Introduction

В этом проекте тебе предстоит реализовать библиотеку s21_decimal.h на языке программирования C. Эта библиотека должна добавить возможность работы с типом «decimal», который отсутствует в стандарте языка. Тем не менее, этот тип критически важен, например, для финансовых расчетов, где недопустимы погрешности вычислений, свойственные типам с плавающей точкой. В рамках этого проекта ты познакомишься с задачами обработки финансовой информации, погрузишься в вопросы внутреннего представления различных типов данных и закрепишь структурный подход.

Information

Тип Decimal представляет десятичные числа в диапазоне от положительных 79,228,162,514,264,337,593,543,950,335 до отрицательных 79,228,162,514,264,337,593,543,950,335. Значение Decimal по умолчанию равно 0. Decimal подходит для финансовых расчетов, которые требуют большого количества значимых целых и дробных цифр и отсутствия ошибок округления. Этот тип не устраняет необходимость округления. Скорее, сводит к минимуму количество ошибок из-за него.

Когда результат деления и умножения передается методу округления, результат не страдает от потери точности.

Decimal число — это значение с плавающей точкой, состоящее из знака, числового значения, где каждая цифра находится в диапазоне от 0 до 9, и коэффициента масштабирования, который указывает положение десятичной точки, разделяющей целые и дробные части числового значения.

Двоичное представление Decimal состоит из 1-разрядного знака, 96-разрядного целого числа и коэффициента масштабирования, используемого для деления 96-разрядного целого числа и указания того, какая его часть является десятичной дробью. Коэффициент масштабирования неявно равен числу 10, возведенному в степень в диапазоне от 0 до 28. Следовательно, двоичное представление Decimal имеет вид ((от -2^96 до 2^96) / 10^(от 0 до 28)), где -(2^96-1) равно минимальному значению, а 2^96-1 равно максимальному значению.

Коэффициент масштабирования также может сохранять любые конечные нули в Decimal. Эти конечные нули не влияют на значение в арифметических операциях или операциях сравнения.

Двоичное представление

Двоичное представление Decimal состоит из 1-разрядного знака, 96-разрядного целого числа и коэффициента масштабирования, используемого для деления целого числа и указания того, какая его часть является десятичной дробью. Коэффициент масштабирования неявно равен числу 10, возведенному в степень в диапазоне от 0 до 28.

Decimal число может быть реализовано в виде четырехэлементного массива 32-разрядных целых чисел со знаком (`int bits[4];`).

`bits[0]` , `bits[1]` , и `bits[2]` содержат младшие, средние и старшие 32 бита 96-разрядного целого числа соответственно.

`bits[3]` содержит коэффициент масштабирования и знак и состоит из следующих частей:

- Биты от 0 до 15, младшее слово, не используются и должны быть равны нулю.
- Биты с 16 по 23 должны содержать показатель степени от 0 до 28, который указывает степень 10 для разделения целого числа.
- Биты с 24 по 30 не используются и должны быть равны нулю.
- Бит 31 содержит знак; 0 означает положительный, а 1 означает отрицательный.

Обрати внимание, что битовое представление различает отрицательные и положительные нули. Эти значения могут считаться эквивалентными во всех операциях.

Пример:

```
typedef struct
{
    int bits[4];
} s21_decimal;
```

Арифметические операторы

Название оператора	Оператор	Функция
Сложение	+	int s21_add(s21_decimal value_1, s21_decimal value_2, s21_decimal *result)
Вычитание	-	int s21_sub(s21_decimal value_1, s21_decimal value_2, s21_decimal *result)
Умножение	*	int s21_mul(s21_decimal value_1, s21_decimal value_2, s21_decimal *result)
Деление	/	int s21_div(s21_decimal value_1, s21_decimal value_2, s21_decimal *result)

Функции возвращают код ошибки:

- 0 — OK;
- 1 — число слишком велико или равно бесконечности;
- 2 — число слишком мало или равно отрицательной бесконечности;
- 3 — деление на 0.

