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

Обработка больших чисел

Студент **Абдуллаев Ш. В.**

Группа **ИУ7-34Б**

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

Студент _____ **Абдуллаев Ш. В.**

Преподаватель _____ **Силантьева А. В.**

2024

Описание условия задачи

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.mE±K1, где m1 - до 35 значащих цифр, а K1 - до 5 цифр.

Описание исходных данных

На вход программе подаются 2 строки. Они могут начинаться с пробелов или ведущих нулей, которые не учитываются в подсчёте длины мантиссы, но учитываются в максимальной возможной длине вводимой строки. На вход программы ожидается следующий формат данных:

1. Сначала вводится действительное число в форме [±]m.n[Ee][±]K, где суммарная длина мантиссы (m+n) — до 35 значащих цифр, а величина порядка K - до 5 цифр. Число может представляться без точки и без знака (если знак отсутствует, число считается положительным), также можно опустить порядок числа и его знак.

2. Затем вводится целое число длиной до 35 десятичных цифр. Первый символ может быть знаком числа (+ или -), если знак отсутствует, число считается положительным.

Наличие в строке побочных символов (кроме пробельных символов в начале) считается некорректным вводом. Пример случаев допустимого и недопустимого ввода представлен на таблице 1.

Корректный ввод	Некорректный ввод
Действительное число	
123.123	+
99e-2	+-.
-3E-190	.
.00025	E

+123001.	e
240.e1	E23
–123.456	24ads1
1234567e–20	23e23.1
1234567e20	23E100000
123.4567e23	12e-100000
+001200.0100e-0090	12e12.2
Целое число	
+5	++2
-20	2.2
209	2qwe
001	+-33
0000	2e2

Таблица 1. Корректность вводимых данных

Описание результатов

Результат программы выводится в нормализованном виде: [±]0.Xe[±]N. Длина мантиссы числа X не более 35, при этом первая цифра X ненулевая. Длина порядка N не более 5 цифр.

Описание задачи, реализуемой программой

Программа выполняет деление действительного числа, записанного в формате [±]m.n[Ee][±]K, на целое число длиной до 35 десятичных цифр и выводит результат деления в нормализованном виде [±]0.Xe[±]N.

Способ обращения к программе

Обращения к программе пользователем происходит с помощью вызова исполняемого файла (app.exe).

Описание возможных аварийных ситуаций и ошибок пользователя

1. Ошибка ввода целого числа. Программа вернет READ_INT_ERROR.

2. Ошибка переполнения целого числа. Программа вернет OVERFLOW_INT_ERROR.
3. Введена пустая строка вместо целого числа. Программа вернет EMPTY_INT_ERROR.
4. Введено не целое число. Программа вернет INPUT_INT_ERROR.
5. Ошибка ввода вещественного числа. Программа вернет READ_REAL_ERROR.
6. Ошибка переполнения вещественного числа. Программа вернет OVERFLOW_REAL_ERROR.
7. Введена пустая строка вместо вещественного числа. Программа вернет EMPTY_REAL_ERROR.
8. Введено не вещественное число. Программа вернет INPUT_REAL_ERROR.
9. При привидении к нормальному виду произошло достижение машинного нуля или машинной бесконечности (переполнение порядка). Программа вернет OVERFLOW_REAL_ORDER_ERROR.
10. Деление на нуль при делении. Программа вернет DIVISION_BY_ZERO_ERROR.
11. Достижение машинного нуля или машинной бесконечности (переполнение порядка при делении. Программа вернет OVERFLOW_ORDER_ERROR.

Описание внутренних структур данных

В программе реализована структура нормального числа, которая представлена в Листинге 1.

```
#define MANTISSA_LEN 35
#define LIMIT_ORDER 99999

typedef struct
{
   int mantissa_sign;
   int mantissa[MANTISSA_LEN];
```

```
    size_t mantissa_len;
    int order;
} normal_t;
```

Листинг 1. Структура нормального числа

Рассмотрим каждое поле структуры:

- mantissa_sign: знак мантиссы, который может быть положительным (0) или отрицательным (1). Это используется для определения знака числа с плавающей запятой.
- mantissa: массив, содержащий цифры мантиссы числа с плавающей запятой. Длина массива задается константой MANTISSA_LEN.
- mantissa_len: размер мантиссы, который указывает количество элементов в массиве mantissa.
- order: порядок числа. Это целочисленное значение, которое определяет положение десятичной точки относительно мантиссы.

Таким образом, структурой normal_t обеспечивается хранение вещественного числа с разделением его знака, мантиссы и порядка.

Описание алгоритма

1. Сначала вводится действительное число, которое при успешном прохождении проверок переводится в нормализованный вид и записывается в структуру normal_t.
2. Затем вводится целое число, которое при успешном прохождении проверок переводится в нормализованный вид и записывается в структуру normal_t.
3. Проводятся проверки:
    a. Если делитель равен нулю, возвращается ошибка деления на нуль.
    b. Если разница порядков чисел слишком велика, возвращается ошибка переполнения.
    c. Если делимое равно нулю, результат — нуль.

