|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**ОТЧЁТ ПО ЛАБОРАТРОНОЙ РАБОТЕ №1**

**«ОБРАБОТКА БОЛЬШИХ ДАННЫХ»**

**по курсу «Типы и структуры данных»**

Студент: Чепиго Дарья Станиславовна

Группа: ИУ7-34Б

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Чепиго Д.С

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Барышникова М.Ю.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2021 г

**Условие задачи**

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.

**Техническое задание**

*Входные данные*

Одно действительное десятичное число, записанное в формате [±]mmm.nnn[e][±]ppppp (в квадратных скобках даны варианты символа на одной и той же позиции), где m, n, p – цифры, причем суммарное количество цифр до и после точки не превышает 30, число цифр после знака порядка не превышает 5 , а сам порядок по модулю меньше, чем 99999, буква «e» – латинские, знак «+» или «-» у мантиссы и у порядка должен обязательно присутствовать. Ведущие нули в целом числе включаются в подсчет длины числа.

Одно целое десятичное число, состоящее не более чем из 30 цифр, перед которыми обязательно должен быть знак «+» или «-».

*Выходные данные*

Одно действительное число, записанное в формате [±]0.mmmm[E/e][±]ppppp (в квадратных скобках даны варианты символа на одной и той же позиции), где m, p – цифры, причем количество цифр в мантиссе не превышает 30, знак порядка может присутствовать, число цифр после знака порядка не превышает 5.

*Способ передачи входных данных*

Входные данные передаются программе путем ввода в стандартный поток ввода на соответствующий запрос программы.

*Способ получения выходных данных*

Выходные данные выводятся в стандартный поток вывода.

*Задача программы*

Программа, получив на вход два числа, осуществляет их умножение и выводит результат в указанном формате. При невозможности получения результата программа выводит сообщение об ошибке.

*Способ обращения к программе*

Обращение к программе осуществляется путем запуска исполняемого файла программы app.exe.

*Возможные ошибочные ситуации*

1. Отсутствие одного или двух чисел в стандартном потоке ввода;
2. Некорректный ввод одного или двух чисел;
3. Переполнение порядка результата.

**Описание используемых структур данных**

*Строка, содержащая число*

Содержит символы, введенные пользователем. Для целого числа длина такой строки равна 32 (знак числа, 30 цифр, символ перехода на новую строку «\n» и символ конца строки «\0»), для вещественного – 41 (знак мантиссы, 30 цифр в мантиссе, точка в мантиссе, знак порядка, 5 цифр в порядке, символ перехода на новую строку «\n» и символ конца строки «\0»).

char real\_num[MAX\_STR]; - строка для действительного числа (MAX\_STR = 41)

char int\_num[MAX\_INT]; - строка для целого числа (MAX\_INT = 33)

*Массив цифр*

Для хранения результата произведения целого числа и мантиссы действительного числа используется массив цифр. Массив имеет длину 60. Это сделано для того, чтобы обеспечить корректность вычислений при переходе через десяток, когда длина мантиссы максимально допустимая. Каждая цифра имеет тип int, в связи с используемым алгоритмом. Цифры записываются в массив справа налево, то есть последняя цифра числа имеет индекс 0, предпоследняя 1 и т. д. Это сделано для удобства вычислений.

int result[MAX\_MULTI] = {0}; - массив для произведения двух чисел (MAX\_MULTI = 60)

*Структура «длинное вещественное»*

Содержит строку, содержащую все число (со знаками, точками и символом экспоненты), знак мантиссы типа char, массив цифр мантиссы типа char, знак порядка типа char, значение порядка типа int и положение точки типа size\_t. Данные хранятся в одной структуре для обеспечения удобной совместной обработки всех данных.

*struct* **real\_number**

{

char real\_num[MAX\_STR]; - строка для действительного числа (MAX\_STR = 41)

char sign\_mantissa; - символ знака мантиссы

char mantissa[MAX\_MANTISSA]; - строка для цифр мантиссы (MAX\_MANTISSA = 32)

char sign\_order; - символ знака порядка

int order; - массив, содержащий порядок

size\_t point\_place; - позиция точки в мантиссе

};

Структура «длинное целое»

Данная структура содержит знак целого числа типа char, массив его цифр типа char. Данные хранятся в одной структуре для обеспечения целостной обработки данных.