Уточнение про числа, не вмещающиеся в мантиссу:

- *При получении чисел, не вмещающихся в мантиссу при арифметических операциях, используй банковское округление (например, 79,228,162,514,264,337,593,543,950,335 — 0.6 = 79,228,162,514,264,337,593,543,950,334).*

Операторы сравнения

Название оператора	Оператор	Функция
Меньше	<	int s21_is_less(s21_decimal, s21_decimal)
Меньше или равно	<=	int s21_is_less_or_equal(s21_decimal, s21_decimal)
Больше	>	int s21_is_greater(s21_decimal, s21_decimal)
Больше или равно	>=	int s21_is_greater_or_equal(s21_decimal, s21_decimal)
Равно	==	int s21_is_equal(s21_decimal, s21_decimal)
Не равно	!=	int s21_is_not_equal(s21_decimal, s21_decimal)

Возвращаемое значение:

- 0 — FALSE;
- 1 — TRUE.

Преобразователи

Преобразователь	Функция
Из int	int s21_from_int_to_decimal(int src, s21_decimal *dst)
Из float	int s21_from_float_to_decimal(float src, s21_decimal *dst)
В int	int s21_from_decimal_to_int(s21_decimal src, int *dst)
В float	int s21_from_decimal_to_float(s21_decimal src, float *dst)

Возвращаемое значение — код ошибки:

- 0 — OK;
- 1 — ошибка конвертации.

Уточнение про преобразование числа типа float:

- *Если числа слишком малы (0 < |x| < 1e-28), возвращай ошибку и значение, равное 0.*
- *Если числа слишком велики (|x| > 79,228,162,514,264,337,593,543,950,335) или равны бесконечности, возвращай ошибку.*
- *При обработке числа с типом float преобразовывай все содержащиеся в нём значимые десятичные цифры. Если таких цифр больше 7, то значение числа должно округляться к ближайшему, у которого не больше 7 значимых цифр.*

Уточнение про преобразование из числа типа decimal в тип int:

- *Если в числе типа decimal есть дробная часть, то её следует отбросить (например, 0.9 преобразуется 0).*

Другие функции

Описание	Функция
Округляет указанное Decimal число до ближайшего целого числа в сторону отрицательной бесконечности.	int s21_floor(s21_decimal value, s21_decimal *result)
Округляет Decimal до ближайшего целого числа.	int s21_round(s21_decimal value, s21_decimal *result)
Возвращает целые цифры указанного Decimal числа; любые дробные цифры отбрасываются, включая конечные нули.	int s21_truncate(s21_decimal value, s21_decimal *result)
Возвращает результат умножения указанного Decimal на -1.	int s21_negate(s21_decimal value, s21_decimal *result)

Возвращаемое значение — код ошибки:

- 0 — OK;
- 1 — ошибка вычисления.

Part 1. Реализация функции библиотеки decimal.h

Тебе необходимо реализовать описанные выше функции библиотеки:

- Библиотека должна быть разработана на языке C стандарта C11 с использованием компилятора gcc.
- Код библиотеки должен находиться в папке src в ветке develop.
- Не используй устаревшие и выведенные из употребления конструкции языка и библиотечные функции. Обрати внимание на пометки legacy и obsolete в официальной документации по языку и используемым библиотекам. Ориентируйся на стандарт POSIX.1-2017.
- При написании кода необходимо придерживаться Google Style.
- Оформи решение как статическую библиотеку с названием *s21_decimal.a* (с заголовочным файлом s21_decimal.h).
- Библиотека должна быть разработана в соответствии с принципами структурного программирования.
- Перед каждой функцией используй префикс s21_.
- Подготовь полное покрытие unit-тестами функций библиотеки с помощью библиотеки Check.
- Unit-тесты должны покрывать не менее 80% каждой функции.

- Предусмотри Makefile для сборки библиотеки и тестов (с целями all, clean, test, s21_decimal.a, gcov_report).
- В цели gcov_report должен формироваться отчёт gcov в виде html-страницы. Для этого unit-тесты должны запускаться с флагами gcov.
- При реализации decimal ориентируйся на двоичное представление с целочисленным массивом `bits` , как указано в примере выше. Соблюдай положение разрядов числа в массиве `bits` .
- Запрещено использование типа __int128.
- Конечные нули можно как оставлять, так и удалять (за исключением функции `s21_truncate`).
- Определяемый тип должен поддерживать числа от -79,228,162,514,264,337,593,543,950,335 до +79,228,162,514,264,337,593,543,950,335.