|  |  |
| --- | --- |
|  | **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**

**Лабораторная работа №1**

**«Обработка больших чисел»**

Группа ИУ7-34Б

Дисциплина Типы и структуры данных

Вариант 14

|  |  |  |
| --- | --- | --- |
| Студент |  | Козлитин Максим Александрович |
| Преподаватель |  | Силантьева Александра Васильевна |

Условие задачи

Реализация арифметических операций над числами, выходящими за разрядную сетку персонального компьютера, выбор необходимых типов данных для хранения и обработки указанных чисел.

Описание задачи, реализуемой программой

Смоделировать операцию умножения действительного числа в форме ±m.n Е ±K, где суммарная длина мантиссы (m+n) - до 30 значащих цифр, а величина порядка K - до 5 цифр, на целое число длиной до 30 десятичных цифр. Результат выдать в форме ±0.m1 Е ±K1, где m1 - до 30 значащих цифр, а K1 - до 5 цифр.

Описание исходных данных

1. Входное действительное число  
     
   Вводится в формате ±m.n Е ±K, где суммарная длина мантиссы (m+n) - до 30 значащих цифр, а величина порядка K - до 5 цифр (от -99999 до 99999).  
   m - целая часть мантиссы.  
   n - дробная часть мантиссы.  
   E - символ порядка.  
   K - порядок или показатель степени.  
   E±K - 10^(±K).  
   ±m.n Е ±K - тоже самое, что ±m.n\*10^(±K).  
     
   Для ввода допускаются символы [+-0123456789.eE].  
     
   Символы знака мантиссы и порядка могут отсутствовать и воспринимаются как +.  
     
   Мантисса может содержать точку, но только в единственном числе. При этом не требуется обязательно указывать целую или дробную часть - отсутствие воспринимается как ноль.  
   Допускается отсутствие мантиссы, при этом ввод порядка - **обязателен**.  
     
   Порядок содержит только цифры и знак, при этом для указания должен предшествовать символ экспоненты. Цифры могут отсутствовать, в этом случае воспринимаются как ноль.  
   Экспонента может полностью отсутствовать.
2. Входное целое число  
     
   Вводится в формате ±d, где длинна d - до 30 значащих цифр.  
   d - цифры целого числа.  
     
   Для ввода допускаются символы [+-0123456789].  
     
   Знак числа может как присутствовать, так и отсутствовать при вводе.
3. Результат - действительное число  
     
   Выводится в форме ±0.m1 Е ±K1, где m1 - до 30 значащих цифр, а K1 - до 5 цифр.  
   m1 - дробная часть мантиссы (так как вид то нормализованный то включает в себя целую и дробную части мантиссы до нормализации).  
   K1 - порядок или показатель степени.  
   E±K - 10^(±K1).  
   ±0.m1 Е ±K1 - тоже самое, что ±0.m1\*10^(±K1).  
     
   При переполнении результата - число округляется.  
     
   Знак порядка и экспоненты присутствует всегда.  
   Целая и дробная части мантиссы, а также порядок - присутствуют всегда.

Способ обращения к программе

Работа с программой осуществляется с помощью консоли.

Программа запускается с помощью команды: **./app.exe**

Пользователю предлагается ввести ввести два числа, требуемые для совершения операции умножения.

В ответ выводятся - исходное выражение и ответ.

| Пример работы программы: |
| --- |
| ./app.exe  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  | [012345678901234567890123456789|01234] |  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  Введите первый множитель| 1  Введите второй множитель| 1  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  Первый множитель | +0.1E+1  Операция |\* |  Второй множитель | +1  ------------------------|------------------------------------------|  Результат | +0.1E+1  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |

Описания возможных аварийных ситуаций и ошибок пользователя

1. Ввод максимального действительного числа хоть и допускается, но так как не может быть нормализован, то обрабатывается как некорректный случай.
2. Ввод экспоненты превышающей максимальное допустимое значение типа данных int
3. Ввод не совпадающий с форматом входных данных (описано в [Описание исходных данных](#ОписаниеИсходныхДанных)).
4. Переполнение порядка или мантиссы в результате операции умножения.

