Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1 ПО ДИСЦИПЛИНЕ ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент **Курачева Ксения Дмитриевна**

Группа **ИУ7-33Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Курачева К. Д.**

Преподаватель _____ **Барышникова М. Ю.**

Оценка _____

2023 г.

Условие задачи

Составить программу деления большого вещественного числа на большое целое. Большое вещественное число – число, порядок которого состоит не более, чем из 5 цифр, а мантисса – до 35 цифр включительно. Большое целое число – число, которое состоит не более, чем из 35 цифр.

Техническое задание

Смоделировать операцию деления действительного числа в форме ±m.nE±K, где суммарная длина мантиссы (m+n) – до 35 значащих цифр, а величина порядка K – до 5 цифр, на целое число длиной до 35 десятичных цифр. Результат выдать в форме ±0.m1E±K1, где m1 - до 35 значащих цифр, а K1 - до 5 цифр. Если при вычислении результата был переполнен порядок, программа выводит сообщение о невозможности провести счёт.

Способ обращения к программе:

В командной строке перейти в папку, в которой находится программа и вызвать команду “./app.exe”.

Входные данные:

1. Строка, в которой записано число в любой форме (целое, действительное, действительное в экспоненциальной форме). Количество цифр в мантиссе — до 35 включительно, количество цифр в порядке — до 5 включительно. Вводить «+» не обязательно ни в случае самого числа, ни в случае порядка. Регистр «e» не важен. Точку ставить не обязательно.

2. Строка, в которой записано целое число, содержащее до 35 цифр включительно. Если число положительное, вводить „+“ не обязательно.

Выходные данные:

Строка, в которой записан результат деления числа, содержащегося в первой входной строке, на число, содержащееся во второй входной строке, при условии их корректности. Выводится в форме ±0.m1E±K1, где m1 - до 35 значащих цифр, а K1 - до 5 цифр.

Возможные аварийные ситуации:

- Ввод данных в неверном формате.

- Попытка деления на 0.

В обоих вышеперечисленных случаях программа выведет описание ошибки и предоставит пользователю возможность повторного ввода.

- Переполнение порядка результата.

Структуры данных

Максимальные ограничения:

#define NDIGITS_IN_FRACTION 35

Максимальное количество цифр в мантиссе большого вещественного числа.

#define NDIGITS_IN_BINT 35

Максимальное количество цифр в целом числе.

#define NDIGITS_IN_EXP 5

Максимальное количество цифр в порядке большого вещественного числа.

Коды ошибок:

#define TOO_BIG_FRACT_ERR -10

Возвращается функцией str_to_bdouble в случае, если в числе оказалось больше, чем 35 цифр в мантиссе.

#define TOO_BIG_EXP_ERR -11

Возвращается функцией str_to_bdouble в случае, если в числе оказалось больше, чем 5 цифр в порядке.

#define TOO_BIG_INT_ERR -12

Возвращается функцией str_to_bint в случае, если в числе оказалось больше, чем 35 цифр.

#define OWERFLOW_ERROR -13

Возвращается функцией div_bdouble_by_bint в случае, если в резуьтате вычисления частного произошло переполнение порядка в большую сторону.

Структура большого вещественного числа:

```
struct my_big_double {
    char fraction_sign;
    int fraction_len;
    char fraction[NDIGITS_IN_FRACTION];
    long exponent;
};
```

Структура имеет четыре поля. Первое содержит знак хранимого числа и является -1, если это число отрицательно, или +1 в обратном случае. Тип char

был выбран, как минимально возможный целый тип данных. Второе поле отвечает за количество элементов мантиссы, записанных в массив цифр fraction. Количество элементов мантиссы записывается в тип int, поскольку он вмещает максимальную длину мантиссы — 35 символов. Третье поле структуры — массив цифр мантиссы fraction. Тип char был выбран, как минимально возможный целый тип данных, поскольку каждый элемент массива — цифра. Последнее поле хранит величину порядка. Тип long был выбран, поскольку это минимальный тип, гарантированно вмещающий в себя максимально возможные порядки, -99999 и 99999, в соответствии со стандартом языка C99.

В данную структуру данных записывается первое входное число и результат вычислений.

Структура большого целого числа:

```
struct my_big_int {
    char sign;
    int digits_len;
    char digits[NDIGITS_IN_BINT];
};
```

В данной структуре три поля:

— знак числа, тип которого, char, был выбран как минимально возможный для хранения двух значений — -1 и +1;

— количество цифр числа, тип которого, int, был выбран, поскольку он вмещает максимальную длину числа — 35 символов.

— массив цифр числа, тип которого, char, был выбран, как минимально возможный целый тип данных, поскольку каждый элемент массива — цифра.