*struct* **int\_number**

{

char sign\_int; - символ знака числа

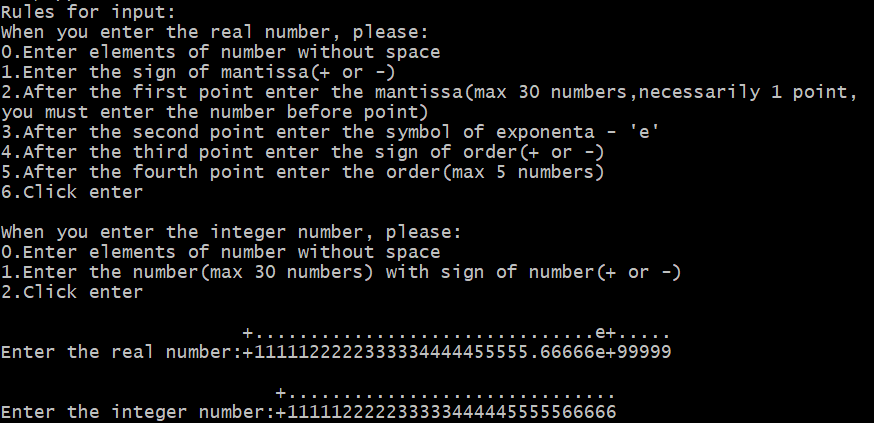
char int\_num[MAX\_INT]; - массив, содержащий целое число (MAX\_INT = 33)

}

**Описание алгоритма**

*1.Ввод данных*

При запуске программы на экран выводятся правила, которые надо соблюдать при вводе чисел. Если хотя бы одно из них не выполнено – программа выдаст ошибку. Ниже – пример программы с правилами и с удобным примером для ввода.



Считываем действительное и целое число в соответствующие строку из стандартного потока ввода. Если они считались успешно, то далее из строк записываем данные в соответствующие элементы структуры.

*2. Запись во внутренние структуры*

Обрабатывая строку действительного числа по очереди проверяется все её содержимое, а именно наличие знаков, символа экспоненты, корректность мантиссы и порядка, находится позиция точки. После этого высчитывается порядок чисел и осуществляется нормализация. Далее удаляется точка в массиве чисел мантиссы для алгоритма умножения.

Целое число аналогично проверяется на наличие знака и корректные цифры. Запоминается знак, цифры записываются в массив символов.

Если в ходе обработки входных данных произошла ошибка, выводим сообщение и завершаем работу программы.

*3.Вычисления*

*Умножение двух чисел*

Для алгоритма умножения двух чисел, размер которых максимум 30 цифр создаётся массив на 60 символов. Это реализуется с помощью двух циклов for с переменной i для первого числа и переменной j для второго числа соответственно. По очереди все цифры первого числа умножаются на все числа второго числа, их произведение записывается в ячейку массива под номером (60 – I - j). При этом после завершения этих циклов происходит проход по массиву с конца для переноса чисел, которые больше 10 в следующий разряд соответственно.

*Вычисление порядка*

После того, как мы знаем количество значащих цифр в результате нашего умножения мы можем вычислить порядок по формуле =количество значащих цифр в результате – изначальный порядок действительного числа – количество цифр в действительном числе после точки. Если при округлении первой цифрой числа была “9” и это округлилось, то порядок меняется на 1.

*Корректировка мантиссы*

Число знаков в мантиссе может превысить максимально допустимое. Найдем разницу между числом знаков результата и максимально допустимым. Если оно не положительно, корректировка не требуется. В противном случае сдвинем разряды результата так, что первая цифра займет 30-е справа место, предварительно запомнив цифру, идущую после первой справа сохраняемой цифры. Если запомненная цифра от 5 до 9, то прибавим 1 в последнем разряде мантиссы.

*Определение знака*

Если два исходных числа имеют одинаковый знак, знак результата принимается равным «+», иначе – «-».

*4.Вывод результата или сообщения*

Если в ходе обработки входных данных произошла ошибка, выводим сообщение и завершаем работу программы. Если ошибки нет, выводим результат в указанном выше формате.

**Контрольные вопросы**

*1.Каков возможный диапазон чисел, представляемых в ПК?*

Если под хранение целого положительного числа выделено 16 разрядов, то его максимальное значение не может превышать 216 – 1 = 65 535, если выделено 32 разряда, то максимальное значение составит 232 – 1 = 4 294 967 295. Для 64 разрядов максимально возможное значение числа равно 264 – 1 = 18 446 744 073 709 551 615.

Вещественные числа представляются в виде мантиссы и порядка. Мантисса – правильная дробь, первая цифра которой отлична от нуля, т.е. находится в интервале [0.1, 1). Порядком – степень основания системы счисления, в которой сохранено число.