Описание внутренних структур данных

| Действительное число |
| --- |
| **typedef** **struct** |
| { |
| **char** m\_sign; |
| **size\_t** m\_size; |
| **size\_t** dot\_pos; |
| **char** mantissa[MANTISSA\_SIZE]; |
| **int** exponent; |
| **int** normalized; |
| } **big\_real\_t**; |
| Мантисса хранится в виде 4 переменных:   * Знак - символ char. * Размер - используется тип данных size\_t, размер которого рассчитывается так, чтобы в него можно было записать максимальный размер теоретически возможного массива любого типа. * Позиция точки. * Массив цифр - использует тип данных char, так как требуется хранить только цифры от 0 до 9, соответственно можно не использовать более вместительный тип данных.   MANTISSA\_SIZE - максимальная длина целого числа (30 символов).  Экспонента хранится в виде целого числа.  Дополнительно хранится информация о том, приведено ли число к нормализованному виду. Это позволяет избежать повторных попыток нормализовать число. |

| Целое число |
| --- |
| **typedef** **struct** |
| { |
| **char** sign; |
| **size\_t** size; |
| **char** digits[BIG\_INT\_SIZE]; |
| } **big\_int\_t**; |
| Знак числа - символ char.  Размер - используется тип данных size\_t, размер которого рассчитывается так, чтобы в него можно было записать максимальный размер теоретически возможного массива любого типа.  Массив цифр - использует тип данных char, так как требуется хранить только цифры от 0 до 9, соответственно можно не использовать более вместительный тип данных.  BIG\_INT\_SIZE - максимальная длина целого числа (30 символов). |

| Целое число - для сохранения промежуточных вычислений |
| --- |
| **typedef** **struct** |
| { |
| **size\_t** size; |
| **char** data[TEMP\_SIZE]; |
| } **temp\_t**; |
| Используется для сохранения результата промежуточных вычислений. Необходимость использования обусловлена возможностью округлить ответ, в случае выхода за пределы действительного числа.  Размер - используется тип данных size\_t, размер которого рассчитывается так, чтобы в него можно было записать максимальный размер теоретически возможного массива любого типа.  Массив цифр - использует тип данных char, так как требуется хранить только цифры от 0 до 9, соответственно можно не использовать более вместительный тип данных.  TEMP\_SIZE - максимальная длина целого числа промежуточного вычисления (60 символов), основывается на максимальной длине которое может получиться при умножении, т.е. MANTISSA\_SIZE + BIG\_INT\_SIZE. |

Описание алгоритма

**Умножение** - функция **bnumsop\_multiply:**

1. Действительное число приводится к нормализованному виду.
2. Действительное число посимвольно умножается на каждую цифру целого числа (в столбик).
3. Результат сначала записывается в развернутом виде, затем переворачивается.  
     
   Хранится 3 переменных для промежуточных вычислений - предыдущее слагаемое, текущее слагаемое, общая сумма.
4. К полученному произведению добавляются нули в конец, в зависимости от того какое по счету умножение производится - 0 нулей для первого, 1 ноль для второго, 2 нуля для третьего вычислений и т. д.
5. При первом умножении результат записывается в текущую сумму и предыдущее слагаемое. Иначе производится вычисление суммы предыдущего слагаемого и текущего, результат записывается в текущую сумму и предыдущее слагаемое.
6. Вычисленное в итоге произведение преобразуется к действительному числу, при этом порядок берется из исходного действительного числа, но обновляется, если исходное действительное число меньшей длины, чем результат (т.к. точка уже стоит не в начале), т.е. добавляется разница между размером числа итогового произведения и размером исходного действительного числа.
7. Знак итогового действительного числа - отрицательный, если знаки отличаются, иначе - положительный.  
     
   Полю, хранящему информацию, число нормализовано или нет, присваивается 0.
8. Проверяется переполнение порядка.
9. Возвращается статус успешности выполнения функции.

**Сумма -** функция **temp\_sum:**

1. Два целых числа посимвольно сумируются (в столбик).
2. Результат сначала записывается в развернутом виде, затем переворачивается.

**Преобразование промежуточного числа к действительному -** функция **bnumsop\_copy:**

1. Число округляется.
2. Пересчитывается порядок, т.е прибавляем значение счетчика, которое вернет функция округления, к порядку.
3. Копируется в переменную действительного числа.

