Floating Number for Infinite Precision Calculation

Jiangnan Long

1. Introduction

For get more precision in some applications(validation results, precomputation lookup table, etc), We summarize a class FloatX<significand_bits, exponent_bits> and some functions in C++ . if increase significand_bits we can get more precision, and allow any increase, so named Infinite. We only discuss inspiration in here, that details of the algorithm are completely recorded in the source code.

2. Floating Number

2.1 Basic Operation are (+ - * /), implemented in FloatX::**do_plus()**, **do_multiply()**, **do_divide()**, **minus is plus a negative number**.

First we want to cover static-floating and dynamic-floating by function once. But this idea should not work, in dynamic-floating operation, two floating has different floating-constants and different bitwise-op. Therefore, we only write the static-floating opeartions in a class, It is the now FloatX::do_XXX(). other methods are meaningless.

What is the use of Dynamic Floating Number? example: 1.0/3. we cannot store irrational number with floating number, dynamic part is only the maximum significand bits. We not implement dynamic version of floating number, but we have dynamic version of rational number.

In basic operation, often use shift, we use shift by size_t. So the maximum of shift is (WIN32)4'294'967'295 (WIN64)18'446'744'073'709'551'615, the maximum of exponent bits is (WIN32)32[bits] (WIN64)64[bits]. Because exponent bits not need so large.

2.2 Factorization are factorize a floating number to significand*exp2(exponent), or to integerpart+fractionpart, implemented in FloatX::**do_modf()**, **do_frexp()**, **do_ldexp()**.

When we implement log(x), want function fast factorize floating numbers, and we often use TRUNC(we implemented simple do_trunc) and FRAC. So we understand the meaning of the standard.

2.3. Conversion, implement in FloatX::**do_cvtf()**, **do_cvtui()**, super large conversion is a problem ((source >> (1<<exponent_bits)) != 0).

3. Transcendental Functions

3.1 Trigonometric Function, direct calculation is absolute error (x - y) < eps, we want relative error (x - y)/y < eps. maximum problem occur to operation with infinite-number. We use BBP-formula to factorize the infinite number PI into three precision parts, so we can (((x - PI[1-14]) - PI[14-28]) - PI[28-42]). We introduce a multiplier into BBP-fomula realized fmod(x,PI). (why use 14 in double, see ZExample1.cpp, fmod(x,PI) is x%CircularConstant<double>())