Длина мантиссы определяет точность представления числа, а длина порядка ограничивает диапазон допустимых значений. Максимально под представление мантиссы отводится 52 разряда в двоичной системе счисления, а под представление порядка – 11 разрядов. В этом случае возможные значения чисел находятся в диапазоне от 3.6 E –4951 до 1.1 E +4932.

*2.Какова возможная точность представления чисел?*

Целые числа хранятся в памяти ПК точно. Вещественные числа могут быть: одинарной точности, тогда точность составляет 7-8 значащих цифр после запятой; двойной точности с 15-16 значащими цифрами; четверной точности – около 34 значащих цифр.

3.*Какие стандартные операции возможны над числами?*

Набор операций над числом определяется его типом. Целые и вещественные числа поддерживают арифметические операции (сложение, вычитание, умножение, деление) и операции сравнения (кроме сравнения на равенство для вещественных чисел, их надо сравнивать, учитывая погрешность – некоторое маленькое число).

Целые числа в языках подобных Си также поддерживают битовые операции (битовые И, ИЛИ, ИЛИ-НЕ, НЕ) и операцию взятия остатка от деления, отдельная операция для целочисленного деления. В некоторых языках программирования, реализована операция возведения в степень.

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

Программист может сохранить цифры числа в массиве или разбить число на части, каждую часть сохранить как отдельное число, обработать эти части и затем склеить их для выдачи результата пользователю.

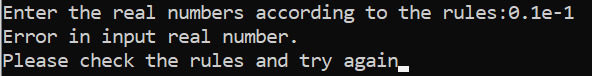
5.Как можно осуществить операции над числами, выходящими за рамки машинного представления?

Можно сохранить числа в самостоятельно разработанном представлении, как в 4 вопросе, и обработать сохраненные таким образом числа с использованием операций над используемыми типами данных. Для четырех арифметических операций – сложения, вычитания, умножения, деления – существует алгоритм вычисления результата «столбиком», который несложно реализовать самим. Для операций сравнения можно использовать поразрядное сравнение.

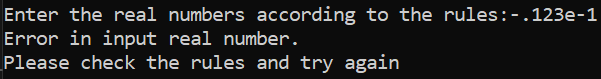
**Тестирование**

**Негативные тесты:**

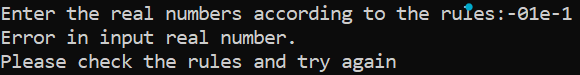
- 01 - у действительного числа отсутствует знак



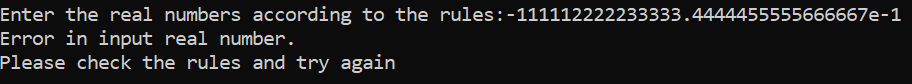
- 02 - у действительного числа нет чисел до точки



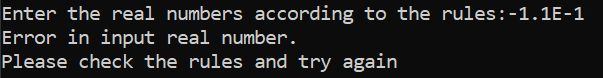
- 03 - у действительного числа нет точки



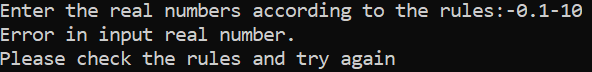
- 04 - у действительного числа мантисса больше, чем 31 знак



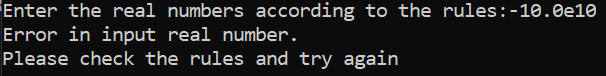
- 05 - у действительного числа неправильно написана экспонента



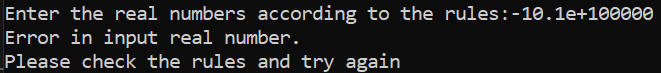
- 06 - у действительного числа нет экспоненты



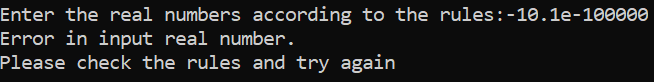
- 07 - у действительного числа нет знака у порядка



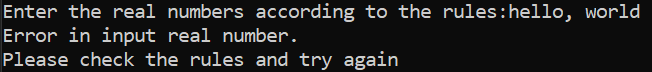
- 08 - у действительного числа порядок больше, чем +99999



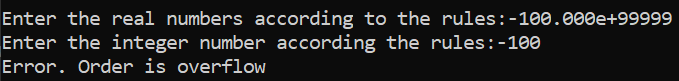
- 09 - у действительного числа порядок меньше, чем -99999



