|  |  |
| --- | --- |
| **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»**

Студент Цветков Иван Алексеевич

Группа ИУ7 – 33Б

*2020 г.*

# **Описание условия задачи**

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.

Десятичное число должно представляться с точкой и экспонентой, а также с обязательным указанием знаков мантиссы и степени, причем возможны подобные варианты записи числа с точкой:

+.00025E+0, +123001.E-5, – 159.456E+5.

Если при перемножении чисел длина мантиссы стала больше 30 знаков, то

необходимо произвести округление – если 31-ый знак больше и равен 5-ти, то число округлется по законам математики, а если 31-ый знак меньше, то он все последующие знаки отбрасываются.

При разработке интерфейса программы следует предусмотреть:

* указание операции, производимой программой,
* указание формата и диапазона вводимых данных,
* указание формата выводимых данных,
* наличие пояснений при выводе результата.

# **Описание технического задания**

**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.

**Обращение к программе:**

Запускается через терминал командой ./app.exe в директории с программой.

**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»
2. Некорректный вывод: в порядке введено не число  
   На выходе сообщение: «ERR\_UNRIGHT\_POWER»
3. Некорректный вывод: порядок превышает число «99999»  
   На выходе сообщение: «ERR\_POWER\_TOO\_BIG»
4. Некорректный вывод: переполнение порядка при умножении  
   На выходе сообщение: «ERR\_POWER\_OVERFLOW»

# **Описание структуры 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.

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

\* MANTISSA\_MAX\_LEN = 32

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**

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

*Описание*: функция совершает чтение вещественного числа и его последующую запись в необохдимые поля структуры number

*Входные значения*: структура *number* для записи в нее числа

*Выходные значения*: структура *number* с записанным в нее числом*; rc –* код ошибки (или ноль при ее отсутствии)

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

*Описание*: функция нормализует мантиссу числа, записанную в *number,* и меняет порядок числа, а также удаляет лишние нули

*Входные значения*: структура *number* для нормализации числа

*Выходные значения*: структура *number* с нормализованным в ней числом

***short int multiply(float\_number num1, float\_number num2, int \*result, int \*res\_power, int \*ind)***

*Описание*: функция перемножает два вещественных числа

*Входные значения*: структуры *num1* и *num2,* числа которых необходимо перемножиить; массив *result,* в который будет записан результат вычислений; *res\_power –* порядок получившегося числа*; ind –* в каком месте массива result находится получившееся число

*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 (илли ноль при его отсутствии)

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

*Описание*: функция округляет полученный результат перемножения

*Входные значения*: массив с получившемся числом после перемножения arr;

ind – индекс начала числа в массиве; end\_arr – индекс конца числа в массиве

*Выходные значения*: округленное число в массиве arr

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

*Описание*: функция печатает на экран результат перемножения двух вещественных чисел

*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

*Выходные значения*: функция ничего не возвращает

# **Набор тестов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Что проверяется | Число №1 | Число №2 | Вывод |
| 1 | Обычный тест | +12.3E+3 | -4.3E+3 | -0.5289E+8 |
| 2 | Не введен знак мантиссы | iu7 | ------- | ERR\_UNRIGHT\_MANTISS\_SIGN |
| 3 | В мантиссе введены неверные символы | -qwerty | -------- | ERR\_UNRIGHT\_MANTISS\_NUMBER |
| 4 | Введена только точка | -.E-00 | --------- | ERR\_ONLY\_POINT |
| 5 | Не введена экспонента | -0.123 | --------- | ERR\_NO\_EPSILON |
| 6 | Порядок содержит недопустимые символы | +12.3E+iu | ---------- | ERR\_UNRIGHT\_POWER |
| 7 | Не введен знак порядка | +12.3E3 | --------- | ERR\_UNRIGHT\_POWER\_SIGN |
| 8 | Мантисса содержит более 30 цифр | +0.99999E-3 (31 девятка) | --------- | ERR\_MANTISSA\_TOO\_LONG |
| 9 | Порядок состоит более, чем из 5 цифр | -.123E+999999 | ------------- | ERR\_POWER\_TOO\_LONG |
| 10 | Введено более 1 точки в мантиссе | -.12.3E+99 | ------------ | ERR\_TOO\_MUCH\_POINTS |
| 11 | Не введена точка в мантиссе | -123E-00 | ------------- | ERR\_NUMBER\_MUST\_BE\_FLOAT |
| 12 | Число не введено | «\n» | ------------ | ERR\_NO\_NUMBER |
| 13 | Первое число ноль | +0.E-0 | -.123E+9 | +0.0E+0 |
| 14 | Второе число ноль | -.123E+9 | +0.E-0 | +0.0E+0 |
| 15 | Переполнение порядка при умножении (содержит более 5-ти цифр) | +.9E+99999 | +9.9E+99999 | ERR\_POWER\_OVERFLOW |
| 16 | Перемножение отрицательного и положительного вещественных чисел | +12.3E+3 | -12.E+3 | -0.1476E+9 |
| 17 | Перемножение двух отрицательных чисел | -12.3E+3 | -2.3E+3 | +0.2829E+8 |
| 18 | Перемножение двух положительных чисел | +12.3E+3 | +12.E+3 | +0.1476E+9 |
| 19 | Округление числа, когда 31 разряд меньше 5 | +99999999999999999999999999999.9E+99 | +99999999999999999999999999999.9E+50 | +0.999999999999999999999999999998E+207 |
| 20 | Округление числа, когда 31 разряд больше или равен 5 | +99999999999999999999999999999.9E+100 | +5.E+0 | +0.5E+130 |
| 21 | Когда после нормализации порядок одного из чисел превышает 99999 | +99999999999999999999999999999.9E+99999 | +5.E-500 | +0.5E+99529 |

# **Контрольные вопросы**

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

Максимальное значение беззаконного целого числа, для которого выделяется 64 разряда, равно 2 в 64 степени (18 446 744 073 709 551 615).

Диапазон чисел зависит от выбранного типа, разрядности процессора и памяти выделенной для хранения числа.

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

Память, выделяемая под хранение мантиссы числа, определяет точность представления вещественных чисел. Для мантиссы числа типа double выделяется 52 бита, с помощью этого мантисса числа может иметь значение до 4 503 599 627 370 496.

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

Операции сложения, вычитания, умножения, деление, взятие остатка, сравнение.

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

Для этого можно использовать массив, в котором каждый элемент будет храниться в своем месте или структуру данных, в которой можно каждую часть числа хранить в определенной переменной

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

Для этого можно воспользоваться библиотечными функциями или собственноручно написать необходимые для решения задачи процедуры

# **Вывод**

При написании лабораторной работы я познакомился с длинной арифметикой. Это помогло мне понять, как же располагаются числа в памяти компьютера, как происходит переполнение чисел и научиться обходить ограничение языка программирования, создавая свои собственные опрации для рабты с такими числами.

В данной рабораторной работе я реализовал возможность перемножения чисел, которые не умещаются в представлении компьютера. Для этого я использовал структуру данных, в которой я смогу хранить все необходимые мне части числа и использвать их по своему желанию. Алгоритм перемножения чисел реализован в виде умножения «в столбик».