CoCoALib - Feature #828

MachineInt: function for checking that value is greater than some lower limit (and below MAXLONG)

30 Nov 2015 17:38 - John Abbott

Status:	In Progress	**Start date:**	30 Nov 2015
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	20%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	1.60 hour

Description

Many fns accepting a MachineInt actually want a positive (or non-negative) value.

There is a fn called IsInRange but the requires both upper and lower bounds to be specified.

It could also be handy to have a fn like IsInRange which returns the value (as a long) if it is in the specified range, and otherwise throws an "out of range" -- perhaps the name of the caller can be passed in so that the error message can give a better idea of where the problem arose?

History

#1 - 30 Nov 2015 17:42 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Perhaps the existing IsInRange is not too far from what I want; the real hitch is that an upperbound has to be specified (when in most cases I want it to be "MAXLONG").

One possibility could be to have a special enum with just one value MaxLong, then a call to IsInRange(1, n, MaxLong) would check that n is at least 1 and at most MaxLong.

Alternatively there could be a new fn, IsLongGreaterThan(n,1). I'm not sure how to achieve a meaningful name which will end up being more compact than IsInRange(1,n,MaxLong)

Any ideas? Comments?

NOTE (2015-12-09) a simpler solution would be to make MaxLong a constant global long whose value is numeric_limits<long>::max()

#2 - 30 Nov 2015 17:56 - Anna Maria Bigatti

John Abbott wrote:

> Perhaps the existing IsInRange is not too far from what I want; the real hitch is that an upperbound has to be specified (when in most cases I want it to be "MAXLONG").
>
> One possibility could be to have a special enum with just one value MaxLong, then a call to IsInRange(1, n, MaxLong) would check that n is at least 1 and at most MaxLong.
>
> Alternatively there could be a new fn, IsLongGreaterThan(n,1). I'm not sure how to achieve a meaningful name which will end up being more compact than IsInRange(1,n,MaxLong)
>
> Any ideas? Comments?

I like MaxLong! And MaxULong?

#3 - 30 Nov 2015 18:12 - John Abbott

In prima battuta farei solo MaxLong perche' voglio "scoraggiare" l'uso di unsigned long.

Non ti viene i mente un bel nome per una fn che controlla se il valore e` un long con valore almeno lwb?

Cosa pensi di una fn tipo RangeCheck(lwb, n, upb) che da` il valore di n se e` nel range indicato, altrimenti da` errore? Forse meglio RangeCheck(lwb,n,upb, NomeFn)?

#4 - 30 Nov 2015 18:50 - Anna Maria Bigatti

Ok, for the Check function (with the function name).
But I 'd rather have (n, lo, hi)... Am I too late to notice this?

#5 - 30 Nov 2015 18:54 - Anna Maria Bigatti

IsGreaterThan?
Or just overwrite operator>?

#6 - 01 Dec 2015 11:25 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 20

Mmm, I guess Scott Meyers would not be impressed by a fn IsInRange which takes 3 integer args -- one could guess that the args are "value", "lwb" and "upb", but what order should the args be in?

Perhaps a cleaner interface would be something like IsIn(val, interval(lwb,upb)) and we hope that the compiler can do a decent job.

Defining operator> and so on is possible, perhaps even a good idea.

Actually what I'm hoping for is a fn like RangeCheck because it is easy to use in the initialization part of a ctor. A call could look like RangeCheck(arg, interval(lwb,upb)); if we just want non-neg values then it would be RangeCheck(arg, interval(0,MaxLong)), and for strictly positive values we would write 1 instead of 0 to get RangeCheck(arg, interval(1,MaxLong)).

I'll think about it over lunch.

#7 - 01 Dec 2015 11:28 - John Abbott

It might make sense to allow other types of "condition":
RangeCheck(arg, PositiveLong) or perhaps simply RangeCheck(arg, positive)
RangeCheck(arg, NonNegLong) or perhaps simply RangeCheck(arg, NonNeg)

In some cases I want to check that the value is greater than 1, but then usually I want to give a specific error message (*e.g.* ERR::BadModulus).

#8 - 01 Dec 2015 15:26 - John Abbott

I've just spoken to Mario about this issue (even though he wasn't on the watchers list).

He made some suggestions:

- a class InRange whose operator() says whether a value is in the given range; a check would look like this InRange(0,100)(n)
- a CPP macro which calls RangeCheck adding an extra arg which is __FUNCTION__ so that the context of the error is automatically passed.

He seemed to think that the InRange class is perhaps the "most C++"-ish solution, but also liked the idea of RangeCheck (even though it seems more "pedestrian").

NOTE it seems that the "macro" FUNCTION is a gcc special feature, while the C++11 standard offers __func__ (but there seemed to be some doubt about how the "name" of a C++ fn is represented).

#9 - 01 Dec 2015 15:29 - John Abbott

I'm mostly interested in a solution which will make most code easy to read (without significant run-time overhead).

Maybe there could simply be some special fns such as CheckPositive and CheckNonNeg.

It is still not clear to me how to distinguish (names) between a fn which returns a boolean, and a fn which either throws or returns the value as a long.

#10 - 09 Dec 2015 13:54 - John Abbott

Currently a MachineInt is converted to a long using the function AsSignedLong which includes a check for overflow.

An idea is to allow AsSignedLong to take a second arg which is a predicate. If we have predicates such as InRange(a,b) and positive and perhaps GreaterThan(a) then a call would look like AsSignedLong(n, InRange(2,32767)). This is a bit verbose, but seems fairly readable to me. Notice that AsSignedLong(n, GreaterThan(1)) automatically implies that the max value is MaxLong.