An Overview of Contracts Papers for Cologne

Document #:	P1807R0
Date:	2019-07-23
Project:	Programming Language C++
Audience:	EWG

Reply-to: Joshua Berne <jberne4@bloomberg.net>

Abstract

There are many contract papers of interest to those wishing to participate in the discussion of what contracts should be in C++ 20. We hope to summarize (from as objective a standpoint as possible, though clearly biased in some ways as we are the authors of a number of these papers) the papers that are going to be presented, as well as any papers available that might provide informative supporting information for a decision.

Contents

1 Cologne Scheduled Contract Discussions

1.1 Monday Part 1 – Design Fixes

The first batch of Monday papers require the most comparative analysis, as they all propose a different final set of features (or multiple different sets of features) coming into Cologne. They also include a fair bit of overlap and some cases where papers explicitly remove functionality that other papers build on.

1.1.1 Proposals – P1711R0, P1429R2, P1290R3, P1730R0, P1782R0, P1607R0

First, we will summarize the options presented in the papers that will be discussed:

- N4820 – *Working Draft, Standard for Programming Language C++* – The definition of the current status quo, i.e., what we get if no other proposal reaches consensus.
  - N4820 – The status quo.
- P1711R0 – *What to do about contracts?* – Proposes 4 possible courses of action, with a preference for the first or fourth:
  - P1711R0:1 – The status quo, the same as N4820
  - P1711R0:2 – Remove contracts entirely.
  - P1711R0:3 – (Minimal change) Add *assumption mode*.
  - P1711R0:4 – (Minimal feature) Add *assumption mode*, remove *continuation mode*.
- D1429R3 – *Contracts That Work* – Proposes literal semantics, and a number of variations on which build time configurations are available:
  - D1429R3:DL – Add literal semantics, add arbitrary assignment of levels to semantics
  - D1429R3:DLG – Add literal semantics, add arbitrary assignment of levels to semantics, add *global contract mode*
  - D1429R3:B – Add literal semantics, keep *build level* and *continuation mode*
  - D1429R3:BG – Add literal semantics, keep *build level* and *continuation mode*, add *global contract mode*
  - D1429R3:A – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*
  - D1429R3:AG – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*, add *global contract mode*
  - D1429R3:BL – Add literal semantics, keep *build level* and *continuation mode*, add arbitrary assignment of levels to semantics

  - D1429R3:BLG – Add literal semantics, keep *build level* and *continuation mode*, add arbitrary assignment of levels to semantics, add *global contract mode*
  - D1429R3:AL – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*, add arbitrary assignment of levels to semantics
  - D1429R3:ALG – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*, add arbitrary assignment of levels to semantics, add *global contract mode*
- P1290R3 – *Avoiding undefined behavior in contracts* – Removes any ability to have *default* or *audit* level contracts assumed, adds a new flag to control the assumption of *axiom* level contracts:
  - P1290R3 – Remove assumption of default/audit, add axiom mode.
- P1730R0 – *Adding a global contract assumption mode* – Adds an assumption mode, with the same functionality as that proposed by P1711R0:3.
- P1782R0 – *Local contract restrictions* – Defines a new set of restrictions that can be assigned to each contract:
  - P1782R0 – 32 combinations of "tentative, halt, static, audit, always".
- P1607R0 – *Minimizing Contracts* – Proposes removing levels entirely, and then potentially adding in literal semantics.
  - P1607R0:1 – Removal of all but default contracts, which can only be 'check_never_continue' or 'ignore'.
  - P1607R0:2 – Removal of all but default contracts, which can only be 'check_never_continue' or 'ignore', add (4) in-code literal semantics.

1.1.2 Comparison Charts

These papers include a fair number of common features, and some fairly unique proposals as well. The upcoming table will include columns with information for each of these papers.

Two columns indicate some metadata about the proposals:

- P – This is one of the preferred variations of the paper author.
- DUP – This proposal is functionally a duplicate an earlier presented proposal - note that this does not include proposals that contain another proposal's changes in addition to other features (see the chart on the following page for those relationships).

Then a column to indicate whether the paper proposes a feature should be included:

- F1 – Contracts as a language feature.
- F2 – Contract definitions should be stated in terms of the semantics of P1429.
- F3 – Unchecked `default` and `audit` contracts are assumable.
- F4 – Behavior defined in terms of hode code and configuration map to clearly named semantics.

