NemoMath - Feature #821
Allow optional 'abort-on-error' instead of exception

01/23/2012 03:49 PM - M. Rolf

Status:	Resolved	**Start date:**	01/23/2012
Priority:	Normal	**Due date:**	
Assignee:	M. Rolf	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	NemoMath 0.4		

Description

Benchmarked operator[] and *begin() for both read and write operations on revision @191 on malachit with GCC 4.4.3

When throwing exception on 'array out of range':

```
    BenchCase: "VectorCreateOperatorRead"
        Estimated cost per operation:   0.00637619 microseconds
    BenchCase: "VectorCreateOperatorReadConst"
        Estimated cost per operation:   0.000708122 microseconds
    BenchCase: "VectorCreateOperatorWrite"
        Estimated cost per operation:   0.00803425 microseconds
    BenchCase: "VectorCreateIteratorRead"
        Estimated cost per operation:   0.00070811 microseconds
    BenchCase: "VectorCreateIteratorWrite"
        Estimated cost per operation:   0.00283319 microseconds
```

The same benchmark, when calling 'abort' on 'array out of range':

```
    BenchCase: "VectorCreateOperatorRead"
        Estimated cost per operation:   0.000708186 microseconds
    BenchCase: "VectorCreateOperatorReadConst"
        Estimated cost per operation:   0.000708156 microseconds
    BenchCase: "VectorCreateOperatorWrite"
        Estimated cost per operation:   0.00283341 microseconds
    BenchCase: "VectorCreateIteratorRead"
        Estimated cost per operation:   0.00070805 microseconds
    BenchCase: "VectorCreateIteratorWrite"
        Estimated cost per operation:   0.00283317 microseconds
```

Non-const operator[] read shows factor 9 speedup, operator[] write factor 3.
In contrast, GCC 4.6.1 shows almost no difference: All costs are on the low level of IteratorRead/Write, except a marginal increase of runtime in "VectorCreateOperatorWrite".

It seems that the older GCC largely benefit from a never-returning error mechanism.
Allow to use abort-on-error using a compiler-flag...

History

#1 - 01/15/2013 05:51 PM - M. Rolf

- Status changed from New to In Progress

#2 - 01/16/2013 01:39 PM - M. Rolf

- % Done changed from 0 to 50

#3 - 01/18/2013 04:59 PM - M. Rolf

- Status changed from In Progress to Resolved

- % Done changed from 50 to 100

Implemented new CMake option ABORT_ON_ERROR (false by default), which activates the compile-time definition -DNEMO_ABORT_ON_ERROR.

When enabled, all exceptions (for example when an index is out of range when accessing a MathVector) are replaced by an error message posted to std::cerr, and an abort()-call which stops the entire program.

Benchmarks with ABORT_ON_ERROR=false (default):

```
   Benchsuite [MathVectorAccess]
     Performing each case 1000000000 times with operation-size 10
     Benchcase [MathVectorAccess:OperatorRead]
         Estimated cost per operation:    0.00699068 us
     Benchcase [MathVectorAccess:OperatorReadConst]
         Estimated cost per operation:    0.00104675 us
     Benchcase [MathVectorAccess:OperatorWrite]
         Estimated cost per operation:    0.00959644 us
     Benchcase [MathVectorAccess:IteratorRead]
         Estimated cost per operation:    0.00070806 us
     Benchcase [MathVectorAccess:IteratorWrite]
         Estimated cost per operation:    0.00283256 us
   Benchsuite [MatrixAccess]
     Performing each case 1000000000 times with operation-size 10
     Benchcase [MatrixAccess:OperatorRead]
         Estimated cost per operation:    0.00596268 us
     Benchcase [MatrixAccess:OperatorReadConst]
         Estimated cost per operation:    0.00597869 us
     Benchcase [MatrixAccess:OperatorWrite]
         Estimated cost per operation:    0.00710284 us
     Benchcase [MatrixAccess:IteratorRead]
         Estimated cost per operation:    0.00105505 us
     Benchcase [MatrixAccess:IteratorWrite]
         Estimated cost per operation:    0.00292096 us
```

Benchmarks with ABORT_ON_ERROR=true (some cases x10 faster, due to aggressive optimization of never-returning error-handling):

```
   Benchsuite [MathVectorAccess]
     Performing each case 1000000000 times with operation-size 10
     Benchcase [MathVectorAccess:OperatorRead]
         Estimated cost per operation:    0.000708147 us
     Benchcase [MathVectorAccess:OperatorReadConst]
         Estimated cost per operation:    0.00106242 us
     Benchcase [MathVectorAccess:OperatorWrite]
```

```
        Estimated cost per operation:    0.00283296 us
    Benchcase [MathVectorAccess:IteratorRead]
        Estimated cost per operation:    0.000708058 us
    Benchcase [MathVectorAccess:IteratorWrite]
        Estimated cost per operation:    0.00319325 us
Benchsuite [MatrixAccess]
    Performing each case 1000000000 times with operation-size 10
    Benchcase [MatrixAccess:OperatorRead]
        Estimated cost per operation:    0.000708069 us
    Benchcase [MatrixAccess:OperatorReadConst]
        Estimated cost per operation:    0.000708512 us
    Benchcase [MatrixAccess:OperatorWrite]
        Estimated cost per operation:    0.00283275 us
    Benchcase [MatrixAccess:IteratorRead]
        Estimated cost per operation:    0.00104229 us
    Benchcase [MatrixAccess:IteratorWrite]
        Estimated cost per operation:    0.00292152 us
```