The C++ `fPoint` class

E. Robert Tisdale

May 15, 1999

Abstract

The C++ `fPoint` class library is used to implement numerical algorithms which will execute in either floating-point or fixed-point mode in order to help convert numerical algorithms from floating-point arithmetic to fixed-point arithmetic. Variables of type `fPoint` are actually represented by high-precision floating-point numbers and all operations on variables of type `fPoint` are implemented with high-precision floating-point arithmetic. In floating-point mode, the fixed-point range and precision are ignored and the assignment operation gathers statistics on the floating-point result which help determine range and precision requirements for the fixed-point variable. In fixed-point mode, the assignment operation coerces the floating-point result into the range and precision specified for the fixed-point variable. Operations on variables of type `fPoint` which return a floating-point result by value may not be combined with other operations in an expression unless the intermediate result is first assigned to a variable of type `fPoint`.

1 Introduction

1.1 Fixed-point arithmetic

A fixed-point number is simply a denormalized floating-point number with a constant exponent. A sign and magnitude representation is typical for a floating-point mantissa but a 2's complement representation is more common for a fixed-point mantissa. A binary fixed-point number

$$x = m \cdot 2^e \tag{1}$$

where the exponent e is a constant integer and the 2's complement mantissa

$$m = -m_0 + \sum_{k=1}^{p} m_k \cdot 2^{-k} \tag{2}$$

where the mantissa bit $m_k \in \{0, 1\}$, the most significant mantissa bit m_0 is the sign bit and $p \geq 0$ is the position of the least significant mantissa bit. The fixed-point range

$$-2^e \leq x \leq +2^e - 2^{e-p} \tag{3}$$

is restricted by the range

$$-1 \leq m \leq +1 - 2^{-p} \tag{4}$$

of the mantissa.

Numerical algorithms are sensitive to both method errors and representation errors but these errors are independent of each other. The numerical analyst simply assumes that the method used to compute the result for any operation on finite precision variables is exact or at least sufficiently accurate to determine the nearest value of the finite precision variable to which the result is assigned.

If the result x is assigned to fixed-point variable, it is truncated

$$x \leftarrow \lfloor x \cdot 2^{p-e} \rfloor 2^{e-p} \tag{5}$$

or rounded

$$x \leftarrow \lfloor x \cdot 2^{p-e} + 1/2 \rfloor 2^{e-p} \tag{6}$$

to the nearest multiple of 2^{e-p} then it wraps around

$$x \leftarrow (x + 2^e) \bmod 2 \cdot 2^e - 2^e \tag{7}$$

the interval $[-2^e, +2^e]$ or saturates

$$x \leftarrow \begin{cases} -2^e & \text{if } x < -2^e \\ +2^e - 2^{e-p} & \text{if } +2^e - 2^{e-p} < x \end{cases} \tag{8}$$

unless $x \in [-2^e, +2^e - 2^{e-p}]$.

2 `class fPoint`

A variable of type `fPoint` has a high-precision floating-point representation and a reference to a fixed-point specifier of type `fPoint::Specification`. Operations on variables of type `fPoint` which compute a floating-point result return type `fPoint::Result` by value.

2.1 `class fPoint::Specification`

The `fPoint::Specification` class is derived from the `fPoint::Statistics` class. It includes constant specifications for fixed-point range and precision, fixed-point rounding, saturation and representation controls and a character string which can be used to describe the specification.

2.1.1 Controls

The `FlagBits` enumeration

fPoint::Specification::FlagBits	description
fPoint::Specification::ZeroBits	clear all control flags
fPoint::Specification::Saturate	saturate on overflow
fPoint::Specification::Roundoff	round to the nearest
fPoint::Specification::Signless	the mantissa is unsigned
fPoint::Specification::SignMagnitude	sign and magnitude mantissa

The `flagbits` type definition

type	synonym
unsigned int	fPoint::Specification::flagbits

2.1.2 Static Data Members

type	default	initial value
int	fPoint::Specification::DefaultExponent	0
unsigned int	fPoint::Specification::DefaultPosition	15
flagbits	fPoint::Specification::DefaultControls	ZeroBits
const char*	fPoint::Specification::DefaultNotation	""

2.1.3 Constructors

The explicit constructor `fPoint::Specification(exponent, position, controls, notation)` where

type	argument	description
int	exponent	is the fixed-point exponent,
unsigned int	position	is the least significant bit position,
flagbits	controls	contains control flag bits and
const char*	notation	is descriptive notation.

Each argument assumes its respective default value if omitted.

2.1.4 Member Functions

`exponent()` returns the exponent.

`position()` returns the least significant bit position.

`controls()` returns the controls for saturation, rounding and representation.

`notation()` returns a pointer to the notation character string.

`saturate()` returns `true` if the `fPoint::Specification::Saturate` control flag is set.

`roundoff()` returns `true` if the `fPoint::Specification::Roundoff` control flag is set.

`signless()` returns `true` if the `fPoint::Specification::Signless` control flag is set.

`signmagnitude()` returns `true` if the `fPoint::Specification::SignMagnitude` control flag is set.

`roundoff(x)` returns `x` truncated or rounded to the nearest multiple of 2^{e-p} if $0 \leq x < 2^e$ where $e =$ `exponent()` and $p =$ `position()`.