The following variations on how contracts can be controlled by a *Build Mode* (i.e., whatever values might be available to translate a program in different ways, generally from the command line) have been proposed:

- B# – The total number of semantically different build modes that are available.
- B1 – A *build level* of off, default, or audit.
- B2 – A *continuation mode* that controls whether checked contracts continue after failed checks or not.
- B3 – A *assumption mode* that controls whether unchecked contracts are assumed.
- B4 – A *axiom mode* that controls whether axiom-level contracts are assumed.
- B5 – The ability to arbitrarily assign default, audit, and axiom to contract semantics.
- B6 – A *global contract mode* that enables/disables all contracts, including those with literal semantics.

The following variations on what can be put into contract specifiers are also proposed:

- C# – Number of distinct annotations that can be marked on a contract (the grammar term defined as *contract mode* in D1429R3.) Note that in most proposals, no extra tokens and the token `default` are equivalent – those have not been double counted.
- C1 – Contracts can have a level of *default*, *audit*, or *axiom*.
- C2 – Contracts can have a semantic of *ignore*, *assume*, *check_maybe_continue*, or *check_never_continue*.
- C3 – Contracts can have a modifier of *tentative*, *halt*, *static*, *audit*, or *always*.

The complete chart of which proposals among those being presented Monday morning have which features follows – note that other papers on Monday and Wednesday are all relatively orthogonal to the choice being made among these options:

Proposal	P	Dup	F1	F2	F3	F4	#	B1	B2	B3	B4	B5	B6	C#	C1	C2	C3
	Preferred		Contracts	Semantics	Unchecked are Assumed	Axioms are Assumed		Build Level	Continuation Mode	Assumption Mode	Axiom Mode	Assignment	Global Mode		Levels	Literal Semantics	Restrictions
N4820			✓	-	✓	✓	5	✓	✓	-	-	-	-	3	✓	-	-
P1711R0:1	✓	N4810	✓	-	✓	✓	5	✓	✓	-	-	-	-	3	✓	-	-
P1711R0:2			-	-				-	-	-	-	-	-		-	-	-
P1711R0:3			✓	-	✓/X	✓/X	10	✓	✓	✓	-	-	-	3	✓	-	-
P1711R0:4	✓		✓	-	✓/X	✓/X	6	✓	X	✓	-	-	-	3	✓	-	-
D1429R3:DL	✓		✓	✓	✓/X	✓/X	32	X	X	-	-	✓	-	7	✓	✓	-
D1429R3:DLG	✓		✓	✓	✓/X	✓/X	33	X	X	-	-	✓	✓	7	✓	✓	-
D1429R3:B			✓	✓	✓/X	✓/X	5	✓	✓	-	-	-	-	7	✓	✓	-
D1429R3:BG			✓	✓	✓/X	✓/X	6	✓	✓	-	-	-	✓	7	✓	✓	-
D1429R3:A			✓	✓	✓/X	✓/X	10	✓	✓	✓	-	-	-	7	✓	✓	-
D1429R3:AG			✓	✓	✓/X	✓/X	11	✓	✓	✓	-	-	✓	7	✓	✓	-
D1429R3:BL			✓	✓	✓/X	✓/X	32	✓	✓	-	-	✓	-	7	✓	✓	-
D1429R3:BLG			✓	✓	✓/X	✓/X	33	✓	✓	-	-	✓	✓	7	✓	✓	-
D1429R3:AL	✓		✓	✓	✓/X	✓/X	32	✓	✓	✓	-	✓	-	7	✓	✓	-
D1429R3:ALG	✓		✓	✓	✓/X	✓/X	33	✓	✓	✓	-	✓	✓	7	✓	✓	-
P1290R3	✓		✓	-	X	✓/X	10	✓	✓	-	✓	-	-	3	✓	-	-
P1730R0	✓	P1711R0:3	✓	-	✓/X	✓/X	10	✓	✓	✓	-	-	-	3	✓	-	-
P1782R0	✓		✓	~✓	✓/X	✓/X	12	✓	✓	✓	-	-	-	12	~✓	~✓	✓
P1607R0:1			✓	✓	X	-	2	-	-	-	-	-	✓	1	-	-	-
P1607R0:2	✓		✓	✓	✓/X	-	2	-	-	-	-	✓	-	5	-	✓	-

