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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

О Т Ч Е Т

По лабораторной работе № 1

Название: Обработка больших чисел

Дисциплина: Типы и структуры данных

Студент ИУ7-32Б (Группа) С.С. Беляк (И.О. Фамилия)

Преподаватель Барышникова М.Ю.

Москва, 2023

Описание условия задачи

Смоделировать операцию умножения действительного числа на действительное число в форме ±m.n E ±K, где суммарная длина мантиссы первого сомножителя (m+n) - до 35 значащих цифр, второго – до 40 значащих цифр, а величина порядка K - до 5 цифр. Результат выдать в форме ±0.m1 E ±K1, где m1 – до 40 значащих цифр, а K1 - до 5 цифр.

Описание ТЗ

1. Описание исходных данных

Данные передаются на вход при помощи чтения функцией fgets с типом char.
На вход программы ожидается следующий формат данных:

1. Первая строка содержит первое действительное число в формате +/-X, затем порядок (E) в формате +/-K.

- Первый символ может быть знаком числа (+ или -), если знак отсутствует, число считается положительным.
- X - цифры в мантиссе числа.
- Длина мантиссы не должна превышать 35 цифр.
- K - цифры в порядке числа.
- Длина порядка не должна превышать 5 цифр.

2. Вторая строка содержит второе действительное число в формате +/-X, затем порядок (E) в формате +/-K.

- Первый символ может быть знаком числа (+ или -), если знак отсутствует, число считается положительным.
- X - цифры в мантиссе числа.
- Длина мантиссы не должна превышать 40 цифр.
- K - цифры в порядке числа.
- Длина порядка не должна превышать 5 цифр.

Имеются следующие ограничения:

- Суммарная длина мантиссы первого сомножителя (m+n) - до 35 значащих цифр,
- Суммарная длина мантиссы второго сомножителя (m+n) - до 40 значащих цифр,
- Длина порядка не должна превышать 5 цифр,
- Порядок имеет до 5 знаков: от –99999 до +99999.

Для корректной работы программы необходимо проверить, что во входных строках нет некорректных символов, которые нельзя преобразовать в вещественное число. Возможен ввод знака перед числом. Первый символ может быть знаком числа. В случае положительного числа знак опционален. То же относится и к знаку порядка. Ввод нескольких знаков + или - в мантиссе или порядке считается некорректным.
Ввод вещественного числа можно осуществлять в нормализованном виде. Ведущие нули в расчете длины числа не учитываются. Значащие нули после точки учитываются при подсчете длины числа. Десятичное число может представляться без точки: 123 (как целое). При наличии точки в числе возможны следующие варианты его представления: .00025, +123001., –123.456. Также допускается представление числа в экспоненциальной форме: 1234567E–20, 1234567E20 или 123.4567E23.
Для корректной работы программы необходимо обеспечить, чтобы во входных строках отсутствовали некорректные символы, которые нельзя преобразовать в вещественное число. Нечисловые символы, за исключением точки, плюса и минуса, считаются некорректными и не должны присутствовать во входных данных.

Таблица примеров ввода:

Допустимый ввод	**Недопустимый ввод**
111	+-111
+111	+111+E+11
-111	-11-1
11.1	11E
-11.1	11E11E11
+1.1E+1	11.11.11
11E1	00++11
+0011	00--11
-0011	000000000000000000000000000111111111
0011	abc

2. Описание результатов программы

Результат выполнения программы умножения будет представлен в виде десятичного числа с плавающей точкой, записанного в формате: +/-XX.XX**E**+/-K.
Где X - цифры мантиссы, K - цифры порядка, а E - обозначение порядка. где X– до 40 значащих цифр, а K1 - до 5 цифр.

3. Описание задачи, реализуемой в программе

Цель работы: реализация арифметической операции умножения действительного числа на действительное число в форме ±m.n E ±K, выходящими за разрядную сетку персонального компьютера, реализация вывода на экран операции умножения.

4. Способ обращения к программе

Способ обращения к программе пользователем происходит через исполняемый файл app.exe.

5. Описание возможных аварийных ситуаций и ошибок пользователя

1.Некорректный формат числа: если пользователь передаст число в неправильном формате, например, с буквами вместо чисел, то функция вернет ошибку ERR_RANGE.
2.Неправильный порядок числа: если порядок, заданный пользователем, превышает максимально допустимый размер, то функция вернет ошибку ERR_MANTISS_SIZE.
3.Неправильная мантисса числа: если размер мантиссы числа превышает максимально допустимый размер, функция также вернет ошибку ERR_MANTISS_SIZE.
4.Некорректный знак числа: если пользователь передаст знак числа, отличный от "+" или "-", то в мантиссе числа будет установлен некорректный знак.
5.Превышение допустимого размера переменных: если размерности мантиссы и порядка числа превысят максимально допустимые значения, то функция вернет ошибку ERR_MANTISS_SIZE и ERR_RANGE.
6.Исключительные ситуации при конвертировании чисел: при использовании функции atoi для конвертации символов в числовые значения могут произойти исключительные ситуации, например, переполнение int или передача невалидных символов.
7.Другие ошибки и исключительные ситуации, связанные с операционной системой или другими программами, которые пользователь мог вызывать до запуска данной программы.