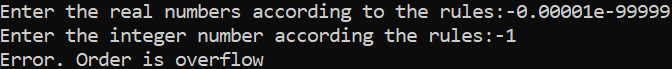
- 10 - действительное число содержит лишние символы



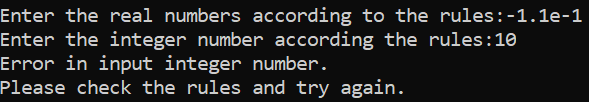
- 11 - при умножении порядок становится больше, чем +99999



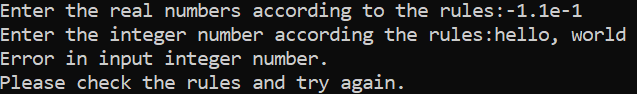
- 12 - при умножении порядок становится меньше, чем -99999



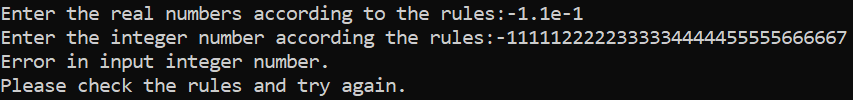
- 13 - у целого числа отсутствует знак



- 14 - целое число содержит лишние символы



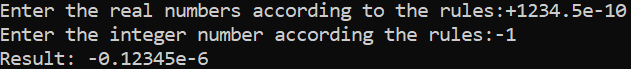
- 15 - целое число состоит больше чем из 30 цифр



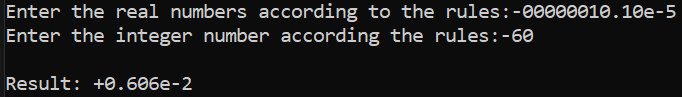
- 16 - после целого числа нет символа новой строки (проверяется через файл и консоль)

**Позитивные тесты:**

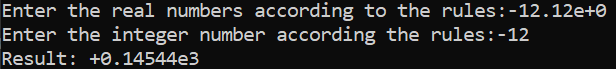
- 01 - действительное число умножается на единичку, то есть нормализуется с учётом знака



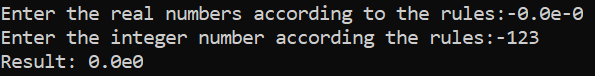
- 02 – действительное число имеет незначащие нули



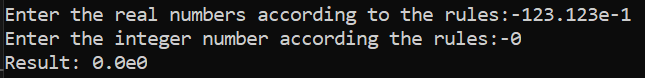
- 03 – действительное число имеет нулевой порядок



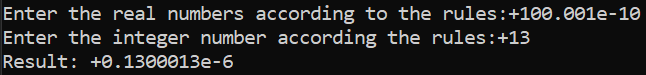
- 04 – действительное число равно 0



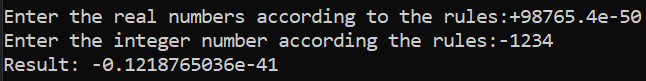
- 05 – целое число равно 0



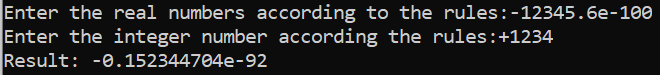
- 06 – действительное число положительное, целое число положительное



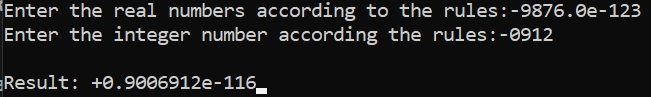
- 07 - действительное число положительное, целое число отрицательное



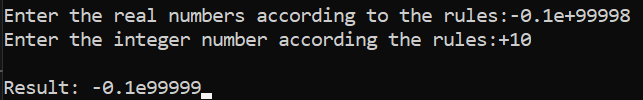
- 08 - действительное число отрицательное, целое число положительное



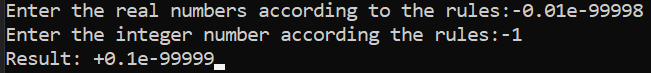
- 09 - действительное число отрицательное, целое число отрицательное



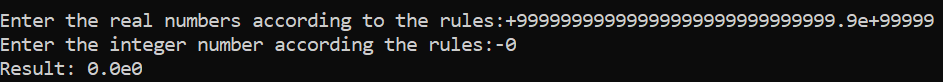
- 10 – при умножении порядок становится максимально возможным +99999