- ✓– Yes
- ~✓– A close approximation of Yes
- ✓/X– Configurable yes or no
- X– No, explicitly not allowed or removed
- - – Feature irrelevant in context of proposal

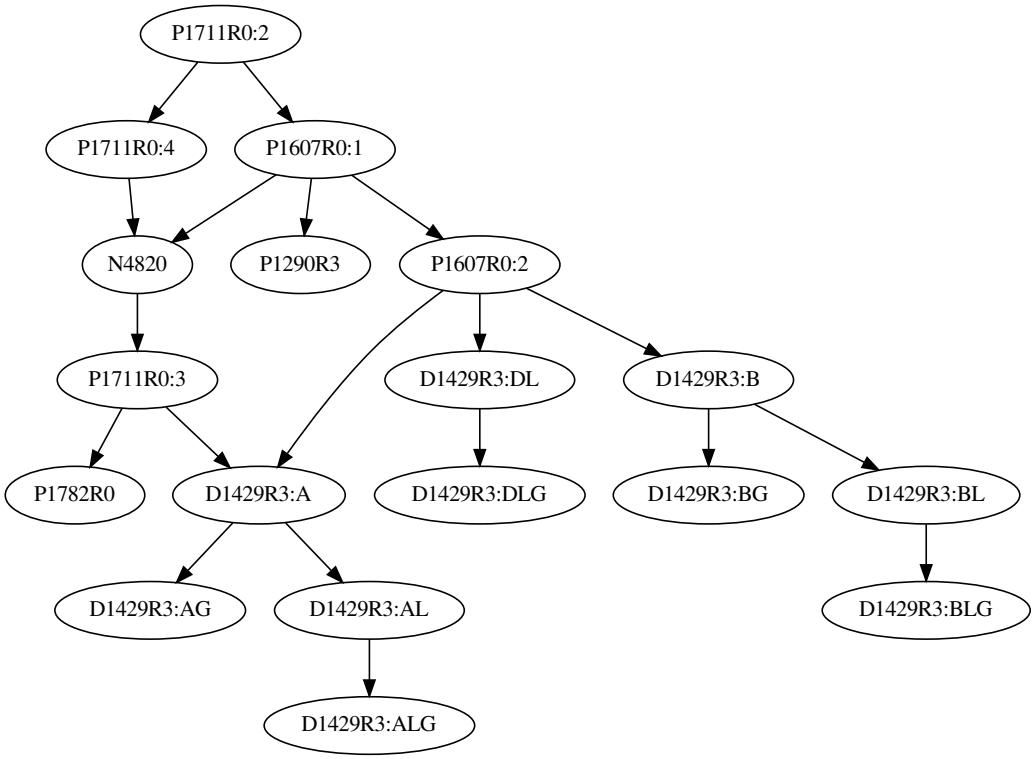
Notes on P1782R0:

- Most of functionality outlined in P1332R0 is available
- Levels are encoded on a contract differently than in other proposals
- The literal semantics can be roughly aproximated - `assume ~= static`, `ignore == tentative static`, `always ~= check_maybe_continue`, `always halt == check_never_continue`.

As a possible source of understanding, below is a graph of the design fix proposals related to one another based on available functionality - both at coding time and build time.

Code written in one of these proposals with certain build configurations will be valid in another proposal with the same available build configurations if there is a path going down from the first to the second proposal.

Note that a separate chart showing the relationships based on available program semantics might be very different.



1.2 Monday Part 2 – Other Changes

The remaining papers for discussion on Monday are all relatively orthogonal to the first set of proposals, excluding the P1711:2 (remove contracts) and possible some different interactions with P1782R0.

1.2.1 P1793R0 – Simplifying Contract Syntax

This paper proposes renaming the first contract level from *default* to *assert*, along with some minor syntax trimming for in-function contracts with that change. Details of how it might work with P1782 would need to be ironed out, but other than that this change would be compatible with any of the earlier proposals.

1.2.2 P1769R0 – The "default" contract build-level and continuation-mode should be implementation-defined

This proposal asks to make the default *build mode* implementation defined, noting that it effectively is anyway, regardless of what we might attempt to standardize. Other than the choice to remove contracts, this change would be compatible with all earlier proposals, though the specifics of the wording change might be different.