`coerce(x)` returns `x` truncated or rounded to the nearest multiple of 2^{e-p} then wrapped around the interval $[-2^e, +2^e]$ or saturated if it is not in the interval $[-2^e, +2^e - 2^{e-p}]$ where $e =$ `exponent()` and $p =$ `position()`.

2.1.5 Operators

`os << s` writes the specification `s` to output stream `os` then returns a reference to `os`.

`is >> s` reads a the specification from input stream `is` and discards it then returns a reference to `is`.

2.2 `class fPoint::Statistics`

The `fPoint::Statistics` class accumulates statistics from all of the variables of type `fPoint` which reference it. It collects and reports different statistics in floating-point mode than it does in fixed-point mode. In floating-point mode, it counts all the non-zero finite floating-point numbers assigned to all the variables which reference it and computes the minimum, average and maximum floating-point exponent and the variance from zero for them. It also counts the zeros and non-finite floating-point numbers and reports them as underflows and overflows respectively. In fixed-point mode, it simply counts the number of underflows and overflows.

2.2.1 Constructors

The default constructor `fPoint::Statistics()` initializes the statistics.

2.2.2 Member Functions

`update(x)` updates the statistics using floating-point value `x` then returns a reference to them.

`samples()` returns the number finite non-zero samples.

`minimum()` returns the minimum exponent.

`average()` returns the average exponent.

`maximum()` returns the maximum exponent.

`variance()` returns the variance from zero.

`overflows()` returns a reference to the number of overflows.

`underflows()` returns a reference to the number of underflows.

2.2.3 Operators

`os << s` writes the statistics `s` to output stream `os` then returns a reference to `os`.

`is >> s` reads the statistics from input stream `is` and discards them then returns a reference to `is`.

2.3 `class fPoint::Result`

Programmers cannot combine fixed-point and floating-point operations in an expression accidentally because no operations are permitted on a variable of type `fPoint::Result` except assignment to a variable of type `fPoint`.

```
fPoint a, b, c, d, t;
a = b + c + d; // error!
a = (t = b + c) + d; // ok
a = fPoint(b + c) + d; // ok
```

2.3.1 Constructors

The default constructor `fPoint::Result(x)` initializes the underlying representation to the floating-point value of argument `x` or to some obviously incorrect floating-point value such as an IEEE NaN if argument `x` is omitted.

2.3.2 Member Functions

`floating()` returns the floating-point value of the underlying representation.

2.3.3 Operators

`operator =(x)` assigns the floating-point value of argument `x` to the underlying representation.

2.4 Mode Types

The `ModeType` enumeration

`fPoint::ModeType`	arithmetic
`fPoint::floatingPoint`	floating-point
`fPoint::fixedPoint`	fixed-point

2.5 Static Data Members

The `fPoint::Mode` After the `fPoint.h` header file has been included, the programmer must include one but not both of the following statements

```
const fPoint::ModeType fPoint::Mode = fPoint::floatingPoint;
const fPoint::ModeType fPoint::Mode = fPoint::fixedPoint;
```

in the global scope of the program.

The `fPoint::InitialSpecifier`

	`const`	
type	`fPoint::Specification`	initial value
data	`fPoint::InitialSpecifier`	`fPoint::Specification()`

is initialized with the initial default values.

The `fPoint::DefaultSpecifier`

	`const`	
type	`fPoint::Specification*`	initial value
data	`fPoint::DefaultSpecifier`	`&fPoint::InitialSpecifier`

2.6 Constructors

The explicit constructor `fPoint(s, x)` initializes a variable of type `fPoint` which references fixed-point specifier `s` and assigns the floating-point value `x` to the underlying representation. It references `*fPoint::DefaultSpecifier` if argument `s` is omitted and initializes the underlying representation or to some obviously incorrect floating-point value such as an IEEE NaN if argument `x` is omitted.

2.7 Member Functions

`floating()` returns the floating-point value of the underlying representation.

`specifier()` returns a constant reference to the fixed-point specifier.

`statistics()` returns a reference to the floating-point statistics.

`assign(x)` assigns the floating-point value `x` to the underlying representation then updates the statistics in floating-point mode or coerces the underlying representation in fixed-point mode and returns a reference to the variable.

2.8 Operators

The `fPoint` class library includes all of the usual floating-point operators.

`(const Specification&)x` returns a reference to the fixed-point specifier referenced by `x`.

`(Statistics&)x` returns a reference to the statistics through the fixed-point specifier referenced by `x`.

`x << n` returns the floating-point value of `x` multiplied by 2^{+n} where argument `n` is a positive integer.

`x >> n` returns the floating-point value of `x` multiplied by 2^{-n} where argument `n` is a positive integer.

`os << x` writes the floating-point value of `x` to output stream `os` then returns a reference to `os`.

`is >> x` reads a floating-point value from input stream `is` and assigns it to the underlying representation of `x` then returns a reference to `is`.

`x <<= n` assigns `x << n` to the underlying representation of `x` the returns a reference to `x`.

`x >>= n` assigns `x >> n` to the underlying representation of `x` the returns a reference to `x`.

2.9 Functions

The `fPoint` class library includes all of the usual floating-point functions.