it not?
Field Optional names: flagDEBUG.

2.- I created a .h file like this:
```
#ifdef flagDEBUG
 #define DEPUR(a) DUMPC(a)
#else
 #define DEPUR(a)
#endif
```

3.- I created an additional flag NORMAL (just to control the usage ot DUMPC)

4.- in the .cpp file:DEPUR(vector);

So, it works but it is not elegant at all; so is there a more direct (less baroque) way to do it?

Thank you,

Javier

Subject: Re: Usage of DUMPC and other macros by using flags
Posted by dolik.rce on Fri, 15 Oct 2010 14:35:51 GMT
View Forum Message <> Reply to Message

Hi Javier,

The DEBUG flag is set up automaticaly when compiling in Debug mode. You can test it with #ifdef flagDEBUG in your code.

The debugging macros DUMP,DUMPC,DUMPM,LOG etc. are already defined only in debug mode, you don't have to take care of that yourself.

I guess that is all you need Your solution was of course correct, but unnecessary. BTW the code

behind the LOG, DUMP,... is basicaly the same as yours

Additionally, if you use the dot (.FLAG), then it is used for the main package and for any package that has this FLAG in its Accept field in package manager.

Best regards,
Honza

Subject: Re: Usage of DUMPC and other macros by using flags
Posted by 281264 on Fri, 15 Oct 2010 15:24:46 GMT
View Forum Message <> Reply to Message

Honza,

Thank you. Definitely U++ design is well thought out!

Cheers,

Javier

Subject: Re: Usage of DUMPC and other macros by using flags
Posted by 281264 on Fri, 15 Oct 2010 15:50:36 GMT
View Forum Message <> Reply to Message

An additional question regarding ASSERT() macro; how can it be activated/deactivated? I have tried compiling both MSC Debug and Optimal and in both cases it is always activated.

Javier

Subject: Re: Usage of DUMPC and other macros by using flags
Posted by dolik.rce on Fri, 15 Oct 2010 17:12:22 GMT
View Forum Message <> Reply to Message

281264 wrote on Fri, 15 October 2010 17:50An additional question regarding ASSERT() macro; how can it be activated/deactivated? I have tried compiling both MSC Debug and Optimal and in both cases it is always activated.

Javier

The idea behind ASSERT is that it should never, ever trigger. The condition in it is ussualy protecting something that would lead into application crash or data coruption, so it is ussualy safer to let the application crash in semi-controllable manner before it demages something. But as I said

alreay: good production code should never trigger them anyway

If you want to use them in debug mode, then you'll have to define your own, using something like the code in the first post.

Honza

Subject: Re: Usage of DUMPC and other macros by using flags
Posted by 281264 on Fri, 15 Oct 2010 17:53:28 GMT
View Forum Message <> Reply to Message

Thanks. Perhaps it is more convenient using standard C/C++ assert and #define NDEBUG macro. What is the actual advantage of ASSERT compared with C/C++ assert?

Cheers,

Javier

Subject: Re: Usage of DUMPC and other macros by using flags
Posted by dolik.rce on Fri, 15 Oct 2010 19:52:03 GMT
View Forum Message <> Reply to Message

281264 wrote on Fri, 15 October 2010 19:53Thanks. Perhaps it is more convenient using standard C/C++ assert and #define NDEBUG macro. What is the actual advantage of ASSERT compared with C/C++ assert?

Cheers,

Javier

I am not really comfortable with standart C++, so I am not sure what assert does But I believe the advantages of U++ ASSERT might be 1) gui error report in GUI, 2) automatic break when run in debugger and 3) possibility to run custom cleanup function (set using "SetAssertFailedHook(void (*h)(const char *))").

Honza
