CoCoALib - Design #894

strict enum types: C++11 extension

21 Jun 2016 14:47 - John Abbott

Status:	In Progress	**Start date:**	21 Jun 2016
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	30%
Category:	Safety	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	4.65 hours

Description

Apparently C++11 offers "strongly typed enum" which do not have automatic conversion to integer values.

This seems like a good idea (in terms of cleanliness and safety), so consider adopting it for the enums which we use.

Related issues:

Related to CoCoA-5 - Design #83: C++11 compatibility questions	**In Progress**	**26 Jan 2012**
Related to CoCoALib - Feature #82: C++11 compatibility questions	**Closed**	**26 Jan 2012**
Related to CoCoALib - Design #1225: Move to C++14 (skipping C++11)	**In Progress**	**06 Sep 2018**

History

#1 - 21 Jun 2016 14:47 - John Abbott

- Related to Design #83: C++11 compatibility questions added

#2 - 26 Jun 2018 15:21 - John Abbott

- Related to Feature #82: C++11 compatibility questions added

#3 - 12 Mar 2020 14:41 - John Abbott

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99800

#4 - 28 Oct 2020 23:20 - John Abbott

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

#5 - 02 Mar 2021 08:52 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

This is what I got searching for enum in the CoCoA-5 sources:

```
AST.H:    enum ImportType { IT_TOPLEVEL, IT_BYREF, IT_BYVALUE };
C5.H:    enum {
Interpreter.H:    enum Flags {
Interpreter.H:    enum EvalKind { EVAL_BY_REF, EVAL_BY_VALUE };
Interpreter.H:    enum InterpreterStatus {
Lexer.H:enum TokenType {
Lexer.H:enum WarningSeverity {
Parser.H:        enum ScopeType { ST_TOPLEVEL, ST_PACKAGE, ST_DEFINE, ST_LAMBDA };
```

#6 - 02 Mar 2021 08:54 - John Abbott

This is what I got searching for enum in the CoCoALib sources: **(I'll remove lines after having revised the files)**

```
bool3.H:    enum TruthValueSet { false3, uncertain3, true3 }; // not enum class: I want the names to be inject
ed!
DynamicBitset.H:    enum OutputStyle {clean, AsRevVecOfLong, WithSeparators};
ExternalLibs-MathSAT.H:  enum RelOp {eq0, neq0, leq0, lt0};
GBEnv.H:    enum type {Field, FrFldOfGCDDomain};
GBEnv.H:enum ComputationInputAndGradingType
MemPool.H:    enum FillNewLoaf_t {DontFillNewLoaf, FillNewLoaf}; // enum to avoid passing a bool argument in t
he ctor
NumTheory-ContFrac.H:  enum class PlusOrMinusEpsilon { MinusEpsilon, ZeroEpsilon, PlusEpsilon };
PolyRing.H:// - added  DontSkipLMg  in  SkipLMFlag  enum
SparsePolyRing.H:    enum SkipLMFlag { SkipLMg, DontSkipLMg }; // used only for AddMul
SugarDegree.H:  enum UninitializedMarker { uninitialized }; // just used for sugar ctor
TmpGPoly.H:enum ClearMarker {clear};
TmpGPoly.H:    enum ReductionFlag { Full, OnlyTail };
TmpGPoly.H:    enum UseBorelFlag { UseBorel, DontUseBorel };
TmpGPoly.H:// -- added SaturatingAlgNoDRL  to enum
TmpGReductor.H:  enum CoprimeFlag   { UseCoprime, DontUseCoprime };
TmpGReductor.H:  enum GMFlag        { UseGM, DontUseGM };
TmpGReductor.H:  enum BackFlag      { UseBack, DontUseBack };
TmpGReductor.H:  enum DivFlag       { UseDiv, DontUseDiv }; ///< remove poly if its LPP is divisible by LPP of
 new poly; true except for RingWeyl
TmpGReductor.H:  enum AllSetMarker  { AllSet };
TmpGReductor.H:    enum UseDynamicAlgFlag    { UseDynamicAlg, DontUseDynamicAlg };
TmpGReductor.H:    enum BuchbergerOpTypeFlag { HomogeneousAlg, SaturatingAlg, AffineAlg };
TmpGReductor.H:    enum ModOrdTypeForcing {NoForcing,PosWDegTO,WDegTOPos,WDegPosTO};
TmpGReductor.H:// -- enum instead of bool arguments
TmpGReductor.H://    Reductors: field: IhaveBorelReductors;  type: enum UseBorelMarker
TmpJBMill.H:    enum StrategyFlag {TQDegree, TQBlockHigh, TQBlockLow, GBCompletion};
TmpJBSets.H:      enum Strategy {
TmpPBMill.H:       * This enum represents the different strategies:
TmpPBMill.H:      enum StrategyFlag
TmpPBMill.H:       * This enum represents the different strategies:
TmpPBMill.H:      enum StrategyFlag
TmpPBMill.H:       * This enum represents the different strategies:
TmpPBMill.H:      enum StrategyFlag
TmpStabilityAlgorithm.H:      enum StatisticLevel {None, Tracking, Logging};
TmpStabilityAlgorithm.H:       * This enum represents the different strategies for the usage of variable permu
tations to
TmpStabilityAlgorithm.H:      enum UsagePermutations
```

#7 - 05 Mar 2021 09:22 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 30

Where I have felt that it is useful to inject the names from an enum,
I have used the following technique to reproduce the name injection of old style enums:

```
enum class ENUM { NAME1, NAME2 };
constexpr ENUM NAME1 = ENUM::NAME1;
constexpr ENUM NAME2 = ENUM::NAME2;
```

Note that C++20 lets one use **using** to export all the enum names in this way.

#8 - 28 Jan 2022 13:06 - John Abbott

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

#9 - 16 Mar 2024 21:38 - John Abbott

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

I wonder whether it might not be better to wait until we switch to the C++20 standard, and so can use **using ENUM**.
In many instances we are using an enum local to a class, so we are not really polluting the global namespace.

The technique from #note-7 would work well enough, and could act as a stop-gap until we switch to C++20. Is it worth the hassle and time to make the changes though?

#10 - 25 Apr 2024 21:49 - John Abbott

This is one of those *mindless* tasks to be done when the brain is taking a nap...