**Округление -** функция **bnumsop\_round:**

1. Если размер числа больше требуемого, то начинаем итерируемся по цифрам с конца.
2. Если текущий размер числа меньше или равен требуемой точности округления. Проверяем округляется текущий разряд вверх, если да - то округляем и сохраняем в переменную прибавку к следующему разряду, если нет - то останавливаемся.  
      
   Если текущий разряд больше или равен 5, то прибавка к следующему разряд - 1, иначе - 0.
3. Иначе просто добавляем к текущему разряду значение прибавки, сохраненное на прошлом разряде, и уменьшаем счетчик на 1 (счетчик отвечает за уменьшения порядка)
4. Так как мы итерируемся с конца, то при циклическом поразрядном переносе размер числа станет равен 0, в таком случае увеличиваем его на 1.
5. Возвращаем счетчик, отвечающий за уменьшение порядка.

Тестовые случаи

| № | Позитивные тесты | | |
| --- | --- | --- | --- |
| Что проверяется | Входные данные | Выходные данные |
| 1 | Отсутствие знака мантиссы | 1234567E20  2 | +0.2469134E+27 |
| 2 | Отсутствие мантиссы | .E-123  1 | +0.0E+0 |
| 3 | Отсутствие точки | 123E+10  1 | +0.123E+13 |
| 4 | Отсутствие экспоненты | 123.123  1 | +0.123123E+3 |
| 5 | Отсутствие знака экспоненты | 1e10  1 | +0.1E+11 |
| 6 | Отсутствие мантиссы и знака мантиссы | +.E-3  1 | +0.0E+0 |
| 7 | Отсутствие мантиссы, знака мантиссы и точки | E-3  1 | +0.0E+0 |
| 8 | Отсутствие экспоненты и точки (т. е. целое число) | 11111111  9 | +0.99999999E+8 |
| 9 | Максимально допустимое значение получается при нормализации | 999999999999999999999999999999E+99969  1 | +0.999999999999999999999999999999E+99999 |
| 10 | Умножение нуля на другое число | 0  9999999 | +0.0E+0 |
| 11 | Умножение на ноль | 999999E-12415  0 | +0.0E+0 |
| 12 | Проверка округления при максимальных значениях мантиссы и целого числа | 999999999999999999999999999999  99999 | +0.999989999999999999999999999999E+35 |
| 13 | Результат такого же размера как исходное | 1e10  5 | +0.5E+11 |
| 14 | Результат длиннее исходного | 1e2  100 | +0.1E+5 |
| 15 | Ведущие нули | 0000000000000000000000000000001e+0000010  1 | +0.1E+11 |
| 16 | Тест при средних значениях | 999E151  1234 | +0.1232766E+158 |
| 17 | Достаточно большие числа, округление | 9E90000  9999999 | +0.89999991E+90008 |
| 18 | Знак результата, два отрицательных | -9  -1 | +0.9E+1 |
| 19 | Знак результата, первое число - отрицательное | -9  1 | -0.9E+1 |
| 20 | Знак результата, второе число - отрицательный | 9  -1 | -0.9E+1 |
| 21 | Циклический поразрядный перенос | 999999999999999999999999999999  3 | +0.3E+31 |
| 22 | Циклический поразрядный перенос  (в обратную сторону) | 3  999999999999999999999999999999 | +0.3E+31 |
| 23 | Целая часть равна нулю  Ведущие нули в дробной части | 0.00044  2 | +0.88E-3 |
|  | Циклический поразрядный перенос, граничное число при округлении | 999999999999999999999999999999  5 | +0.5E+31 |
|  | Циклический поразрядный перенос, граничное число при округлении  (в обратную сторону) | 5  999999999999999999999999999999 | +0.5E+31 |