В данную структуру данных записывается второе входное число.

int str_to_bdouble(char *number, struct my_big_double *n);

Функция выполняет считывание вещественного числа в структуру struct my_big_double *n в любой форме, со знаком или без него, с мантиссой до 35 цифр включительно и порядком до 5 цифр включительно, и возвращает 0 в случае корректной работы и код ошибки в противном случае.

int str_to_bint(char *number, struct my_big_int *n);

Функция выполняет считывание целого числа в структуру struct my_big_int *n в любой форме с количеством цифр до 35 включительно, со знаком или

без него, и возвращает 0 в случае корректной работы или ненулевой код ошибки в противном случае.

int div_bdouble_by_bint(struct my_big_double *dividend, struct my_big_int *divisor, struct my_big_double *quotient);

Функция выполняет деление большого вещественного числа на большое целое, записывает результат в *quotient и возвращает 0 в случае корректного вычисления или ненулевой код ошибки в обратном случае.

void print_bdouble(struct my_big_double *n);

Функция выводит содержимое структуры my_big_double на экран в нормальной форме.

void print_bint(struct my_big_int *n);

Функция выводит содержимое структуры my_big_int на экран.

int is_bint_zero(struct my_big_int *bint);

Функция проверяет равенство большого целого числа нулю и возвращает 1 в случае положительного результат или 0 в обратном случае.

Описание алгоритма

В программе использован классический алгоритм деления “столбиком”.

1) Определяется длина неполного делимого. Для этого из мантиссы делимого выделяется число, состоящее из первых n её цифр, где n — количество цифр делителя. Делитель и выделенное неполное делимое сравниваются. Если второе число меньше, его длина увеличивается на единицу и в конец числа дописывается (n + 1)я цифра делимого (случай 1), в противном случае выделенное неполное делимое остаётся прежним (случай 2).

2) Далее вычисляется частное неполного делимого и делителя:

- Случай 1. Длина неполного делимого больше длины делителя на единицу.

Первая цифра неполного делимого умножается на 10, к ней прибавляется вторая цифра неполного делимого, и результат делится на первую цифру делителя.

- Случай 2. Длина неполного делимого равна длине делителя.

Первая цифра неполного делимого делится на первую цифру делителя.

3) Частное неполного делимого и делителя умножается на делитель. Если результат больше делителя, частное уменьшается на единицу, опять умножается на делитель и результат сравнивается с делителем. Когда

результат станет меньше или равен делителю, находится разность неполного делимого и полученного результата умножения.

4) К полученному в предыдущем пункте числу в конец сносится следующая цифра делимого. Если её нет, ставится 0, а порядок уменьшается на единицу. Действия с п. 2) повторяются до тех пор, пока неполное делимое и остаток не будут равны 0 или пока количество значащих цифр не будет равным длине результата + 1.

5) Если длина ответа равна максимальной длине и было вычислено ещё одно значение следующей цифры, результат округляется в большую сторону, если эта цифра больше или равна 5, или остаётся прежним в противном случае.

6) Результат приводится к нормальному виду.

7) Экспонента проверяется на переполнение. Если она переполнена и положительна, возвращается ошибка, если она переполнена и отрицательна, в результат записывается значение машинного нуля.

Тесты

Негативные тесты.

Test_01: Введено не вещественное число.

asas

> Incorrect dividend.

Not a big real number.

Test_02: Введённое вещественное число содержит некорректный символ в порядке.

Enter dividend, big real number: 900000000000000000000000002.12-12

> Incorrect dividend.

Not a big real number.

Test_03: Введённое вещественное число содержит некорректный символ в мантиссе.

Enter dividend, big real number: -11-1-

> Incorrect dividend.

Not a big real number.

Test_04: Вместо ввода делимого введён Enter.

Enter dividend, big real number:

> Incorrect dividend.

Not a big real number.

Test_05: Вместо ввода делителя числа введён Enter.

Enter divisor, big integer:

> Incorrect divisor.

Test_06: Вместо ввода целого числа введено вещественное.

Enter divisor, big integer: 12.3e12

> Incorrect divisor.

Test_07: Слишком большая мантисса вещественного числа.

Enter dividend, big real number: 1111111111111111111111111111111111e12

> Incorrect dividend.

Factorial is too big. Number of digits in it must be less than 35

Test_08: Слишком большой порядок вещественного числа.

Enter dividend, big real number: 111e111111

> Incorrect dividend.

Exponent is too big. Number of digits in it must be less than 5

Test_09: Целое число слишком большое.

Enter divisor, big integer: 12222222222222222222222222222222222

> Incorrect divisor.

Number of digits is too big. Should be less than 35

Test_10: Деление на 0.

Enter divisor, big integer: 0

> Incorrect divisor.

> It is impossible to divide by 0.

Test_11: В результате деления переполнился порядок в большую сторону.

Enter dividend, big real number: 1111111111111111111111111111111111e99967

Enter divisor, big integer: 1

Overflow error.

