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

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1 «ДЛИННАЯ» АРИФМЕТИКА. ТИП ДАННЫХ - МАССИВ**

Студент Ву Минь Куанг

Группа ИУ7И – 34Б

Преподаватель Силантьева А. В.

*2021 г.*

**Оглавление**

[**1.** **Описание условия задачи** 3](#_Toc82092627)

[**2.** **Описание ТЗ** 3](#_Toc82092628)

[**3.** **Описание внутренних структур данных** 5](#_Toc82092629)

[**4.** **Описание алгоритма** 6](#_Toc82092630)

[**5.** **Набор тестов с указанием, что проверяется** 7](#_Toc82092631)

[**6.** **Выводы по проделанной работе** 9](#_Toc82092632)

[**7.** **Ответы на вопросы** 10](#_Toc82092633)

# **Описание условия задачи**

Составить программу умножения или деления двух чисел, где порядок имеет до 5 знаков: от –99999 до +99999, а мантисса – до 30 знаков. Программа должна осуществлять ввод чисел и выдавать либо верный результат в указанном формате (при корректных данных), либо сообщение о невозможности произвести счет.

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

# **Описание ТЗ**

**Входные данные:**

Действительное число: строка, содержащая вещественное число в виде <+\-m.nE+\-K>. Знак перед числом и перед порядком обязательно нужно вводить. Также знак экспоненты <E> нужно вводить обязательно и в верхнем регистре. Суммарная длина <m+n> - до 31 цифры, включая точку; длина порядка — до 5 цифр. Пробелы при вводе числа недопустимы.

**Выходные данные:**

Длинное число в виде <+\-0.m1E+\-K1>. Длинна мантиссы <m1> - до 30 цифр; длинна порядка <K1> — до 5 цифр.

**Действие программы:**

Перемножение вещественных чисел.

**Аварийные ситуации:**

1. Некорректный ввод: строка с действительным числом не содержит знак мантиссы (+\-).

На выходе сообщение: «ERR\_UNRIGHT\_MANTISS\_SIGN»

1. Некорректный ввод: строка с вещественным числом содержит символ не цифру и не «.».

На выходе сообщение: «ERR\_UNRIGHT\_MANTISS\_NUMBER»

1. Некорректный ввод: строка с действительным числом не содержит знак экспоненты

На выходе сообщение: «ERR\_NO\_EPSILON»

1. Некорректный ввод: строка с вещественным числом содержит более одной точки.

На выходе сообщение: «ERR\_TOO\_MUCH\_POINTS»

1. Некорректный ввод: превышение длины при вводе вещественного числа (больше 31 цифры, включая точку)

На выходе сообщение: «ERR\_MANTISSA\_TOO\_LONG»

1. Некорректный ввод: введена только точка

На выходе сообщение: «ERR\_ONLY\_POINT»

1. Некорректный ввод: не введена точка

На выходе сообщение: «ERR\_NUMBER\_MUST\_BE\_FLOAT»

1. Некорректный ввод: введена пустая строка (т. е. «\n»).

На выходе сообщение: «ERR\_NO\_NUMBER»

1. Некорректный вывод: не введен знак порядка

На выходе сообщение: «ERR\_UNRIGHT\_POWER\_SIGN»

1. Некорректный вывод: в порядке введено не число

На выходе сообщение: «ERR\_UNRIGHT\_POWER»

1. Некорректный вывод: порядок превышает число «99999»

На выходе сообщение: «ERR\_POWER\_TOO\_LONG»

1. Некорректный вывод: переполнение порядка при умножении

На выходе сообщение: «ERR\_POWER\_OVERFLOW»

# **Описание внутренних структур данных**

При вводе число записывается в массив типа char, который хранит в себе все числа и точку. Также отдельно выбраны переменные, в которых хранится порядок числа и знаки мантиссы и порядка.

Число при вводе сразу записывается в нужные поля структуры float\_number.