1.2.3 P1320R2 – Allowing contract predicates on non-first declarations

This paper proposes allowing contract annotations to appear on any declaration, as well as allowing redeclaration of member functions to provide an additional place outside the class declaration to put contract specifiers. This would, again, be compatible with any of the earlier proposals that retain contracts as a language feature.

1.3 Wednesday

Wednesday's discussion focuses on bug fixes. Note that, depending on the outcome of Monday's decisions, the papers presented here might vary widely in content and relevance. This is likely also when clarifications from CWG will come back based on what decisions were made on Monday.

1.3.1 P1704R0 – Undefined functions in axiom-level contract statements

This paper focuses on the behavior of enabled axiom-level contracts.

1.3.2 P1670R0 – Side Effects of Checked Contracts and Predicate Elision

This paper focuses on the behavior of checked contracts, and whether they can have side effects.

1.3.3 P1448R0 – Simplifying Mixed Contract Modes

This paper focuses on how we want to standardize how *build mode* interacts with modules.

1.3.4 P1672R0 – "Axiom" is a False Friend

This name change is relevant to any state of affairs that still includes axiom as a contract level.

1.3.5 P1671R0 – Contract Evaluation in Constant Expressions

This paper focuses on how assumed contracts should behave during constant expression evaluation, notably allowing them (and the compiler) the same leeway that currently exists for contracts on standard library functions.

2 All Papers

Below is our attempt at an object, though probably at least partially biased, summary of all papers related to contracts since the post-San Diego mailing.

2.1 Mailing: Post-San Diego

P1332R0 *Contract Checking in C++: A (long-term) Road Map* link

Authors: Joshua Berne, Nathan Burgers, Hyman Rosen, John Lakos

Tags: should be EWG, contracts, closed

An overview of why contracts should be designed in terms of semantics, an initial proposal for roles on top of that. P1607 and P1429 are the currently proposed for C++20 parts of this paper, and P1606 discusses additional aspects of the needs for a design for roles.

P1333R0 *Assigning Concrete Semantics to Contract-Checking Levels at Compile Time* link

Authors: Joshua Berne, John Lakos

Tags: (should be EWG, contracts, superseded by P1429R0)

An initial proposal for redefining contracts in terms of semantics and allowing arbitrary mapping of levels to semantics.

P1334R0 *Specifying Concrete Semantics Directly in Contract-Checking Statements* link

Authors: Joshua Berne, John Lakos

Tags: (should be EWG, contracts, superseded by P1429R0)

A proposal on top of P1333R0 to allow specifying literal semantics.

P1335R0 *"Avoiding undefined behavior in contracts" [P1290R0] Explained* link

Authors: John Lakos

Tags: (should be contracts, info, closed)

A redescription of P1290R0 in terms of the semantics defined by P1333R0.

2.2 Mailing: Pre-Kona

P1290R1 *Avoiding undefined behavior in contracts* link

Authors: J. Daniel Garcia, Ville Voutilainen

Tags: EWG, contracts

Proposes an axiom mode to control assumption of axiom level contracts, removing assumption of default and audit level checks.

P1296R0 *[[assert: std::disjoint(A,nA, B,nB)]]: Contract assertions as an alternate spelling of âĂŸrestrictâĂŹ* link

Authors: Phil Miller, Justin Szaday

Tags: EWG-I, contracts issue

An overview of how contracts might be used to implement the functionality of `restrict` or `__restrict__`.

P1320R1 *Allowing contract predicates on non-first declarations* link

Authors: Ville Voutilainen

Tags:EWG, contracts

Proposes allowing member function redeclaration and allowing contracts to be on any declaration and not just the first declaration.

P1344R0 *Pre/Post vs. Enspects/Exsures* link

Authors: Nathan Myers

Tags: EWG, contracts

Alter the tokens "expects" and "ensure" to "pre" and "post". Approved in Kona.

P1421R0 *Assigning semantics to different Contract Checking Statements* link

Authors: Andrzej Krzemieński,

Tags: contracts, info, closed

An informative discussion of roles, levels, and a suggestion to adopt semantics and simplified arbitrary-identifier roles.

