**Заколесник Максим ИУ7-33**

**Лабораторная работа №1**

Обработка больших чисел

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

**Задание (Вариант 4)**:

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 деления целого числа длиной до 30 десятичных цифр на действительное число в формате +-m.n E +-K где суммарная длина мантиссы (m+n) до 30 значащих цифр, а величина порядка К — до 5 цифр. Результат выдать в форме ± 0.m1 E K1, где m1 — до 30 значащих цифр, а K1 — до 5 цифр.

**Входные данные:**

На вход подается 2 числа в десятичной системе счисления:

1. Целое число длиной до 30 десятичных цифр. Знак “+” можно вводить или не вводить, однако знак “-”, если используется, то вводить обязательно. Ведущие нули тоже учитываются при проверке превышения лимита в 30 цифр. Допускаются пробелы между цифрами. (программа самостоятельно производит склейку числа). Ввод любых других символов является недопустимым.

2. Вещественное число. Формат ввода: +-m.n E +-K. Мантисса длиной до 30 десятичных цифр. Знак “+” можно вводить или не вводить, однако знак “-”, если используется, то вводить обязательно. Ведущие нули тоже учитываются при проверке превышения лимита в 30 цифр. Допускаются пробелы между цифрами. (программа самостоятельно производит склейку числа). Ввод любых других символов является недопустимым. Вводить точку не обязательно, но ее можно ввести лишь один раз. Можно ввести E/e (или не вводить). Порядок — целое число (с знаком или без, но одним). По модулю K не превышает 99999. Если введено e/E после этого обязательно нужно ввести число.

**Выходные данные:**

Результат выдается в форме ± 0.m1 E K1, где m1 — до 30 значащих цифр, а K1 — до 5 цифр.

Если делитель равен нулю, программа выдает соответствующее сообщение.

**Возможные ошибки:**

1. Несоответствие формату входных данных (выдает сообщение об ошибке и завершается)

2. Переполнение результата, что привело к невозможности его представить в указанном формате. (сообщение об ошибке и завершается без вывода результата)

**Обращение к программе:**

Через консоль

**Внутренняя структура данных:**

Большие числа представлены в виде двух структур: MAX\_COUNT = 30

**struct** MyDOUBLE {

**MyDOUBLE**(**const** **int**& size) {

data.resize(size, 0);

}

**MyDOUBLE**(){

data.resize(MAX\_COUNT, 0);

}

**int** count = 0;

vector<**int**> data;

**int** exp = 0;

**bool** sign = **false**;

**int** error\_input = 0;

};

**struct** MyINT {

**MyINT**(**const** **int**& size) {

data.resize(size, 0);

}

**MyINT**(){

data.resize(MAX\_COUNT, 0);

}

**int** count = 0;

vector<**int**> data;

**bool** sign = **false**;

**int** error\_input = 0;

};

Тут *count* - количество цифр, важных для нас, хранящихся в массиве data,

*data* - массив, состоящий из цифр числа

*sign* - число, отвечающие за знак большого числа (1 ~ «+», -1 ~ «-» )

*error\_input* - хранит код ошибки.

*exp* - порядок вещественного числа (характерно только для структуры MyDOUBLE)

**Описание алгоритма:**

1. Считывание делимого и делителя. Проверка правильности ввода. Формирование и запись данных в структуру.

2. Приведение целого числа к вещественному типу

3. Деление чисел, реализованных на основе алгоритмов длинной арифметики (потребуется реализация сравнение (<=), разности больших чисел, а также умножения большого на маленькое). Также дополнительно реализуется функция увеличения большого числа на один разряд для реализации деления в столбик.

4. Нормализация результата вычислений и его округление. Проверка возможности представления числа в указанном формате выхода.

5. Печать результата на экран.