\* MANTISSA\_MAX\_LEN = 32

Структура float\_number:

typedef struct

{

char mantis\_sign;

char mantissa[MANTISSA\_MAX\_LEN];

char eps\_sign;

int eps\_num;

int point\_place;

int num\_of\_digits;

} float\_number;

Поля структуры:

**mantis\_sign** – знак мантиссы

**mantissa** – мантисса числа

**eps\_sign** – знак экспоненты

**eps\_num** – значение порядка

**point\_place** – место точки в мантиссе

**num\_of\_digits** – количество цифр в мантиссе

После перемножения числа хранятся в дополнительном массиве result[60], который затем используется для вывода.

# **Описание алгоритма**

1. Программа считывает две строки, которые содержат вещественные числа, и записывает части числа в нужные места структуры *float\_number*

**(short int read\_number(float\_number \*number))**

1. Далее правильно введенные числа передаются в функцию нормализации, которая убирает ненужные нули и сдвигает точку в начало числа, изменяя его порядок

**(void normalization(float\_number \*number))**

1. После успешно проведенной нормализации числа передаются в функцию умножения, в которой создается массив максимально возможной длины числа с элементами типа *int* и в него по принципу умножения «в столбик» записываются числа, учитывая все переносы десятков в новые разряды числа

**(void check\_arr(int \*arr, int ind, int end\_arr))**

1. После умножения в числе проводится проверка, которая при необходимости, удаляет лишние нули или округляет число

**(void check\_arr(int \*arr, int ind, int end\_arr))**

1. Затем результат выводится в нормализованном виде в соответствии со спецификацией, указанной в ТЗ (<+\-0.m1E+\-K1>)

**(void print\_result(float\_number num1, float\_number num2, int \*result, int res\_power, int ind))**

**Функции программы**

1. ***short int read\_number(float\_number \*number)***
2. **Описание**: функция совершает чтение вещественного числа и его последующую запись в необходимые поля структуры *number*
3. **Входные значения**: структура *number* для записи в нее числа
4. **Выходные значения**: структура *number* с записанным в нее числом; *rc* – код ошибки (или ноль при ее отсутствии)
5. ***void normalization(float\_number \*number)***
6. **Описание**: функция нормализует мантиссу числа, записанную в *number*, и меняет порядок числа, а также удаляет лишние нули
7. **Входные значения**: структура *number* для нормализации числа
8. **Выходные значения**: структура *number* с нормализованным в ней числом
9. ***short int multiply(float\_number num1, float\_number num2, int \*result, int \*res\_power, int \*ind)***
10. **Описание**: Переводить каждые символы массива mantissa char на int и функция перемножает два вещественных числа
11. **Входные значения**: структуры num1 и num2, числа которых необходимо перемножить; массив *result*, в который будет записан результат вычислений; res\_*power* – порядок получившегося числа; *ind* – в каком месте массива *result* находится получившееся число
12. **Выходные значения**: массив *result*, в который будет записан результат вычислений; res\_*power* – порядок получившегося числа; *ind* – в каком месте массива *result* находится получившееся число; *rc* – код ошибки (или ноль при его отсутствии)
13. ***void check\_arr(int \*arr, int ind, int end\_arr)***
14. **Описание**: функция проверяет и округляет полученный результат перемножения
15. **Входные значения**: массив с получившемся числом после перемножения *arr*; *ind* – индекс начала числа в массиве; *end\_arr* – индекс конца числа в массиве
16. **Выходные значения**: округленное число в массиве *arr*
17. ***void print\_result(float\_number num1, float\_number num2, int \*result, int res\_power, int ind)***
18. **Описание**: функция печатает на экран результат перемножения двух вещественных чисел
19. **Входные значения**: структуры num1 и num2, знаки которых необходимо учесть при печати ответа; массив *result*, из которого будет распечатан результат вычислений; *res\_power* – порядок получившегося числа; *ind* – в каком месте массива *result* находится получившееся число
20. **Выходные значения**: функция ничего не возвращает