P1426R0 *Pull the Plug on Contracts?* link

Authors: Nathan Myers

Tags: EWG, contracts, closed

A proposal to remove contracts from the standard, not accepted in Kona.

P1429R0 *Contracts That Work* link

Authors: Joshua Berne, John Lakos

Tags: EWG, contracts

Proposal for redefining contracts in terms of semantics, allowing literal semantics, and allowing arbitrary level->semantic assignment, presented in Kona.

P1448R0 *Simplifying Mixed Contract Modes* link

Authors: Nathan Burgers

Tags: EWG, contracts – scheduled Wednesday issue

A discussion of the impacts and meaning of heterogeneous builds, with a suggestion for a fix that has the build mode of the definition take priority (specializing the case for inline functions which allow the build mode of any definition to take effect, but leave which specific one unspecified).

2.3 Mailing: Post-Kona

P1290R2 *Avoiding undefined behavior in contracts* link

Authors: J. Daniel Garcia, Ville Voutilainen

Tags: EWG, contracts issue

Proposes removing assumption from unchecked default and audit contracts, adding an "axiom mode" to control assumption of axiom level contracts, add meaning for "continue" on contracts to choose continuation in code, removing global continuation mode.

P1290R3 *Avoiding undefined behavior in contracts* link

Authors: J. Daniel Garcia, Ville Voutilainen

Tags: EWG, contracts – scheduled Monday issue

Proposes the removing assumption from unchecked default and audit contracts. Proposes considering that "axiom contracts are considered as if they had been evaluated when they are enabled" with some incomplete indication that it might be possible to not enable axioms, but no concrete proposal in that direction.

- P1290R3 – Remove assumption of default/audit, add axiom mode.

P1344R1 *Pre/Post vs. Enspects/Exsures* link

Authors: Nathan Myers

Tags: CWG, contracts issue

Alter the tokens "expects" and "ensure" to "pre" and "post". Now with wording.

P1429R1 *Contracts That Work* link

Authors: Joshua Berne, John Lakos

Tags: EWG, contracts

Updates to P1429R0 resulting from discussion in Kona, mostly to details of the wording. Superseded completely by P1429R2/3.

P1486R0 *United Amendment to Contracts Facility for C++20* link

Authors: John Lakos

Tags: contracts, info issue

A step towards determining an acceptable design compromise after the voting that occurred in Kona on contract desing.

P1487R0 *User Experience with Contracts That Work* link

Authors: John Lakos

Tags: contracts (should be info) issue

A informative history of Bloomberg's use of contracts and how they led to the proposals that culminate in P1429.

P1490R0 *Contract-Related Issues* link

Authors: Andrzej Krzemieński,

Tags: contracts, info issue

An overview discussion of contracts and assumption, focusing on how contracts can be introduced and what impact assumption has on them. No direct suggestion for a solution.

P1494R0 *Partial program correctness* link

Authors: S. Davis Herring

Tags: EWG, LEWG, contracts issue

A proposal for adding an observable marker to the standard which can be used to prevent time traveling of undefined behavior (ideally suited to be included in the definition of 'check_maybe_-continue' from P1429).

P1517R0 *Contract Requirements for Iterative High-Assurance Systems* link

Authors: Ryan McDougall

Tags: contracts, info issue

An informative paper discussing the needs of safety-critical systems and what needs they have for contracts, especially for how they will interact with assuming of unchecked contracts. Suggests changing axiom and providing a facility that is safer to use than the draft.

P1606R0 *Requirements for Contract Roles* link

Authors: Joshua Berne

Tags: contracts, info issue

An informative paper discussing what design decisions need to be considered for more general 'roles' as proposed in P1332

P1607R0 *Minimizing Contracts* link

Authors: Joshua Berne, Jeff Snyder, Ryan McDougall

Tags: EWG, contracts – scheduled Monday issue

Proposes the same semantic redefinition as P1429, but different exposed options.

- P1607R0:1 – Removal of all but default contracts, which can only be 'check_never_continue' or 'ignore'.
- P1607R0:1 – Removal of all but default contracts, which can only be 'check_never_continue' or 'ignore', add (4) in-code literal semantics.

