**Условие задачи**

Составить программу умножения двух чисел, где порядок имеет до 5 знаков: от –99999 до +99999, а мантисса – до 40 знаков.

**Техническое задание**

Смоделировать операцию умножения действительного числа на действительное число в форме ±m.n Е ±K, где суммарная длина мантиссы первого сомножителя (m+n) - до 35 значащих цифр, второго – до 40 значащих цифр, а величина порядка K - до 5 цифр. Результат выдать в форме ±0.m1 Е ±K1, где m1 – до 40 значащих цифр, а K1 - до 5 цифр.

**Входные данные:**

Строка в которой записаны действительные число в форме ±m.n Е ±K, где суммарная длина мантиссы первого сомножителя (m+n) - до 35 значащих цифр, а величина порядка K - до 5 цифр

Строка в которой записаны действительные число в форме ±m.n Е ±K, где суммарная длина мантиссы второго сомножителя (m+n) - до 40 значащих цифр, а величина порядка K - до 5 цифр

**Выходные данные:**

Строка, в которой записан результат умножения в форме ±0.m1 Е ±K1, где m1 – до 40 значащих цифр, а K1 - до 5 цифр

**Возможные аварийные ситуации:**

Некорректный ввод данных, переполнение мантиссы или порядка числа одного из чисел или результата.

**Способ обращения к программе**

В папке с программой вводим команду запуска *./app.exe*

**Структуры данных**

В данной программе используется одна структура данных.

typedef struct

{

int is\_negative;

int mantissa[MAX\_MANTISSA\_LEN];

size\_t mantissa\_len;

int32\_t exponet;

} bignum\_t;

В данной структуре в поле «mantissa» хранятся значащие цифры числа в порядке от младшего разряда к старшему.

Интерфейс для работы со структурой

// Перевод строки в большое число

int str\_to\_bignum(char s[], size\_t len, bignum\_t \*a);

// Перевод большого числа в строку

int bignum\_to\_str(char s[], size\_t \*len, bignum\_t \*a);

// Печать большого числа в консоль

int bignum\_print(bignum\_t \*a);

// Считывание большого числа из консоли

int bignum\_scan(bignum\_t \*a);

// Умножение двух больших чисел

int bignum\_mul(bignum\_t \*a, bignum\_t \*b, bignum\_t \*result);

Используемые константы

#define ERROR\_MANTISSA\_LEN 1

#define ERROR\_EXPONET\_SIZE 2

#define ERROR\_INVALID\_SYMBOL 3

#define ERROR\_EMPTY\_INPUT 4

#define ERROR\_NUM\_LEN 5

#define MAX\_STR\_LEN 100

#define MAX\_MANTISSA\_LEN 40

**Описание алгоритма**

В программе использован классический алгоритм умножения «столбиком». Берется мантисса второго множителя, из которой поочерёдно берутся все цифры. При каждой итерации поочерёдно умножается текущая цифра на все цифры мантиссы первого множителя, при этом результат умножения цифр складывается с данными ячейки временного массива длинна которого в два раза больше мантиссы (номер ячейки вычисляется как номер цифры первого множителя + номер цифры второго множителя). После этого результат сложения кладется в эту же ячейку. Из этого массива после полного прохода и округления будет перенесен результат умножения в структуру описанную выше, отделив только ту часть, которая войдет в мантиссу (то есть старшие разряды).

**Тесты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Первое число** | **Второе число** | | **Результат** | |
| **Некорректный ввод** | | | | |
| 1.0.1E1 | | - | | ОШИБКА в сканирование числа a типа: 3 |
| 1 | | 1.0.1E1 | | ОШИБКА в сканирование числа b типа: 3 |
| 999999999999999999999999999999999999 | | - | | ОШИБКА: мантисса первого числа слишком длинная |
| 1 | | 99999999999999999999999999999999999999999 | | ОШИБКА: мантисса второго числа слишком длинная |
| 1E9999999 | | - | | ОШИБКА в сканирование числа a типа: 2 |
| 1 | | 1E9999999 | | ОШИБКА в сканирование числа b типа: 2 |
| **Граничные значения** | | | | |
| 99999999999999999999999999999999999E99964 | | 1 | | Результат: 0.99999999999999999999999999999999999E99999 |
| 99999999999999999999999999999999999E-99999 | | 1E-35 | | Результат: 0.99999999999999999999999999999999999E-99999 |
| **Умножение на ноль** | | | | |
| 0.1 | | 0 | | Результат: 0E0 |
| 0 | | 100E100 | | Результат: 0E0 |
| **Переполнение порядка** | | | | |
| 100E99998 | | 100 | | ОШИБКА: экспонента результата слишком большая |
| 0.01E-99999 | | 0.1 | | ОШИБКА: экспонента результата слишком большая |
| **Округление** | | | | |
| 99999999999999999999999999999999999 | | 9999999999999999999999999999999999999999 | | Результат: 0.9999999999999999999999999999999999899999E75 |
| 2 | | 9999999999999999999999999999999999999999 | | Результат: 0.2E41 |
| **Разные знаки** | | | | |
| -100 | | 2 | | Результат:  -0.2E3 |
| -10 | | -5 | | Результат: 0.5E2 |

**Выводы по проделанной работе**

Для реализации умножение над числами, выходящими за разрядную сетку персонального компьютера, лучше всего подходит алгоритм умножения “столбиком”, из-за своей простоты реализации. Для хранения и обработки таких чисел лучше всего использовать такую структуру данных как запись (структуру), с полями, содержащими данное число “по частям”: знак (хранится в типе int), мантисса (хранится в массиве типа int) и порядок (хранится в стандартном типе int32\_t).

**Контрольные вопросы**

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

Зависит от разрядности процессора, типа данных, формата хранения и в целом объёма памяти, необходимым для его хранения.

В случае с 64-разрядного процессора не получиться использовать больше 20 десятичных разрядов для представления числа.

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

Точность представления вещественного числа зависит от максимально возможной длины мантиссы. Если длина мантиссы выходит за границы разрядной сетки, то происходит округление. При машинном слове 4 байта (1 бит под знак, 52 двоичных разряда под мантиссу и 11 под порядок). Максимальная точность числа тогда 15 - 16 значащих цифр (2^52).

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?*

Тип, который позволят хранить число по частям или написать свой тип, например, каждая цифра числа которого будет храниться в массиве

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

Хранить число в типе данных, который позволяет производить действия поэлементно (над каждой цифрой)