CoCoALib - Feature #1218

32-bit or 64-bit preprocessor macro?

08 Aug 2018 18:33 - John Abbott

Status:	Closed	**Start date:**	08 Aug 2018
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Tidying	**Estimated time:**	1.95 hour
Target version:	CoCoALib-0.99880	**Spent time:**	1.95 hour

Description

To avoid compiler warnings etc. I have used (3 times so far) a CPP trick to "hide" code from 32-bit platforms (which may complain of overly large integer literals).

Perhaps make a single CoCoA macro for distinguishing the two cases?

Discuss, decide, implement.

Related issues:

Related to CoCoALib - Design #1225: Move to C++14 (skipping C++11)	**In Progress**	**06 Sep 2018**

History

#1 - 08 Aug 2018 18:38 - John Abbott

Currently I use the following "trick" to hide code on 32-bit platforms:

```
/* The obvious test is the line below, but constants bigger than 32767 are   */
/* not portable, so instead we divide by 1000 twice then check the quotient. */
/*#if ULONG_MAX == 4294967295UL*/
#if (ULONG_MAX/1000)/1000 == 4294

/* Code for a 32-bit machine */

#else

/* Code for a 64-bit machine */

#endif
```

It might be more readable (and safer?) to have a single macro which says whether platform is 32-bit or 64-bit (or other???)

Possible macro name is **CoCoA_PLATFORM_BITSIZE** with values 32, 64 or ??? (maybe 0 to mean "unknown/unusual")

#2 - 05 Apr 2019 15:47 - John Abbott

- Related to Design #1225: Move to C++14 (skipping C++11) added

#3 - 05 Apr 2019 15:48 - John Abbott

We should check whether this still makes sense with C++14. I should also check on my little 32-bit machine...

#4 - 30 Apr 2019 17:20 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99700

#5 - 30 Oct 2019 21:56 - John Abbott

2019-10-30 I have just checked, and CoCoALib and CoCoA-5 both compile, run and all tests pass on my little 32-bit machine.

While I think we no longer need to release CoCoA-5 executables for 32-bit machines, it would be nice if the code could remain compilable on such machines (provided they have a C++14 compatible compiler).

Note that compilation and testing took quite a long time (*e.g.* just compiling and running all the examples took 31 mins).

If the changes needed for 32-bit compatibility are only minor, we can maintain them. If they become onerous, then we should probably abandon 32-bits.

#6 - 30 Oct 2019 22:10 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Exactly what wordsize assumptions do we want to make?

- **(A)** 32-bits or not (meaning more than 32?)
- **(B)** 32-bits or 64-bits
- **(C)** 32-bits or (at least 64)-bits

Option **(A)** seems to awkward to program for; though making the macro would be relatively easy.

Option **(B)** is probably easy to program for; but what happens if the platform is neither 32 nor 64? Compilation error?

Option **(C)** could also be awkward to program for, but not as awkward as **(A)**.

At this link **https://en.cppreference.com/w/cpp/language/types** there is a list of common data models. In my mind, I expect the model to be **LP64** (int is 32-bit, long is 64-bit, not sure about long long). C++ standard says long long must be at least 64-bits; it may let us avoid having to distinguish 32-bit and 64-bit platforms.

One could imagine some new, even bigger data model appearing in the future. If this happens then option **(B)** might mean that we would have to revise some parts of our code. I suspect this is unlikely, at least for many years.

#7 - 09 Jan 2020 12:24 - John Abbott

- Target version changed from CoCoALib-0.99700 to CoCoALib-0.99800

#8 - 03 Nov 2021 19:08 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

#9 - 08 Mar 2023 19:55 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880

#10 - 22 Apr 2024 20:21 - John Abbott

- Status changed from In Progress to Closed

- Assignee set to John Abbott

- % Done changed from 10 to 100

- Estimated time set to 1.95 h

This has effectively been resolved by issue #1661. The solution there assumes **either 64-bit or 32-bit long**; other data models are not catered for. Closing.