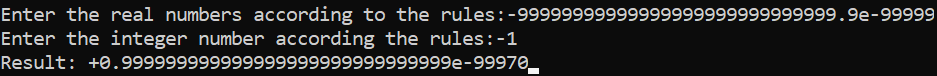


- 11 - при умножении порядок становится минимально возможным – 999999

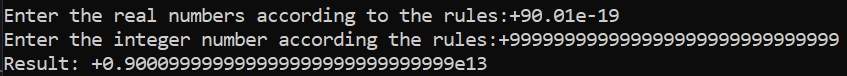


- 12 – ввод максимально возможного действительного числа

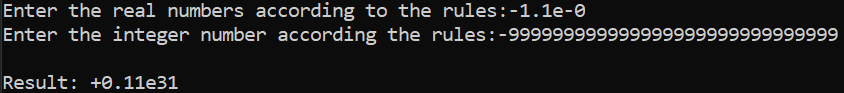


- 13 – ввод минимально возможного действительного числа 

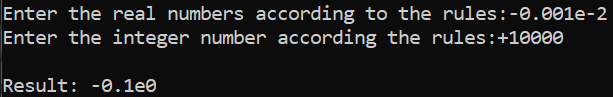
- 14 – ввод максимально возможного целого числа

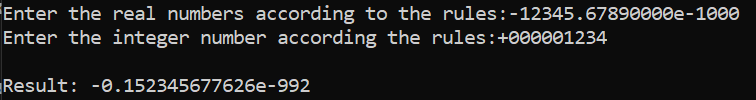


- 15 – ввод минимально возможного целого числа

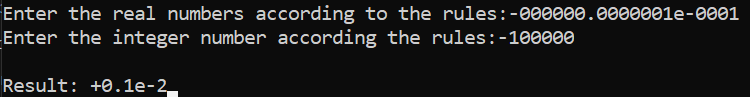


- 16 – при умножении порядок становится нулевым

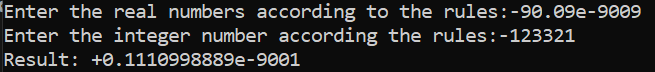


- 17 - обычный тест

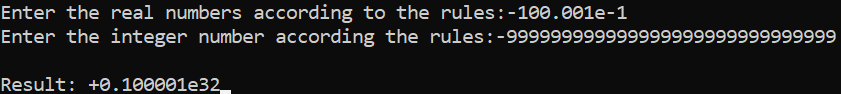
- 18 – обычный тест



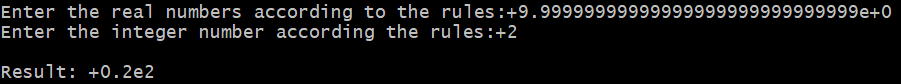
- 19 - обычный тест

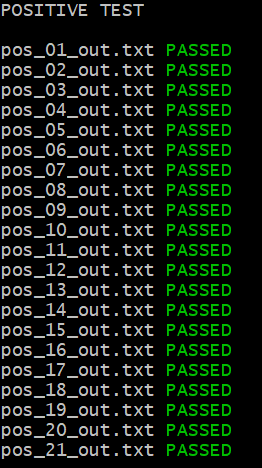
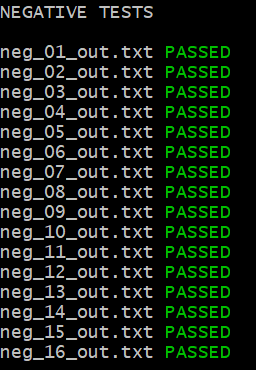
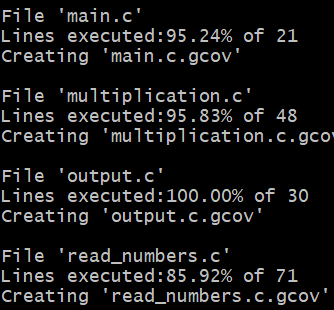


- 20 – умножение чисел с последующим округлением до 30 порядка



- 21 – умножение с округление



Сами тестовые файлы находятся в папке func\_tests.

**Выводы**

По итогам лабораторной работы заключаем следующее:

1. Целые и вещественные числа имеют различное хранение в памяти компьютера: целые – точно, а вещественные – приближенно (точность зависит от используемого типа данных).

2. Невозможно сохранить сколь угодно большое (маленькое) по модулю число со сколь угодно большой точностью.

3. В случае, если не удается использовать стандартные типы данных, то программист может самостоятельно решать этот вопрос разными способами.