1. Описание условия задачи

Смоделировать операцию умножения целого числа длиной до 40 десятичных цифр 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 40 значащих цифр, а K1 - до 5 цифр.

2. Техническое задание

Исходные данные и результаты

Исходные данные

- Целое число длиной до 40 дес. цифр, в формате [+-]\d+
- 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

Результаты

- Результат умножения двух входных чисел, представленный в нормальном виде, округленный с точностью до 40 знач. цифр и порядком до 5 цифр. (Если порядок превышает 5 цифр, то результатом будет машинный ноль или бесконечность, в зависимости от знака)

Описание задачи реализуемой программой

- Принять с клавиатуры целое (до 40 значащих дес. цифр) и действительное число (длинна **мантиссы** до 30 **значащих** дес. цифр; длинна порядка до 5 значащих дес. цифр)
- Произвести перемножение этих чисел
- Вывести результат на экран в **нормальном** виде

Способ обращения к программе

Взаимодействие с программой происходит через консольный интерфейс, входные данные вводятся пользователем с клавиатуры Запуск программы из рабочей директории:

```
./app.exe
```

Возможные аварийные ситуации ошибок пользователя

- Некорректный ввод вещественного числа
  - Неверный формат числа
    Примеры: `abc`, `123e`, `+123-2`, `1.23.2`, `12.3e.-2` и тд.

    > Корректный формат числа: `[+-]\d*[.\d+][eE\d+]`
  - Мантисса числа превышает длину в 30 значащих цифр
    Пример: `1.23456789012345678901234567890`

    > Учитываются только **значащие** цифры. То есть число `0.000000000012345678901234567890123456789` будет считаться корректным
  - Порядок числа превышает дилину в 5 цифр
    Пример: `1.12e123456` или `1.12e-123456`
- Некорректный ввод целого числа
  - Неверный формат числа
    Примеры: `abc`, `123.12`, `123a`, `+` и тд.

    > Корректный формат числа: `[+-]\d+`
  - Число состоит более чем из 40 знач. цифр **Пример:** `12345678901234567890123456789012345678901234567890`
- Ошибки при вычислениях
  - Переполнение порядка
    Пример: `10 * .1e99999`

3. Описание внутренних структур данных

`digit_t` - тип данных для представления одной цифры числа

```
typedef unsigned char digit_t;
```

`extended_int_t` - тип данных для "расширенных" целых чисел

```
struct extended_int_t
{
    digit_t digits[EXTENDED_INT_BUFFER_SIZE];
    size_t len;
    bool negative;
}
```

> **EXTENDED_INT_BUFFER_SIZE** - максимальный размер буфера, для хранения цифр, равно **70**

> **digits** - массив, где поразрядно хранятся цифры, образующие число
> **len** - длинна числа
> **negative** - знак числа

`extended_float_t` - тип данных для "расширенных" вещественных чисел

```
struct extended_float_t
{
    extended_int_t mantissa;
    int ord;
}
```

> **mantissa** - мантисса числа
> **ord** - порядок числа

4. Алгоритм

Алгоритм умножения работает по принципу умножения в столбик и работает следующим образом:

1. Целое число и мантисса действительного разворачиваются, для удобства вычислений.
2. Выполняется поразрядное умножение (начиная с меньшего разряда целого) целого числа на мантиссу вещественного. Для каждого следующего разряда результат умножения сдвигается вправо на номер разряда
3. Результат каждого умножения прибавляется к результату вычислений по правилам сложения, то есть с переносом единиц из переполненных разрядов
4. Разница длинны мантисс результирующего числа и изначального действительного прибавляется к порядку результирующего числа
5. Если длинна мантиссы результирующего числа превысила максимальную допустимую длину, то выполняется округление, до максимальной допустимой длинны (по правилам математики)
6. Если после округления разряд переполнился, выполняется перенос единицы, как при сложении
7. Мантисса результирующего числа разворачивается обратно

Сигнатуры используемых функций

Функция сложения целых чисел в развернутом виде

```
return_code add_reversed(const extended_int_t a, const extended_int_t b, extended_int_t *result);
```

Ввод

> a - Первое слагаемое
> b - Второе слагаемое

Вывод

> rc - статус-код
> result - результат сложения (в развернутом виде)

Функция умножения целых чисел в развернутом виде

```
return_code multiply_reversed(const extended_int_t a, const extended_int_t b, extended_int_t *result);
```

Ввод

> a - Первый множитель
> b - Второй множитель

Вывод

> rc - статус-код

> result - результат умножения (в развернутом виде)