Основные функции, реализованные для выполнения алгоритма:

Для структуры MyDOUBLE реализованы функции:

**void** extend(); - расширяет число на один младший разряд разряд.

**void** updateAsDividend(); - добавляется незначимые нули к делимому

**void** normalize(); - нормализирует результат для его вывода

**Перезагрузка операторов**

**MyDOUBLE operator**\*(**const** MyDOUBLE &a, **const int** b); - функция умножения большого числа на стандартное целое

**MyDOUBLE** **operator**-(**const** MyDOUBLE &a, **const** MyDOUBLE &b); - функция разности двух больших чисел

**MyDOUBLE operator**/(**const** BigInt &a, **const** BigDouble &b); - функция деления двух больших чисел

**bool operator** <=(**const** BigDouble &a, **const** BigDouble &b); - оператор сравнение больших чисел

ostream& **operator**<< (ostream &stream, **const** BigDouble &obj); - оператор вывода большого числа

istream& **operator**>> (istream &stream, BigDouble &obj); - оператор ввода большого числа

Для структуры MyINT реализован только оператор ввода и конструктор.

**Оценка эффективности работы программы:**

Программа не измерялась на эффективность.

**Оценка эффективности представления чисел в программе:**

Для реализации алгоритмов длинной арифметики наиболее рационально выбрать представление больших чисел в виде структуры, содержащей собственное цифры числа в виде массива, а также дополнительную информацию о числе, необходимую для его корректной обработки. В массиве числа хранятся в обратном порядке, чтобы младший разряд числа соответствовал младшему индексу. С одной стороны, алгоритм увеличивает время работы за счет использования функции переворота массива, но с другой стороны, это снижает затраты на добавление старшего разряда в случае переноса при сложении или умножении чисел. Также, это повышает читабельность кода, поскольку операции в столбик происходят от младших разрядов, а следовательно и от младших индексов массива.

**Тесты:**

Проверка некорректного ввода чисел

|  |  |
| --- | --- |
| 7h | Error Input(INT) |
| 9999999999999999999999999999990 | Error Input |
| 1  1ee | Error Input(DOUBLE) |
| 5e3 | Error Input(INT) |
| +6+ | Error Input(INT) |
| 1  222.25. | Error Input(DOUBLE) |

Проверка деления (деление на ноль, ведущие нули, выход за пределы порядка, округление)

|  |  |
| --- | --- |
| 1  000000.1 | +0.100000000000000000000000000000 E 2 |
| 1  0.00000 e 25 | **Devided by zero** |
| 00000  5 | +0.000000000000000000000000000000 E -50 |
| 999999999999999999999999999999  0.000000001 e -99999 | **Out of range** |
| 1  99999999999 e +99999 | +0.000000000000000000000000000000 E -99999 |
| 2  3 | +0.666666666666666666666666666667 E 0 |
| 3256  53.68 e -4 | +0.606557377049180327868852459016 E 6 |
| 999999999999999999999999999999  5e-5 | +0.200000000000000000000000000000 E 35 |

**Теоретическая часть:**

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

Диапазон значений чисел зависит от размера области памяти, выделяемой под хранение переменной этого типа, от наличия знака в числе и от типа представления числа (целое или вещественное).

**2.Какова возможная точность представления чисел?**

Точность представления вещественного числа зависит от максимально возможной длины мантиссы, которая зависит от области выделяемой памяти и наличия знака. Если длина мантиссы выходит за границы разрядной сетки, то происходит округление.

**3.Какие стандартные операции возможны над числами?**

Сложение, вычитание, умножение, деление.

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

Можно воспользоваться типами данных из дополнительно подключаемых библиотек, предназначенных для работы с большими числами.

Также большие числа можно хранить в виде структуры, в которой можно поразрядно хранить число в массиве, а также информацию о знаке, количестве цифр и т.д..

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

Если это операция деления или умножения, то в таком случае операции проводятся поразрядно, как при вычислениях столбиком.

**Вывод:**

Совершил работу с struct в C++, научился перезагружать операторы. Выбрал, на свой взгляд, удачный тип для очень длинных чисел.

Выбрал этот язык для дальнейшего его изучения.