P1625R0 *Contracts: why the house is not on fire (i.e. why the status quo is tolerable)* link

Authors: Ville Voutilainen

Tags: (should be EWG, contracts), closed

Proposes that the status quo is ok, has not resurfaced after the post-Kona mailing.

2.4 Mailing: Pre-Cologne

N4820 *Working Draft, Standard for Programming Language C++* link

Authors: Richard Smith

The current proposed working paper. Contracts as they exist in this paper include a *build level* and *continuation mode*, and are always assumed when not checked.

P1320R2 *Allowing contract predicates on non-first declarations* link

Authors: Ville Voutilainen

Tags: EWG, contracts – scheduled Monday issue

Motivates and defines the ability to have contracts be declared for a function on declarations other than the first declaration of that function (up to and including the definition of the function). Proposes also allowing redeclaration of member functions to provide an additional position to declare implementation details for a class (alongside out-of-class inline functions) while still keeping them in the header and available to client compilers.

- P1320R2:1 – Allow contracts on any declaration
- P1320R2:2 – Allow contracts on any declaration, allow member function redeclaration

P1429R2 *Contracts That Work* link

Authors: Joshua Berne, John Lakos

Tags: EWG, contracts – (scheduled Monday, but superseded with new revision).

A proposal to redefine contracts in terms of their effective semantics, including the ability to specify those semantics in code as well as assign the levels independently to those semantics.

P1639R0 *Unifying source_location and contract_violation* link

Authors: Corentin Jabot

Tags: LEWG, contracts issue

A library proposal to change 'std::contract_violation' to use 'std::source_location' and 'const char *' instead of 'std::string_view'.

P1670R0 *Side Effects of Checked Contracts and Predicate Elision* link

Authors: Joshua Berne, Alisdair Meredith

Tags: EWG, contracts – scheduled Wednesday issue

Proposes changing how side effects are treated in contracts so that they can never be depended on, but don't actually introduce language-level UB simply by existing.

P1671R0 *Contract Evaluation in Constant Expressions* link

Authors: Joshua Berne, Alisdair Meredith

Tags: EWG, contracts – scheduled Wednesday issue

Proposes changing the effects of contracts in constant expressions to be more consistently in line with what standard library contracts can do, allowing "assumed" contracts to be validated at compile time if possible, but not requiring it.

P1672R0 *"Axiom" is a False Friend* link

Authors: Joshua Berne

Tags: EWG, contracts – scheduled Wednesday issue

Proposes changing the token 'axiom' to something less distracting – i.e. any other sequence of characters.

P1680R0 *Implementing Contracts in GCC* link

Authors: Andrew Sutton, Jeff Chapman

Tags: contracts, info issue

Informative paper discussing the results of implementing the P1429 semantics in GCC, as well as implementing most of the other design proposals on top of that. Section 5 is also relevant to P1320R2.

P1704R0 *Undefined functions in axiom-level contract statements* link

Authors: Andrzej Krzemieński, Joshua Berne

Tags: EWG, contracts – scheduled Wednesday issue

Proposes a change to the definition of axiom level contracts so they no longer ODR-use their predicates.

P1710R0 *Adding a global contract assumption mode* link

Authors: Ville Voutilainen

Tags: EWG, contracts, closed (superseded) issue

Proposes adding a single extra flag that controls whether unchecked contracts introduce undefined behavior or not. P1711R0 includes the same proposal as an option, and P1730R0 proposes the same solution with more/different motivation.

P1711R0 *What to do about contracts?* link

Authors: Bjarne Stroustrup

Tags: EWG, contracts – scheduled Monday issue

Proposes 4 alternatives for design changes for contracts:

- P1711R0:1 – The status quo.
- P1711R0:2 – Remove contracts entirely.
- P1711R0:3 – (Minimal change) Add *assumption mode*.
- P1711R0:4 – (Minimal feature) Add *assumption mode*, remove *continuation mode*.

Suggests 1 or 4 are the preferred alternatives.

P1728R0 *Preconditions, axiom-level contracts and assumptions – an in depth study* link

Authors: Andrzej Krzemieński,

Tags: contracts, info issue

Discusses contracts, the appropriateness of the word axiom for use in contracts, how contracts can be leveraged for static analysis and runtime checking, and some of the implications of undefined behavior on program behavior.