Функция умножения действительного числа на целое в развернутом виде

```
return_code multiply_float_reversed(const extended_double_t a, const extended_int_t b, extended_double_t *result);
```

Ввод

> a - Первый множитель (действительное)
> b - Второй множитель (целое)

Вывод

> rc - статус-код
> result - результат умножения (в развернутом виде)

Функция умножения действительного числа на целое

```
return_code multiply_float(const extended_double_t a, cosnt extended_int_t b, extended_double_t *result);
```

Ввод

> a - Первый множитель (действительное)
> b - Второй множитель (целое)

Вывод

> rc - статус-код
> result - результат умножения

5. Тесты

Проверка корректности ввода

- Неверный формат числа

Ввод	Вывод
abc, .1	Error: Invalid integer input
++123, .1	Error: Invalid integer input
123.1, .1	Error: Invalid integer input
1, -0.23.12e2	Error: Invalid double input
1, -0.1	-0.1e-1

- Переполнение целого

Ввод	Вывод
99999999999999999999999999999999999999999, 1	Error: Invalid integer input
99999999999999999999999999999999999999999, 1	0.99999999999999999999999999999999999999999e40

- Переполнение мантиссы действительного

Ввод	Вывод
1, .00000000000000000000000000001	0.1e-29
1, 0000.00000000000000000000000000001	0.1e-29
1, .00000000000000000000000000010000000	0.1e-29
1, .11111111111111111111111111111	Error: limit of meaning numbers is 30
1, .00000000000000000000000000001	Error: limit of meaning numbers is 30

- Переполнение порядка действительного

Ввод	Вывод
1, .1e123456	Error: Order overflow
1, 1e-123456	Error: Order overflow

Проверка корректности операции

- Округление

Ввод	Вывод
11111111111111111111111111111, 99999999999999999999999999999	0.11111111111111111111111111108888888889e60
11111111111111111111111111111, 11111111111111111111111111111	0.12345679012345679012345679012320987654325e59
99999999999999999999999999999999999999999, 11	0.11e42

- Порядок результата больше 5 цифр

Ввод	Вывод
10000, 1e99999	INF
1, .0000001e-99999	ZERO

- Умножение на 0

Ввод	Вывод
0, 1e99999	0
0, -8594e-23	0
1234, 0	0
-45678654, 0	0

- Умножение чисел с разными знаками

Ввод	Вывод
+2, +2	0.4e1
-2, 2	-0.4e1
-2, -2	0.4e1
-0, 100	0

6. Контрольные вопросы

1. Каков возможный диапазон чисел, представляемых в ПК?

В 64-х разрядной машине максимальное количество байт, затрачиваемое на переменную, составляет 64 бита или 8 байт. Таким образом максимальный диапазон значений целого беззнакового числа `от 0 до 2^64 - 1` или `18’446’744’073’709’551’615`, а знакового числа. Если же число знаковое, то диапазон его значений заключен между `-2**63 = –9’223’372’036’854’775’807 - 1` и `2**63 - 1 = 9’223’372’036’854’775’807`.

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

Точность числа определяется количеством знач. цифр мантиссы, которое можно сохранить. То есть, чем больше памяти отводится для хранения мантиссы, тем точнее может быть число. Однако максимальное количество памяти, которое может занимать одна переменная числового типа ограничена разрядностью процессора. Согласно стандарту IEEE 754, наибольшее количество бит, выделенное под мантиссу - 52 (для double). Таким образом, максимальная точность равна log10(2**52) или **15 значащим цифрам**

3. Какие стандартные операции возможны над числами?

Стандартные операции над числами: сложение, вычитание, умножение, деление.

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

Если числа превышают возможный диапазон представления в ПК, то программист может использовать массивы из чисел для представления разрядов числа. Для экономии памяти можно взять массив однобайтовых символов и хранить в нем цифры, тогда каждый индекс массива будет представлять разряд ”длинного” числа.

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

Операции над "длинными" числами можно совершать поразрядно, подобно тому, что мы привыкли делать на бумаге: умножать, делить, складывать и вычитать в столбик

7. Выводы

В случаях, когда требуется повышенная точность, например до 30 знаков, или приходится работать с огромными числами, программисту приходится выбирать структуры данных для реализации, так как мантисса числа выйдет за разрядную сетку компьютера. В этом случае можно использовать поразрядное представление чисел в виде массива цифр, а арифметические операции реализовать аналогично тем, что мы используем на бумаге, в столбик.