МГТУ им. Н.Э. Баумана

**Дисциплина Типы и Структуры данных.**

**Лабораторный практикум №1**

**по теме: «**Обработка больших чисел**»**

Работу выполнил:

студент группы ИУ7-35Б

Прянишников Александр

**Цель работы**: реализация арифметических операций над числами, выходящими за разрядную сетку персонального компьютера, выбор необходимых типов данных для хранения и обработки указанных чисел.

# Условие задачи. Вариант 2

Смоделировать операцию умножения действительного числа в форме ±m.n Е ±K, где суммарная длина мантиссы (m+n) - до 30 значащих цифр, а величина порядка K - до 5 цифр, на целое число длиной до 30 десятичных цифр. Результат выдать в форме ±0.m1 Е ±K1, где m1 - до 30 значащих цифр, а K1 - до 5 цифр

# Требования к задаче

1. **Исходные данные и результат.**

**Ввод.**

С консоли вводится вещественное число, до его обработки в структуру оно хранится в строке. Максимальная длина мантиссы – 30, максимальная длина порядка – 5. При вводе целого числа программа продолжает свое выполнение, в дальнейшем нормализуя его. При некорректном вводе программа завершает работу и выдает сообщение об ошибке.

*Примеры корректного ввода:*

* -123239
* +12345678932626722736826628
* 0.34567
* .32456E24
* –0.456E–24
* 0.0

Далее с консоли вводится целое число, до его обработки в структуру оно хранится в строке, максимальная длина – 30 знаков. Программа в дальнейшем нормализует это число. При некорректном вводе программа завершает работу и выдает сообщение об ошибке.

*Примеры корректного ввода:*

* -13537
* +338
* 0

**Вывод.**

Выводится нормализованное произведение в формате (+/-)0.m E(+/-)K, где m – максимальная длина мантиссы – 30, K – максимальная длина порядка – 5.

*Примеры вывода:*

* -12.3673423E20

1. **Описание задачи, реализуемой программой**

Программа получает на вход два числа: вещественное и целое. Далее она нормализует их, выполняет умножение одного на другое, затем нормализирует результат на экран и выводит его.

1. **Способ обращения к программе.**

Программу можно запустить через MSYS либо через QT Creator и любую среду разработки.

Выполняется следующая последовательность команд:

* *gcc –std=c99 –Wall –Werror –pedantic –c \*.c*
* *gcc –o main.exe \*.o*
* *./main.exe*

1. **Описание возможных аварийных ситуаций и ошибок пользователя.**

**Ошибки пользователя при вводе вещественного числа.**

* Использование посторонних символов  
  *Пример*: 123ewe34
* Некорректный ввод числа  
  *Пример: –*1–23.44
* Слишком длинный порядок числа  
  *Пример*: 1E999999
* Слишком длинная мантисса  
  *Пример*: 1234567890123456789012345678990

**Ошибки пользователя при вводе целого**

* Использование посторонних символов  
  *Пример*: 123ewe34
* Слишком длинная мантисса  
  *Пример*: 1234567890123456789012345678990
* Некорректный ввод числа  
  *Пример: –*1–2344
* Введено вещественное число  
  *Пример*: 1.32

# Описание внутренних структур данных.

Для решения данной задачи мной был выбран вариант хранения длинных чисел в виде структуры по нескольким причинам:

1. Вещественное число разделено на три части: знак мантиссы, мантисса и порядок. При таком разделении для восприятия более удобно отдельное хранение каждой части.
2. Для порядка не требуется отдельная переменная для хранения знака, так как тип данных int прекрасно справляется с любым значением.

Для целого и вещественного числа были составлены две различные структуры. Также отдельная структуры была сделана для хранения результата.

Вот так описаны структуры:

struct **real**

{

char mantiss\_sigh; Знак мантиссы

int mantiss[MAX\_LEN\_MANTISS]; Массив цифр в мантиссе

int order; Значение порядка

};

struct **integer**

{

char mantiss\_sigh; Знак чисоа

int mantiss[MAX\_LEN\_MANTISS]; Массив цифр числа

};

struct **result**

{

char result\_sigh; Знак мантиссы

int mantiss[2 \* MAX\_LEN\_MANTISS + 1]; Массив цифр в результате

};

# Описание алгоритма

Сначала программа должна обеспечить корректный ввод данных. Используется функция gets для вещественного числа, которое вводится первым. Предполагается, что пользователь может ввести знак порядка через пробел, поэтому для таких случаев предусмотрен gets. До переноса числа в структуру оно хранится в строке. После ввода из введённого числа убираются незначащие нули впереди, затем программа переходит к этапу проверки ввода на корректность. В этот момент число представлено в символьном массиве

Существует несколько вариантов ввода. Чтобы правильно проверить данные, программа анализирует каждый символ, подсчитывает количество знаков “–“, “+”, “E”, “.”, их нахождение в числе. Если в числе отсутствуют посторонние знаки, все символы соответствуют ТЗ, длина мантиссы не больше 30 знаков, то программа переходит к вводу и проверке целого числа.