# **Набор тестов с указанием, что проверяется**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название теста | Число №1 | Число №2 | Вывод |
| 1 | Пустой ввод | - | - | Incorrect input. |
| 2 | Обычный тест | +12.3E+3 | -4.3E+3 | -0.5289E+8 |
| 3 | Не введен знак мантиссы | quang | - | ERR\_UNRIGHT\_MANTISS\_SIGN |
| 4 | В мантиссе введены неверные символы | -qwerty | - | ERR\_UNRIGHT\_MANTISS\_NUMBER |
| 5 | Введена только точка | -.E-00 | - | ERR\_ONLY\_POINT |
| 6 | Не введена экспонента | -0.123 | - | ERR\_NO\_EPSILON |
| 7 | Порядок содержит недопустимые символы | +12.3E+iu | - | ERR\_UNRIGHT\_POWER |
| 8 | Не введен знак порядка | +12.3E3 | - | ERR\_UNRIGHT\_POWER\_SIGN |
| 9 | Мантисса содержит более 30 цифр | +0.999..99E-3 (31 девятка) | - | ERR\_MANTISSA\_TOO\_LONG |
| 10 | Порядок состоит более, чем из 5 цифр | -.123E+999999 | - | ERR\_POWER\_TOO\_LONG |
| 11 | Введено более 1 точки в мантиссе | -.12.3E+99 | - | ERR\_TOO\_MUCH\_POINTS |
| 12 | Не введена точка в мантиссе | -123E-00 | - | ERR\_NUMBER\_MUST\_BE\_FLOAT |
| 13 | Число не введено | «\n» | - | ERR\_NO\_NUMBER |
| 14 | Первое число ноль | +0.E-0 | -.123E+9 | +0.0E+0 |
| 15 | Второе число ноль | -.123E+9 | +0.E-0 | +0.0E+0 |
| 16 | Переполнение порядка при умножении (содержит более 5-ти цифр) | +.9E+99999 | +9.9E+99999 | ERR\_POWER\_OVERFLOW |
| 17 | Перемножение отрицательного и положительного вещественных чисел | +12.3E+3 | -12.E+3 | -0.1476E+9 |
| 18 | Перемножение двух отрицательных чисел | -12.3E+3 | -2.3E+3 | +0.2829E+8 |
| 19 | Перемножение двух положительных чисел | +12.3E+3 | +12.E+3 | +0.1476E+9 |
| 20 | Округление числа, когда 31 разряд меньше 5 | +99999999999999999999999999999.9E+99 | +99999999999999999999999999999.9E+50 | +0.999999999999999999999999999998E+207 |
| 21 | Округление числа, когда 31 разряд больше или равен 5 | +99999999999999999999999999999.9E+100 | +5.E+0 | +0.5E+130 |
| 22 | Когда после нормализации порядок одного из чисел превышает 99999 | +99999999999999999999999999999.9E+99999 | +5.E-500 | +0.5E+99529 |

# **Выводы по проделанной работе**

При написании лабораторной работы я познакомился с длинной арифметикой. Я понял, как располагаются числа в памяти компьютера и как происходит переполнение чисел. В процессе написания программы, научился обходить 10 ограничение языка программирования, создавая свои собственные операции для работы с такими числами.

В своей работе я реализовал возможность перемножения чисел, которые не умещаются в представлении компьютера. Алгоритм перемножения чисел реализован в виде умножения «в столбик».

# **Ответы на вопросы**

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

Диапазон чисел зависит от наличия/отсутствия знака у числа, выбранного типа, отведенной для него памяти, разрядности процессора. Например: знакового короткого целого, под который выделено 2 байта диапазон будет - [−32768, +32767].

Максимально под представление мантиссы отводится 52 разряда, 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.

1. **Какова возможная точность представления чисел, чем она определяется?**

Точность вещественного числа определяется количеством разрядов, отведённых для хранения мантиссы. Например: из 4 байт, выделенных под хранение цифры, 1 (8 разрядов) отдается под данные о порядке и его знаке, а 3 байта (24 разряда) уходят на хранение мантиссы и её знака по тем же принципам, что и для целочисленных значений. Тогда точность будет равна 7 десятичным знакам.

Максимально под представление мантиссы отводится 52 разряда, 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.

1. **Какие стандартные операции возможны над числами?**

Сложение, вычитание, умножение, деление, сравнение.

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

Для представления такого числа программист может разбить число на мантиссу и порядок, на числа/цифры (с помощью массива чисел/символов/строк), отдельно хранить знаки мантиссы и порядка. Каждый из этих элементов должен удовлетворять диапазону представления чисел в ПК.

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

Можно реализовать собственные функции, которые выполняли бы указанные операции, используя представление числа в заранее выбранном формате (см пункт 4).