6.Описание внутренних структур данных

Программа содержит в себе структуру, представляющую вещественное число my_float

Листинг структуры

```
struct my_float
{
    int mantiss[MAX_MANTISS_SIZE];
    int size;
    int order;
    int mant_sign;
};
```

- mantiss - массив целых чисел, которые представляют мантиссу числа. Массив имеет максимальный размер, то есть может содержать до MAX_MANTISS_SIZE элементов.

- size - целое число, представляющее размер мантиссы. Оно указывает на количество элементов массива, которые являются значащими.

- order - целое число, представляющее порядок числа. Оно указывает на местоположение десятичной запятой в числе. Число может быть положительным или отрицательным.

- mant_sign - целое число, представляющее знак мантиссы числа. Значение может быть положительным (1) или 0.

Таким образом, структура my_float позволяет хранить вещественное число, разделяя его на мантиссу, порядок и знак мантиссы.

7. Алгоритм программы

Общий алгоритм программы:

1. Начинается выполнение программы в функции main. Пользователю выводится информация о программе и ограничениях на формат вводимых чисел для умножения.
2. Программа просит ввести первое число, используя функцию printf.
3. Пользователь вводит первое число, используя функцию fgets.
4. Проверяется корректность длины вводимого числа. Если длина не соответствует описанию, выводится сообщение об ошибке и программа завершается.
5. Проверяется корректность ввода первого числа. Если ввод не соответствует описанию, выводится сообщение об ошибке и программа завершается.
6. Программа просит ввести второе число, используя функцию printf.
7. Пользователь вводит второе число, используя функцию fgets.
8. Проверяется корректность длины вводимого числа. Если длина не соответствует описанию, выводится сообщение об ошибке и программа завершается.
9. Проверяется корректность ввода второго числа. Если ввод не соответствует описанию, выводится сообщение об ошибке и программа завершается.
10. Происходит умножение введенных чисел.
11. Выводится итоговый результат умножения.
12. Возвращается код ошибки.

Функция testInputFloat:

- Проверяет модульные тесты и выдает количество некорректных результатов

Функция take_null:

- Заполняет массив мантиссы нулями.
- Устанавливает порядок и размер в 0.
- Устанавливает знак мантиссы в 0.

Функция input_float:

- Принимает строку числа, указатель на структуру с максимальными размерами мантиссы и порядка числа.
- Инициализирует переменные и флаги.
- Проверяет знак числа и записывает его.
- Пропускает ведущие нули в строке.
- Обрабатывает мантиссу числа.
- Обрабатывает порядок числа.
- Проверяет ограничения на размеры мантиссы и порядка числа.
- Обнуляет поля структуры.
- Заполняет массив мантиссы числа.
- Рассчитывает смещение для порядка числа.
- Записывает цифру в мантиссу числа.
- Уменьшает счетчик текущей позиции мантиссы.
- Устанавливает размер мантиссы числа.
- Возвращает значение ERR_OK.

Функция shift_array_left:

Принимает массив, размер массива и количество позиций сдвига.
- Сдвигает элементы массива влево на указанное количество позиций.
- Возвращает измененный массив.

Функция print_res:

- Выводит символ "+" или "-" в зависимости от знака мантиссы.
- Выводит "0.".
- Для каждого элемента мантиссы от последнего до первого:
- Выводит значение элемента.
- Выводит "E" и символ "+" или "-" в зависимости от значения порядка.
- Выводит значение порядка.

Функция multiply:

- Создает массив и инициализирует переменные.
- Инициализирует результат нулями.
- Заполняет массив нулями.
- Выполняет цикл для каждого элемента первой структуры:
- Выполняет вложенный цикл для каждого элемента второй структуры:
- Увеличивает элемент массива на произведение соответствующих элементов мантисс.
- Пока значение элемента массива больше или равно базе и индекс меньше размера массива - 1:
- Увеличивает следующий элемент на частное от деления текущего элемента на базу.
- Заменяет текущий элемент на остаток от деления на базу.
- Увеличивает индекс на 1.
- Выполняет цикл, пока индекс не превышает размер массива:
- Проверяет элемент массива и устанавливает размер массива в соответствии с отличным от нуля значением.
- Увеличивает индекс на 1.
- Если размер массива больше максимального размера мантиссы:
- Рассчитывает разницу и увеличивает порядок на нее.
- Инициализирует переменные и цикл для сдвига элементов массива.
- Вызывает функцию для сдвига массива влево на разницу.
- Вычитает разницу из размера.
- Устанавливает знак мантиссы в зависимости от знаков мантисс первых двух структур.
- Увеличивает порядок на сумму порядков первых двух структур.
- Устанавливает размер мантиссы равным размеру массива.
- Копирует элементы массива в мантиссу результата.
- Возвращает ERR_OK.