4. Вычисляются знак итогового числа и его порядок. Если после пересчета порядок превышает допустимые пределы, снова возвращается ошибка переполнения.
5. Чтобы определить неполное делимое, сравниваются старшие разряды делителя и делимого. Если старший разряд делимого больше, длина неполного делимого равна длине делителя. Если меньше — длина неполного делимого на 1 больше. Если разряды одинаковы, продолжается сравнение следующих разрядов.
6. Создается массив для хранения временного неполного делимого, который копируется из мантиссы делимого. Выполняется итеративное деление с помощью функции quotient, которая вычисляет частное от деления текущей части делимого на делитель и обновляет остаток.
7. Частные циклично записываются в мантиссу результата. Постепенно к текущему делимому добавляются новые цифры из мантиссы делимого, и производится повторное деление. Если текущий остаток после деления равен нулю, процесс добавления нулей в результат продолжается. Алгоритм вычисляет новые цифры результата до тех пор, пока не будет обработана вся мантисса делимого или не будет достигнута максимальная длина мантиссы результата.
8. Если мантисса результата достигает максимальной длины, происходит округление. Если следующая цифра частного после максимальной длины больше или равна 5, мантисса округляется в большую сторону.
9. В конце функция корректирует длину мантиссы в результатах, удаляя ведущие нули.
10. Результат деления выводится в нормализованном виде пользователю.

Набор тестов

Ввод		Вывод	Комментарий
2	2	+0.1e1	Делимое и делитель совпадают

2e2	2	+0.1e3	Делимое с экспонентой
0000444	4	+0.111e3	Делимое с ведущими нулями
121	0011	+0.11e2	Делитель с ведущими нулями
321	1	+0.321e3	Деление на 1
999e0	9	+0.111e3	Нулевая экспонента
-24e1	2	-0.12e3	Делимое кратно делителю, разные знаки
100e2	10	+0.1e4	Степени 10
2.	3	+0.66…67e0	Результат с округлением
1	3	+0.33…33e0	Результат без округления
99...9E+99999	99...9E+99999	+0.1e1	Максимальные по длине мантиссы и порядка числа
2	99…99	+0.2e-34	Округление с удалением нулей
0.0	2	+0.0e0	Деление нуля на число
5	99…9	+0.500…01e-34	Округление последней цифры

Таблица 2. Набор позитивных тестов

Ввод		Комментарий
100	0	Деление на нуль
0.0	0	Деление нуль на нуль
.	2	Делимое является точкой
2q	2	В делимом нечисловые символы
2	32w	В делителе нечисловые символы
123e123.2	2	Делимое не вещественное число
99	3.3	Делитель не целое число
	2	Делимое введено пустой строкой
2		Делитель введён пустой строкой

23e100000	3	Переполнение порядка у делимого
2E+99999	1	Переполнение порядка в ответе
99…99	9	Превышение длины мантиссы у делимого
3e2.2	4	Нецелая экспонента

Таблица 2. Набор негативных тестов

Выводы

Для работы с числами, выходящими за пределы стандартных разрядных ограничений, можно создать тип данных, поддерживающий произвольное количество цифр. Арифметические операции для этого типа данных могут быть реализованы поразрядно, что позволит обрабатывать числа любой длины, обеспечивая точность и корректность вычислений.

Контрольные вопросы

1. Каков возможный диапазон чисел, представляемых в ПК?

Диапазон чисел, представляемых в ПК, зависит от разрядности системы и типа данных. Например, для целых знаковых чисел (тип int) в 32-разрядной системе диапазон составляет от -2^31 (−2 147 483 648) до 2^31-1 (2 147 483 647), а в 64-разрядной системе — от -2^63 (−9,223372×10^18) до 2^63-1 (9,223372×10^18).

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

Точность представления чисел зависит от длины их мантиссы. Существуют числа одинарной точности (float), точность мантиссы которых 23 разряда (8 388 608), и числа двойной точности (double) с точностью в 52 (4 503 599 627 370 496) двоичных разряда.

3. Какие стандартные операции возможны над числами?

Стандартными операциями, которые можно выполнять над числами, являются: сложение, вычитание, умножение, деление, равенство и сравнение.

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

Если обрабатываемые числа превышают возможный диапазон представления чисел в ПК, программист может выбрать тип данных, позволяющий работать с числами большего диапазона и более высокой точностью. В качестве альтернативного варианта программист может реализовать собственный тип данных для представления длинных чисел.

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

Операции с числами, превышающими границы машинного представления, можно выполнять, разделяя числа на более мелкие блоки и обрабатывая каждый блок по отдельности. Для работы с большими числами часто применяют специализированные библиотеки, которые поддерживают числа произвольной длины и выполняют операции, не ограниченные размером машинного слова. В таких библиотеках числа обычно представляются как массивы или списки, где каждая ячейка хранит часть числа, а операции выполняются поразрядно. Это позволяет работать с числами любой разрядности. Программист также может создать свой собственный тип данных для представления больших чисел и реализовать необходимые арифметические операции через написание соответствующих функций для работы с ними.