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

2021 г.

Оглавление

1. 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 E ±K, где суммарная длина мантиссы (m+n) - до 30 значащих цифр, а величина порядка K - до 5 цифр. Результат выдать в форме ±0.m1 E ±K1, где m1 - до 30 значащих цифр, а K1 - до 5 цифр.

При хранении чисел в оперативной памяти компьютера необходимо обеспечить следующий формат их представления

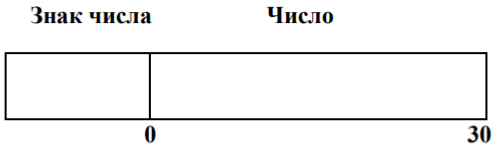

Рис.1. Представление целого числа

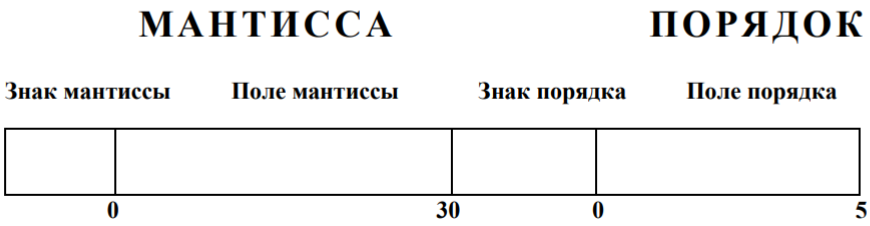

Рис.2. Представление вещественного числа

Десятичное число может представляться без точки: 123, при наличии десятичной точки в числе возможны следующие варианты его представления: .00025, +123001., – 123.456. Также допускается представление числа в экспоненциальной форме: 1234567 E –20, 1234567 E 20 или 123.4567 E23. В программе должна быть реализована возможность ввода чисел в любом из перечисленных представлений. Результат при выдаче на печать должен быть нормализован в виде: знак 0.мантисса E знак порядок.

Если при делении чисел длина мантиссы стала больше 30 знаков, то необходимо произвести округление (если 31-й разряд больше или равен 5, то к 30-му разряду добавляется единица, если меньше 5, то 31-й разряд отбрасывается).

Все логически завершенные фрагменты алгоритма (ввод, вывод, обработка и т.п.) необходимо оформить в виде подпрограмм.

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.

2. Описание ТЗ

1. **Исходные данные**: 2 строки, содержащие два действительных числа вида: [+/-]m.nE[+/-]k, где:
    - Суммарная длина мантиссы m+n – до 30 цифр.
    - Порядка k – до 5 цифр.
    - Допускается отсутствие точек, e, знаков.

2. **Результат**: действительное число в форме ±0.m1 e ±k1, где:
    - m1 - до 30 значащих цифр.
    - k1 - до 5 цифр.
3. **Задачи, реализуемой программой**: произведение действительного числа на действительное число.

3. 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.

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.

4. 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))
2) 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))
3) 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))

4) 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))
5) 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))

1. Функции программы

1) short int read_number(float_number *number)

a) **Описание**: функция совершает чтение вещественного числа и его последующую запись в необходимые поля структуры *number*
b) **Входные значения**: структура *number* для записи в нее числа
c) **Выходные значения**: структура *number* с записанным в нее числом; *rc* – код ошибки (или ноль при ее отсутствии)

2) void normalization(float_number *number)

a) **Описание**: функция нормализует мантиссу числа, записанную в *number*, и меняет порядок числа, а также удаляет лишние нули
b) **Входные значения**: структура *number* для нормализации числа
c) **Выходные значения**: структура *number* с нормализованным в ней числом

3) short int multiply(float_number num1, float_number num2, int *result, int *res_power, int *ind)

a) **Описание**: функция перемножает два вещественных числа
b) **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
c) **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)

4) void check_arr(int *arr, int ind, int end_arr)

a) **Описание**: функция округляет полученный результат перемножения

b) **Входные значения**: массив с получившемся числом после перемножения *arr*; *ind* – индекс начала числа в массиве; *end_arr* – индекс конца числа в массиве

c) **Выходные значения**: округленное число в массиве *arr*

5) ***void print_result(float_number num1, float_number num2, int *result, int res_power, int ind)***

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

b) **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

c) **Выходные значения**: функция ничего не возвращает

2. Аварийные ситуации:

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

   На выходе сообщение: «ERR_UNRIGHT_MANTISS_SIGN»

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

   На выходе сообщение: «ERR_UNRIGHT_MANTISS_NUMBER»

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

   На выходе сообщение: «ERR_NO_EPSILON»

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

   На выходе сообщение: «ERR_TOO_MUCH_POINTS»

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

   На выходе сообщение: «ERR_MANTISSA_TOO_LONG»

6. Некорректный ввод: введена только точка

   На выходе сообщение: «ERR_ONLY_POINT»

7. Некорректный ввод: не введена точка

   На выходе сообщение: «ERR_NUMBER_MUST_BE_FLOAT»

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

   На выходе сообщение: «ERR_NO_NUMBER»

9. Некорректный вывод: не введен знак порядка

   На выходе сообщение: «ERR_UNRIGHT_POWER_SIGN»

10. Некорректный вывод: в порядке введено не число

    На выходе сообщение: «ERR_UNRIGHT_POWER»

11. Некорректный вывод: порядок превышает число «99999»

    На выходе сообщение: «ERR_POWER_TOO_LONG»

12. Некорректный вывод: переполнение порядка при умножении

    На выходе сообщение: «ERR_POWER_OVERFLOW»

5. 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	+0.99999999999999999999999999999998E+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

6. 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».

7. 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.

2) 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.

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?

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.

5) **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).