Вывод по алгоритму работающей программы:

Поскольку программа оперирует с многоразрядными числами, удобно хранить их в массиве. Вызов функции take_null нужен для инициализации результата, после умножения чисел результат будет храниться в этом массиве.

Заполнение массива нулями нужно для того, чтобы изначально все ячейки массива имели значение 0 и не заполнялись лишними данными. Цикл в программе нужен для операции умножения чисел в столбик.

Вложенный цикл также используется для произведения умножения каждой цифры первого числа на каждую цифру второго числа, введенного пользователем.
Так как результат может быть большим числом, исчисление происходит по разрядам.

В случае, если значение в какой-то разряд превышает порог, оно переносится на следующий разряд и выполняется дополнительная операция.
Если размер полученного массива превышает максимальный размер заданного мантиссы, то происходит обработка переполнения.

Знак результата определяется знаком умножаемых чисел. Если они отличаются, то результат будет отрицательным, иначе положительным.
Порядок результата равен сумме порядков умножаемых чисел.

Результат копируется в структуру, чтобы можно было его вывести или продолжить операции с ним. Возвращается код ошибки ERR_OK, чтобы показать, что программа успешно выполнена.

Такой алгоритм был выбран, потому что он позволяет умножать многоразрядные числа, учитывать переносы и получать корректный результат. Благодаря использованию массива и циклов, умножение выполняется поэтапно и эффективно

Таблица положительных тестов.

Положительные тесты	**Ввод**	**Вывод**
Максимальное значение порядка	99999999999999E0 9	+0.1E+99999
Минимальное значение порядка	0.01E-99998 1	+0.1E -99999
Умножение положительного числа на положительное	3E0 2	+0.6E+1
Умножение положительного числа на отрицательное	11 -4	-0.44E+2
Умножение отрицательного числа на отрицательное	-11 -4	+0.44E+2
Умножение отрицательного числа на положительное	-11 4	-0.44E+2
Отсутствует знак у всех чисел	11 4	+0.44E+2
Результат округляется в меньшую сторону	5.8E2 1.2E-1	+0.696E+0
Числа имеют максимальный размер	99999999999999999999999999999999, 99999999999999999999999999999999	+0.99999999999999999999999999999998999990E+73
Результат округляется в большую сторону	-3.2E3 -2.5E-2	+0.800E+2
Умножение числа на 1	+2.03E1 1	+0.203E+2
Умножение числа, которое содержит символ "."	11.1 2	+0.22E+1
Один из множителей равен 0	2E0 0	+0.E+0
Умножение двух простых чисел	4 48	+0.192E+3
Округление результата	2 99999999999999999999999999999999	+0.200000000000000000000000000000000000000000E+41

Таблица негативных тестов.

Негативные тесты	Ввод	Вывод
Первое число содержит лишние символы "."	0.1.E10 0.1E2	Некорректное набранное первое число, ввод должен соответствовать описанию
Второе число содержит лишние символы "."	0.1E10 0.1E.2	Некорректное набранное второе число, ввод должен соответствовать описанию
Первое число содержит лишние символы "+"	0.1E++10 0.1E2	Некорректное набранное первое число, ввод должен соответствовать описанию
Второе число содержит лишние символы "-"	0.1E10 0.1E--2	Некорректное набранное второе число, ввод должен соответствовать описанию
Пустая строка вместо первого числа	“ “ 0.1E10	Некорректное набранное первое число, ввод должен соответствовать описанию
Пустая строка вместо второго числа	0.1E10 “ “	Некорректное набранное второе число, ввод должен соответствовать описанию
Первое число содержит недопустимые символы	Hello 0.1E10	Некорректное набранное первое число, ввод должен соответствовать описанию
Второе число содержит недопустимые символы	0.1E10 hello	Некорректное набранное второе число, ввод должен соответствовать описанию
Превышение порядка	1E99999999999999999999999999999	Некорректное набранное первое число, ввод должен соответствовать описанию
Первое число превышает максимальный допустимый размер	99999999999999999999999999999E0 1	Некорректная длина первого вводимого числа, длина должна соответствовать описанию
Второе число превышает максимальный допустимый размер	9 99999999999999999999999999999E0 1	Некорректная длина второго вводимого числа, длина должна соответствовать описанию
Переполнение порядка	0035E99999 1	Переполнение порядка

Оценка эффективности

1.Время выполнения: В программе используется функция умножения, которая выполняет умножение двух чисел. Программа работает достаточно быстро и эффективно для заданных входных данных.