| № | Негативные тесты | | |
| --- | --- | --- | --- |
| Что проверяется | Входные данные | Выходные данные |
| 1 | Максимальное значение действительного числа  (переполнение экспоненты при нормализации) | 0.001e-99999  1000 | Ошибка: Порядок переполнился! |
| 2 | Переполнение экспоненты при умножении | 999999999999999999999999999999E99969  99999 | Ошибка: Порядок переполнился! |
| 3 | Пустой ввод |  | Ошибка: Вы не ввели первое число! |
| 4 | Слишком большая мантисса при вводе | 999999999999999999999999999999999999999E0 | Ошибка: Вы ввели слишком много! |
| 5 | Целое число не введено | 1231231 | Ошибка: Вы не ввели второе число! |
| 6 | Действительное число не введено | 123 | Ошибка: Вы не ввели первое число! |
| 7 | Лишний знак экспоненты | -1EE  1 | Ошибка: Вы ввели недопустимые символы! |
| 8 | Второе число - действительное | 31231  -123E1 | Ошибка: Вы ввели недопустимые символы! |
| 9 | Лишний знак плюса | ++1e1  1 | Ошибка: Вы ввели недопустимые символы! |
| 10 | Две точки | 1.1245.12e+1  1 | Ошибка: Вы ввели недопустимые символы! |
| 11 | Буква во вводе (не Е) | 1244a124  1 | Ошибка: Вы ввели недопустимые символы! |
| 12 | Слишком большое целое (31 знак) | 1e1  9999999999999999999999999999999 | Ошибка: Слишком большое целое число! |

Оценка эффективности

M - макс. размер мантиссы

N - макс. размер целого числа

T - макс. размер промежуточного числа

T >= M + N

Разворот промежуточного: O\_1(T / 2) ~ O\_1(T) - линейное время работы.

Сумма: O\_2(T + 1 + O\_1) ~ O\_2(T) - линейной время работы.

Округление: O\_3(T) - линейной время работы.

Преобразование к действительному: O\_4(O\_3 + M) ~ O\_4(T + M) ~ O\_4(max(T, M)) ~ O\_4(T) - линейное время работы.

Умножение: O(N \* (M + O\_1 + N) + (N - 1) \* (O\_3) + O\_4) ~ O(N \* (M + T + N) + N \* T + T) ~ O(N \* max(M, T, N) + N \* T + T) ~ O(N \* max(M, T, N) + N \* T + T) ~ O(N \* max(M, T, N)) ~ O(N \* T) - квадратичное время работы.

Итоговая асимптотика: O(N \* T) - квадратичное время работы.

Вывод

В ходе проделанной работы, я научился обрабатывать числа, выходящие за разрядную сетку персонального компьютера. В частности выполнять операции сложения и умножения между целыми и действительными числами. Удобно выполнять арифметические операции в столбик, благодаря наглядности алгоритма и простоте понимания. Стоит отметить, что сам алгоритм при этом не является очень быстрым и имеет квадратичную асимптотику. Для хранения же больших чисел приоритетно использование структур данных (запись), таким образом повышается читаемость исходного кода и удобство использования данного числа.

Контрольные вопросы

1. **Каков возможный диапазон чисел, представляемых в ПК?**   
   Целые положительные числа: 0 < x <= 2^n – 1   
   Целые отрицательные числа: -2^n-1 <= x < 0  
   для n-разрядного машинного слова  
   Для 32-разрядного процессора: 2^32 - 1 = 4 294 967 295   
   Для 64-разрядного процессора: 2^64 - 1 = 18 446 744 073 709 551 615  
     
   Действительные числа: 3.6E–4951 <= x <= 1.1E+4932   
   (максимальный размер мантиссы 52 разряда, порядок – 11 разрядов.)  
     
   *Примечание:* *Имеется ввиду двоичные разряды, а не десятичные*
2. **Какова возможная точность представления чисел, чем она определяется?**   
   Точность представления вещественного числа зависит от максимально возможной длины мантиссы, которая, опять-таки, зависит от области выделяемой памяти и наличия знака. Если длина мантиссы выходит за границы разрядной сетки, то происходит округление. Обычно, длина мантиссы это 20 десятичных разрядов.
3. **Какие стандартные операции возможны над числами?**   
   Сложение, вычитание, умножение, деление, сравнение.
4. **Какой тип данных может выбрать программист, если обрабатываемые числа превышают возможный диапазон представления чисел в ПК?**Можно представлять числа в виде массивов цифр. Таким образом числа будут обрабатываться посимвольно, соответственно любые операции над ними также будут производиться посимвольно.
5. **Как можно осуществить операции над числами, выходящими за рамки машинного представления?**В этом случае мы представляем число в виде массива цифр и обрабатываем поэлементно. Арифметические операции будет удобно выполнять в столбик, за счет наглядности метода и очевидной реализации.