Test_11: В результате деления переполнился порядок в меньшую сторону.

Enter dividend, big real number: 0.11111e-99999

Enter divisor, big integer: 10

Overflow error.

Позитивные тесты

Тестирование правильности деления

Test_01: Число, состоящее из 35 цифр в мантиссе делится на целое число, состоящее из 35 цифр.

Enter dividend, big real number: 900000000000000000000000000000009

Enter divisor, big integer: 100000000000000000000000000000001

Quotient: +0.9E+1

Test_02: Число, состоящее из 35 цифр в мантиссе и 5 в порядке, делится на целое число, состоящее из 35 цифр.

Enter dividend, big real number: 99999999999999999999999999999999e10000

Enter divisor, big integer: 999999999999999999999999999999999

Quotient: +0.1E+10034

Enter dividend, big real number: 11111111111111111111111111111111e10000

Enter divisor, big integer: 111111111111111111111111111111111

Quotient: +0.1E+10001

Test_03: Деление чисел до 100 на числа до 100.

Enter dividend, big real number: 100

Enter divisor, big integer: 100

Quotient: +0.1E+1

Enter dividend, big real number: 81

Enter divisor, big integer: 9

Quotient: +0.9E+1

Enter dividend, big real number: 1

Enter divisor, big integer: 100

Quotient: +0.1E-1

Test_04: Число, состоящее из 35 цифр в мантиссе и имеющее отрицательный порядок из 5 цифр, делится на целое число, состоящее из 35 цифр.

Enter dividend, big real number: 11111111111111111111111111111111e-11111

Enter divisor, big integer: 11111111111111111111111111111111

Quotient: +0.1E-11110

Тестирование определения знака

Enter dividend, big real number: -1

Enter divisor, big integer: +1

Quotient: -0.1E+1

Enter dividend, big real number: +1

Enter divisor, big integer: +1

Quotient: +0.1E+1

Enter dividend, big real number: -1

Enter divisor, big integer: -1

Quotient: +0.1E+1

Enter dividend, big real number: +1

Enter divisor, big integer: -1

Quotient: -0.1E+1

Тестирование округления результата

Enter dividend, big real number: 99999999999999999999999999999999

Enter divisor, big integer: 2

Quotient: +0.5E+36

Enter dividend, big real number: 2

Enter divisor, big integer: 3

Quotient: +0.66666666666666666666666666666667E+1

Выводы по проделанной работе

Для реализации деления чисел, выходящих за разрядную сетку ПК, можно использовать алгоритм “столбиком”. Он нагляден и прост в реализации. Для хранения и обработки чисел, выходящих за разрядную сетку ПК, удобно

использовать структуру с полями, содержащими данное число “по частям”: знак (хранится в типе char), мантисса (хранится в массиве типа char) и порядок (хранится в стандартном типе long).

Контрольные вопросы

1) Каков возможный диапазон чисел, представляемых в ПК?

Диапазон чисел зависит от разрядности системы.

Целые беззнаковые числа: 0 < x <= 2^n – 1, где n — количество бит, выделенных под переменную.

Целые знаковые числа: -2^(n-1) < x < +2^(n-1), где n — количество бит, выделенных под переменную.

Действительные числа, абсолютное ограничение по модулю: 3.6E–4951 <= x <= 1.1E+4932 (максимальный размер мантиссы двоичных 52 разряда, порядок – 11 разрядов).

2) Какова возможная точность представления чисел, чем она определяется?

Точность представления вещественного числа зависит от максимально возможной длины мантиссы, которая, в свою очередь, зависит от области выделяемой памяти и наличия знака. Если длина мантиссы выходит за границы разрядной сетки, то происходит округление. Длину мантиссы у целых чисел проще всего посчитать, выведя максимальное значение данного числа, но можно взять в качестве примера тип long long языка Си – 19 разрядов (или 20, если тип беззнаковый).

3) Какие стандартные операции возможны над числами?

Для каждого числа должны быть определены арифметические операции (сложение, вычитание, деление, умножение) и операции сравнения.

4) Какой тип данных может выбрать программист, если обрабатываемые числа превышают возможный диапазон представления чисел в ПК?

Программист может написать свою структуру, однако в её основе удобнее использовать массив. Тип элементов массива зависит от реализуемой идеи — если действия производятся по одной цифре, то каждую цифру удобно хранить поэлементно, а тип использовать char, как минимально возможный, способный хранить от 0 до 9. Если действия производятся над группой цифр, удобно хранить по группе цифр в каждом элементе. Тип такого массива зависит от количества цифр в группе.

5) Как можно осуществить операции над числами, выходящими за рамки машинного представления?

Сохранить число в массив, в котором каждый элемент — одна или несколько цифр, и производить действия поэлементно (над каждой цифрой или группой цифр), например, используя классический алгоритм “столбиком”.