2.Потребление памяти: Программа использует структуры для хранения чисел и результата, а также массивы символов для ввода чисел.

3.Надежность: Программа корректно обрабатывает все возможные варианты входных данных и устойчива к ошибкам.

4.Удобство использования: Пользователь видит информационное сообщение, вводит необходимые значения с клавиатуры, получает результат на экран.

5.Расширяемость: Программа имеет модульную структуру, что позволяет легко добавлять новую функциональность и модифицировать существующий код, не затрагивая другие части программы. Это способствует легкой расширяемости программы и облегчает ее поддержку и сопровождение.

6.Корректность: созданы функциональные тесты для корректной работы умножения двух действительных чисел, в коде присутствует проверка на ограничения. Программа работает корректно и дает правильные результаты для всех возможных входных данных.

С учетом заданных условий и работы с многоразрядными числами, я постаралась осуществить максимально эффективный код.
Так как результат может быть большим числом, исчисление происходит по разрядам. В случае, если значение в какой-то разряд превышает порог, оно переносится на следующий разряд и выполняется дополнительная операция.
Такой алгоритм был выбран, потому что он позволяет умножать многоразрядные числа, учитывать переносы и получать корректный результат. Благодаря использованию массива и циклов, умножение выполняется поэтапно и эффективно.
В целом, эффективность данного кода зависит от размеров входных данных. Это позволяет оценить его производительность и улучшить его при необходимости.

Вывод

В результате проведенной работы был разработан эффективный алгоритм умножения многоразрядных чисел, который позволяет учитывать переносы и получать корректный результат. Алгоритм основан на использовании массива для хранения чисел и циклов для выполнения умножения поэтапно.
Оценка эффективности данного алгоритма проведена с учетом заданных условий и работы с многоразрядными числами. Было определено, что использование массива и циклов позволяет выполнить умножение точно, с учетом переносов и заполнения массивов нулями.
Полученный результат является многоразрядным числом, и его размер может превышать максимальный размер заданной мантиссы. В случае переполнения происходит соответствующая обработка переполнения.
Определяется знак результата, который зависит от знаков умножаемых чисел. Если они отличаются, результат будет отрицательным, в противном случае положительным. Порядок результата равен сумме порядков умножаемых чисел. Копирование результата в структуру позволяет дальше с ним работать или вывести его на экран. Код ошибки ERR_OK возвращается для показа успешного выполнения программы.
Такой выбранный алгоритм позволяет эффективно умножать многоразрядные числа, учитывать переносы и получать корректный результат. Использование массива и циклов позволяет выполнить умножение поэтапно и эффективно. Вывод результатов операции на экран также был реализован.

Ответы на контрольные вопросы

1. Каков возможный диапазон чисел, представляемых в ПК?

Диапазон чисел, представляемых в ПК, зависит от длины машинного слова. Например, в 32-разрядной системе возможный диапазон целых знаковых чисел составляет от -2^31 до 2^31-1, а в 64-разрядной системе - от -2^63 до 2^63-1.

2.Какова возможная точность представления чисел?

Точность представления чисел зависит от длины мантиссы. Наиболее точное представление числа достигается при использовании длины мантиссы, равной размеру машинного слова. Например, в 64-разрядной системе максимальная точность составляет 52 двоичных разряда.

3. Какие стандартные операции возможны над числами?

Стандартные операции, которые можно выполнять над числами, включают сложение, вычитание, умножение и деление.

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. Например, можно использовать специализированные библиотеки для работы с произвольной точностью или реализовать собственный тип данных.

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

Операции над числами, выходящими за рамки машинного представления, можно осуществить путем разделения чисел на более мелкие блоки и вычисления операций с каждым блоком отдельно. Также они могут быть осуществлены с использованием библиотек для работы с большими числами. Эти библиотеки позволяют представлять и выполнять операции с числами, имеющими произвольную длину, и не ограничены размером машинного слова. В таких библиотеках и алгоритмах представление чисел может быть реализовано с использованием более сложных структур, таких как массивы или списки, а операции выполняются путем обхода и манипулирования этими структурами данных. Это позволяет работать с числами любой разрядности и выполнить операции, которые не уместились бы в пределах машинного представления.