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 Романов Семен Константинович

Группа ИУ7 – 35Б

Описание задачи

Смоделировать операцию умножения целого числа длиной до 30 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 30 значащих цифр, а K1 - до 5 цифр.

Техническое Задание

Входные данные:

1. ***Целое число:*** строка, содержащая число в виде ±m. (m<=30 символов)
2. ***Действительное число:*** строка вида ±m.n E ±K. Суммарная длина строки - до 30 значащих цифр (m+n) + символ точки и знака, а величина порядка K - до 5 цифр и символ знака

Выходные данные:

1. Программа выводит либо верный результат в форме ±0.m1 E ±K1, где m1 - до 30 значащих цифр, а K1 - до 5 цифр, либо сообщение о невозможности произвести счет

Способ обращения к программе:

Обращение происходит через консоль

Аварийные ситуации:

1. Ошибка при вводе целого числа

2. Ошибка при вводе действительного числа
3. Превышение длины целого числа
4. Превышение длины действительного числа
5. Превышение длины порядка (на вход)
6. Переполнение длины порядка (после вычислений)

Структуры данных:

1. Для чтения как целого числа, так и действительного, создается массив типа char длиной LEN + 10 (длина мантиссы LEN + место для служебных знаков в виде точки, знака, пробелов, экспоненты)
2. Далее создается такая структура:

```
typedef struct
{
    char number[LEN];
    short int len;
    char sign;
    short int after_dot_len;
    short int order;
} number;
```

Обозначение полей:

- ***char number[LEN]*** – массив для сохранения числа. В данном массиве сохраняются только цифры числа, без точки
- ***short int len*** – количество цифр в number
- ***char sign*** – знак числа. При отсутствии оного во входной строке, записывается плюс
- ***short int after_dot_len*** – количество чисел после запятой (в целом числе это значение равно 0)
- ***short int order*** – значение порядка (в целом числе это значение равен len)

3. Результат сохраняется в структуре result.

Алгоритм:

1. На вход подается 2 числа, сначала целое, затем действительное
2. Идет их обработка в структуры

   Оба числа сохраняются как набор цифр в number без точки и первого нуля в случае десятичного числа, также вычисляется их порядки. В целом числе порядок равен длине числа, в действительном прибавляется (len – after_dot_len). Также в действительном числе убираются первые нули, при их наличии (очистка идет после проверки на длину входных данных), также в этом случае уменьшается порядок

3. Высчитываем верный знак результата
4. Если одно из чисел равно 0, то все результат становится равен 0 и идет переход на шаг 6
5. Идет их перемножение “столбиком” и их запись в во временную строку result_number[LEN * 2 + 1]. Двойная длина нужна для дальнейшего округления числа. Два числа длиной LEN при перемножении не могут дать число, длиной больше чем 2 * LEN. После этого будет произведено округление до 30 разряда, в случае если в конце будут указаны незначащие нули, они будут стерты. Первые нули также убираются с изменением порядка.
6. Далее идет сложение порядков.
7. Вывод корректного результата будет произведено так: printf(“%c 0.%s E%d”, result.sign, result.number, result.order);

Тесты

N	Test	Number 1	Number 2	Result
1	Некорректный ввод	123.1	123	“Invalid Input”
2	Некорректный ввод	123	123.1.1	“Invalid Input”
3	Некорректный ввод	123	123.1 E11.1	“Invalid Input”
4	Некорректный ввод	+-123	123.1 E1	“Invalid Input”
5	Некорректный ввод	123	Qwerty	“Invalid Input”
6	Превышение длины порядка	123	123 E999999	“Invalid Input”
7	Переполнение порядка	123	123 E99999	“Order Overflow”
8	Переполнение порядка	1	0.01 E-99999	“Order Overflow”
9	Превышение длины мантиссы	111…111(31 единица)	123.1	“Invalid Input”
10	Превышение длины мантиссы	123	0.111…111(30 единиц)	“Invalid Input”
11	Умножение на ноль	0	-123.1	-0.0 E 0

12	Умножение на ноль	123	0.0 E0	0.0 E 0
13	Округление	11	0.111…111(29 единиц)	0.1233…32 (27 троек) E 2
14	Округление	111	0.555…555(29 пятерок)	0.6166…61 (27 шестерок) E 2
15	Округление	999…999(30 девяток)	2	+0.2 E 31

Контрольные вопросы:

1. **Каков возможный диапазон чисел, представляемых в ПК?** На возможный диапазон чисел влияет их тип, размер выделенной для их хранения память, разрядность ОС. Так для целого числа выделяется 16 бит, то есть его максимальное значение – ±2,147,483,647 (int).
2. **Какова возможная точность представления чисел, чем она определяется?** Точность представления вещественных чисел зависит от количества памяти, выделенной для хранения мантиссы. Для мантиссы типа double выделяется 52 бита, то есть мантисса может принимать значения до 4 503 599 627 370 496, а под представление порядка – 11 бит
3. **Какие стандартные операции возможны над числами?** Сложение, вычитание, умножение, деление, сравнение, взятия остатка от деления.
4. **Какой тип данных может выбрать программист, если обрабатываемые числа превышают возможный диапазон представления чисел в ПК?** Массив символов для обычного

представления числа или структуру, в которую можно разделить различные данные (знак, мантиссу, порядок).

5. **Как можно осуществить операции над числами, выходящими за рамки машинного представления?** Разработать собственные решения, либо же использовать специальные библиотеки.

Вывод

В ходе лабораторной работы было установлено, что при работе с длинной арифметикой крайне эффективным оказался способ представления числа в качестве массива символьного типа. Несмотря на проигрыши по скорости, мы получили выигрыш по памяти, а также возможность умножения чисел любой длины, даже превышающих 30 символов, как было установлено в ходе работы. Также мной была создана функция, которая может умножать не только целые и действительные вместе, но и отдельно 2 целых или 2 действительных.