Boost.FixedPoint

I would like to thank Vicente J. Botet Escriba and Boost community for their feedback in creating this proposal.

Abstract

“Fixed-point number representation is a real data type for a number that has a fixed number of digits after (and sometimes also before) the radix point (after the decimal point '.' in English decimal notation). Fixed-point Number representation can be compared to the more complicated (and more computationally demanding) floating-point number representation.

Fixed-point numbers are useful for representing fractional values, usually in base 2 or base 10, when the executing processor has no floating point unit (FPU) or if fixed-point provides improved performance or accuracy for the application at hand . Most low-cost embedded microprocessors and microcontrollers do not have an FPU.”

Wikipedia

Fundamental C++ types (like integer and floating) have many problems which can solved by fixed-point arithmetic. C++ does not support built-in fixed-point numbers. I propose extending the Boost Libraries to provide binary fixed-point arithmetic.

Personal Details

Name: Dmitriy Gorbel

Email: dmitriycpp@gmail.com

Location: Kharkiv, Ukraine, UTC+3

Program: Bachelor of Computer Science

University: National Technical University "Kharkiv Polytechnic Institute"

Proposal

Introduction

Goal of the project - provide an implementation of a FixedPoint library based on the C++1y proposal. Therefore I use it as the specification. I think, the binary fixed-point type is a good choice, because the rescaling operations can be implemented as fast bit shifts. I will use implementation by Vicente J. Botet Escriba as a prototype.

Fundamental types in C++, like integer and floating types, have some problems. For example, integer types have platform-depended ranges, signed integer overflow causes undefined behavior. Floating types also have problems like limited exponent range, loss of significance, unsafe standard operations, rounding error, etc. A lot of systems do not have hardware support for floating-point operations.

Fixed-point arithmetic resolves this problems. I think main advantages of the fixed-point numbers is determinism and customizable range and resolution, hence more efficient.

Specification

C++1y proposal suggest extending the standard library to provide general purpose binary fixed-point arithmetic. Fixed-point library will provide *nonnegative* and *negatable* class templates for fractional arithmetic, and *cardinal* and integral for integer arithmetic. In total, four class templates.

The range and the resolution(resolution only for fractional types) of types can be set by an integer. The range must be greater then the resolution. For example:

cardinal<16>	0 <= n < 65536
integral<4>	-16 < n < 16
nonnegative<8,-4>	0 <= *n* < 256 in increments of 2^{-4} = 1/16
negatable<16,-8>	-65536 < *n* < 65536 in increments of 2^{-8} = 1/256

This notation required by C++1y proposal, but there are other popular notations. I will implement at least one alternative notation, for example like this: *FractionalArithmeticClass*<Magnitude bits, Fractional Bits> .

“The template class type signatures are as follows

```
template<
    int Crng,
    overflow Covf = overflow::exception
>
class cardinal;

template<
    int Crng,
    overflow Covf = overflow::exception
>
class integral;

template<
    int Crng,
    int Crsl,
    round Crnd = round::nearest_odd,
    overflow Covf = overflow::exception
>
class nonnegative;

template<
    int Crng,
    int Crsl,
    round Crnd = round::nearest_odd,
    overflow Covf = overflow::exception
>
class negatable;”
```

C++1y proposal

Fixed-point library will provide next basic arithmetic operations: addition, subtraction, multiplication and division. Subtraction/negation on unsigned types produce a signed type, mixing of different types produce result with negatable type(of course except cardinal class type – if cardinal type mixing with another type result have type of second operand).

Resolution and range of the result of basic operations, or conversions must be large enough to store the correct value, and expand according to this. But library must provide alternative variant, when precision bounded and behavior may be defined by user.

Fixed-point allows numbers representation with huge resolution, but in some spesial cases, for example division, or when user choose closed arithmetic(when result have the same number of bits as the operands) value must be rounded to a representable size. Library will provide pretty rounding modes, which can be set programmatically, with a value of type *`enum class round`*.

Application behavior in case when an overflow will happened may be defined programmatically, with a value of type *`enum class overflow`*.

I want to add some math functions, which must work similar to standard C function with same name, but with fixed point numbers.
List of functions: ceil, floor, sqrt, sin, cos, exp, fabs, fmod, modf, exp. Also, some functions can be added to the list.

For more information and details of the specification see C++1y proposal.

Timeline

I plan spend on Boost.fixedpoint project around 30 - 40 hours per week. I will spend much time for testing – I think it's very important, my code must be clean and effective.

So far as in project has four main class templates, here will be four identical development phase for each class template.

Phase take 20 day and composed of(days):

- Design and Code(10);
- Testing(2);
- Bug fix and optimization(3);
- Writing documentation(2). Documentation for each class include sections: Tutorial, Examples, Reference, Rationale;
- Provide results to Boost community. Getting feedback and correct project with suggestion from the feedback(3);

Development phase for the math functions module will be similar, but twice shorter and take 10 days.

Present time – May 31

Getting closer with Boost community, Boost libraries, Boost coding standards, organization of the documentation. Explore the problem domain, the alternatives, and make design choice.

June 1 - June 20

Development cardinal class template. Provide intermediate results to Boost community at June 18.

June 21 - July 10

Development integral class template. Provide intermediate results to Boost community at July 8.

July 11 - July 30

Development nonnegative class template. Provide intermediate results to Boost community at July 28.

July 31 - August 19

Development negatable class template. Provide intermediate results to Boost community at August 17.

August 20 - August 29

Development module with math functions.

August 30 - September 8

Final testing and bug fix. Preparing to release. Submit results to Boost at 7 September.

September 9 - September 15

Write more examples with FixedPoint library.

September 16 - September 22

I will use this week for developing report – with detailed description, comparing with other implementation, performance testing. Submit final result and report.

About Me

Goals and motivations

I really enjoy open-source software - I use GNU/Linux and many open-source programs and tools. Also I respect Boost libraries - there is a great wealth of cool libraries. I understand that if I involved in Boost project, I must write very clean and very effective code. I think it would be great experience, so I want to get closer with Boost.

Education

Kharkiv Radiotechnical College,
Junior Specialist in Computer Science,
graduated with distinction in 2010

National Technical University "Kharkiv Polytechnic Institute" ,
4th course student

Experience

Operating systems

I most familiar with Linux, also I have experience with Mac OS X and iOS development.

C++

I have 3 years of experience in programming with C++, include 1,5 years of work as software developer.

Programming Paradigms

I familiar with Procedural, Object Oriented and Generic programming paradigms, I frequently use OOP in educational projects and projects from my job. Also I use Generic programming (C++ templates).

Source Control

I have experience with Subversion, I use SVN in my work, also I have basic knowledge about git.

Design Patterns

I familiar with Design Patterns. In my current project I use patterns like: Observer, Singleton, Iterator, MVC Pattern.

Libraries

I have experience with QT framework – I used it in educational projects and projects from my job. Unfortunately, I haven't much experience with Boost – I just used few libraries for training projects, include thread, chrono, foreach, lexical_cast. I want to fix this, so I have been exploring Boost in these latter days.

Favorite IDE

QTCreator, Geany, Xcode