P1730R0 *Adding a global contract assumption mode* link

Authors: Hyman Rosen, John Lakos, Alisdair Meredith

Tags: contracts, info – scheduled Monday issue

Proposes adding a single extra flag that controls whether unchecked contracts introduce undefined behavior or not. Same implementation but additional motivation on top of P1710R0. Also the same proposal as P1711R0:3.

P1743R0 *Contracts, Undefined Behavior, and Defensive Programming* link

Authors: Rostislav Khlebnikov, John Lakos

Tags: contracts, info issue

A white paper discussing the meaning of defensive programming through contracts.

P1744R0 *Avoiding Misuse of Contract-Checking* link

Authors: Rostislav Khlebnikov, John Lakos

Tags: contracts, info issue

A white paper discussing the intended uses of a contract checking facility to help bring understanding for making design decisions about contracts.

P1769R0 *The "default" contract build-level and continuation-mode should be implementation-defined* link

Authors: Ville Voutilainen

Tags: EWG, contracts – scheduled Monday issue

Proposes to remove stating a default from the standard itself, as it has no teeth. Compatible with any other design changes.

P1773R0 *Contracts have failed to provide a portable "assume"* link

Authors: Timur Doumler

Tags: EWG, contracts, closed (should be info) issue

A discussions of the needs for assumption, how they overlap with contracts, how the status quo fails to satisfy them, the history of the relationship between the two, and a suggestion to adopt P1607 or P1429 to satisfy most or all of those needs.

P1774R0 *Portable optimisation hints* link

Authors: Timur Doumler

Tags: EWG issue

A proposal for adding 'std::assume' in some form, including questions on whether the syntax should overlap with contracts or not.

P1782R0 *Local contract restrictions* link

Authors: S. Davis Herring

Tags: EWG, contracts – scheduled Monday issue

An alternate design proposal, replacing contract mode with any of the 32 combinations of "tentative, halt, static, audit, always".

P1786R0 *Adding a global contract assumption mode* link

Authors: Hyman Rosen, John Lakos, Alisdair Meredith

mistake, removed, see P1730R0

P1793R0 *Simplifying Contract Syntax* link

Authors: Alisdair Meredith

Tags: EWG, contracts – scheduled Monday issue

A proposal to remove confusing use of "default" and name the contract levels "assert/audit/axiom", simplifying syntax for in-function contracts.

2.5 On-Wiki/Mailing: Post-Cologne

D1429R3 *Contracts That Work* link

Authors: Joshua Berne, John Lakos

Tags: EWG, contracts – scheduled Monday issue

A proposal to redefine contracts in terms of their effective semantics, including the ability to specify those semantics in code as well as assign the levels independently to those semantics. Proposes a number of alternatives to consider, with a few major decisions regarding build time configuration:

- D – Discard *build level* and *continuation mode*.
- B – Keep *build level* and *continuation mode* (B is for Build Mode).
- A – Keep *build level* and *continuation mode*, add *assumption mode* (A is for Assumption Mode).
- L – Add arbitrary assignment of levels to semantics (L is for Levels).
- G – Add a *global contract mode* (which also disables contracts with literal semantics, G for Global Contract Mode).

This leads to the following (10) proposed variations, with a recommendation that arbitrary assignment be allowed, and if the status quo options are kept then *assumption mode* should be removed.

- D1429R3:DL – Add literal semantics, add arbitrary assignment of levels to semantics

- D1429R3:DLG – Add literal semantics, add arbitrary assignment of levels to semantics, add *global contract mode*
- D1429R3:B – Add literal semantics, keep *build level* and *continuation mode*
- D1429R3:BG – Add literal semantics, keep *build level* and *continuation mode*, add *global contract mode*
- D1429R3:A – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*
- D1429R3:AG – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*, add *global contract mode*
- D1429R3:BL – Add literal semantics, keep *build level* and *continuation mode*, add arbitrary assignment of levels to semantics
- D1429R3:BLG – Add literal semantics, keep *build level* and *continuation mode*, add arbitrary assignment of levels to semantics, add *global contract mode*
- D1429R3:AL – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*, add arbitrary assignment of levels to semantics
- D1429R3:ALG – Add literal semantics, keep *build level* and *continuation mode*, add *assumption mode*, add arbitrary assignment of levels to semantics, add *global contract mode*