Целое число вводится с помощью функции scanf, так как в нём не может быть пробелов. Проверка на валидность заключается в анализе цифр и символов “–“ и “+”.

Следующий этап работы программы – парсинг полученных чисел и представление их в нормализованном виде. Для вещественного числа алгоритм такой:

* Все цифры до знака порядка “E” отправляются в массив мантиссы, при этом к порядку числа добавляется количество цифр, которые находятся перед дробной частью.
* Знак числа передаётся в символ структуры, отвечающий за знак.
* Пользователь также указывает порядок числа через “E”. Эта часть преобразовывается в int и передаётся в структуру.

Для целого числа всё проще: знак передаётся в закреплённый для этого в структуре символ, а все цифры заносятся в массив.

Умножение происходит по аналогии с умножением столбиком: каждая цифра целого числа перемножается на каждую цифру вещественного числа, после чего каждый раз записывается ответ в соответствующий элемент результирующего массива. Знак в результирующей структуре выбирается из математических правил и знаков в структурах целого и вещественного чисел.

Затем полученная структура нормализуется. Если в полученном результате значащих цифр больше 30, то 30–я цифра округляется по математическим правилам, а все остальные отбрасываются. Программа высчитывает порядок получившего числа, после чего все данные переносит в структуру вещественного числа, в которой будет храниться ответ.

В конце выводится результат согласно ТЗ. В случае ошибки программа сразу же приостанавливает свою работу.

# Тесты

Негативные тесты:

|  |  |
| --- | --- |
| **123.345fd** | **Буквы в вещественном числе** |
| **123.45 34f** | **Буквы в целом числе** |
| **–213.45E+234** | **Знак + в порядке недопустим по ТЗ** |
| **.0021300 333.23** | **Целое введено как вещественное** |
| **12345678901234567890123456878901** | **Слишком длинная мантисса** |
| **12E99999 20** | **Порядок результата превышен** |
| **1E999999** | **Слишком длинный порядок** |
|  | **Введена пустая строка** |

Положительные тесты:

|  |  |
| --- | --- |
| **12 12** | **0.144E3** |
| **0.12 12** | **0.144E1** |
| **0.0 23** | **0E0** |
| **.2345 5467** | **0.12820115E4** |
| **456.34E2 1** | **0.45634E3** |
| **–31.43E–5 -24** | **0.75432E–3** |
| **999999999999999999999999999999 2** | **0.2E31** |
| **0.1E–99999 100** | **0.1E-99997** |

# Оценка эффективности

Программа выполняет вычисления меньше, чем за одну секунду. Количество проведённых операций не так много: проверка на валидность, парсинг чисел осуществляется за один цикл, а перемножение – за два цикла. Итоговая зависимость O(N) ~ N2, где N – максимальная длина мантиссы. Алгоритм достаточно эффективен.

# Выводы по проделанной работе

Для выполнения длинных вычислений программисту требуется самому реализовывать обработку и хранение данных. Для этого нужно создавать структуры для различных элементов, а действия реализовывать через математические правила и школьными методами вычислений (столбик). Также я познакомился с нормализованными числами, освоил алгоритмы их обработки и вывода.

# Ответы на контрольные вопросы

1. Каков возможный диапазон чисел, представляемых в ПК?

Для целых чисел зависит от количества выделенных разрядов. Для 64-разрядного процессора невозможно использовать больше 20 десятичных разрядов для представления числа, так как 264 примерно равно 18 квантимиллиардам, но этого не всегда достаточно для научных вычислений.

Для вещественных чисел 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?

Если десятичная точка расположена в мантиссе перед первой значащей цифрой числа, то при фиксированном количестве разрядов, отведённых под мантиссу, 2 обеспечивается возможность сохранить максимальное количество значащих цифр, то есть обеспечить максимальную точность представления числа в ПК.

Длина мантиссы определяет точность представления числа, а длина порядка ограничивает диапазон значений. При этом, если мантисса выходит за разрядную сетку ПК, то происходит ее округление. При выходе за заданный диапазон величины порядка могут возникнуть проблемы с переполнением порядка.

Если рассматривать встроенные типы данных, то возможная точность определяется типом системы, а также выбранным программистом типом данных. Но в случае, если программист самостоятельно реализовывает длинную арифметику, то точность зависит только от выбранного числа разрядов в мантиссе и порядком.

3. Какие стандартные операции возможны над числами?

Все классические из школьной программы: сложение, вычитание, деление и умножение, а также их производные (возведение в квадрат, выраженное через умножение на само себя).

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

Программист может создать структуру, которая будет состоять из символа знака числа, массива цифр в мантиссе и порядка числа.

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

Операции нужно выполнять с помощью стандартных школьных алгоритмов деления, умножения, сложения и вычитания в столбик.