**Отчет**

**По первой лабораторной работе**

**“Обработка больших чисел”**

**Вариант 1**

**Подготовил студент группы ИУ7-32Б**

**Блохин Дмитрий**

**Условие**

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

**Техническое задание**

Задача программы:

Реализовать деление целого числа на вещественное. Размерность чисел не позволяет произвести операцию, используя встроенные типы языка СИ и операции над ними.

**Входные данные:**

1. Строка, содержащая целое число в виде [-]LE[-]K, где L - длина, до 30 цифр, K - величина порядка. Допускается отсутствие знака экспоненты и порядка. Наличие пробелов не допускается!
2. Строка, содержащая вещественное число, в виде [-]LE[-]K, где L - суммарная длина мантиссы (m + n), до 30 цифр, K - величина порядка. Допускается отсутствие вещественной точки, отсутствие знака экспоненты, порядка. Наличие пробелов не допускается!

Выходные данные:

Строка, имитирующая “большое число” в нормализованном виде [-]0.LE[i]K, где L - длина до 30 цифр, K - величина порядка, до 5 цифр.

Функция программы:

Деление целого числа на вещественное Если мантисса деления выходит за 30 значащих цифр, то произвести округление.

Аварийные ситуации

Некорректный ввод строки с числом

Вход: Строка, включающая в себя символы, отличные то “-”, “.”, “E”

Выход: “Некорректный ввод!”

Пустой ввод

Вход: “\n”

Выход: “Пустой ввод!”

Порядок введенного числа по модулю больше, чем 99999

Вход: “1E100000”

Выход: “Слишком большая степень числа!”

Деление на 0

Вход: “123\n0”

Выход: “Делить на ноль нельзя!”

Используемые структуры

num - Служит для хранения большого числа

typedef struct

{

char mant\_sign; // Знак мантиссы (0 для +, 1 для -)

int mant[70]; // Мантисса

int mant\_len; // Длина мантиссы

int power; // Порядок

} num;

Размер: 1 + 4\*70 + 4 + 4 = **285 байт**

stream - Массив, в котором хранятся символы из потока ввода. Служит для последующей обработки входных данных.

char stream[100];

Алгоритм

Прежде всего необходимо считать символы из потока ввода с помощью функции input(). При неудачном вводе программа выводит соответствующее сообщение и аварийно завершается.

int input(num\* number)

Параллельно с проверкой входных данных происходит заполнение структуры num.

Если входные данные валидны, далее происходит деление двух чисел с помощью функции div(). На вход, помимо делимого и делителя, подаётся также и частное, для хранения результата деления. Вкратце алгоритм можно описать так: Вычитаем поразрядно пока есть возможность, иначе переходим на следующие разряды, количество вычитаний запоминаем.

int div(num\* number1, num\* number2, num\* div\_res)

С помощью вспомогательной ф-ии bigger() выясняется, нужно ли сносить цифру из младших разрядов.

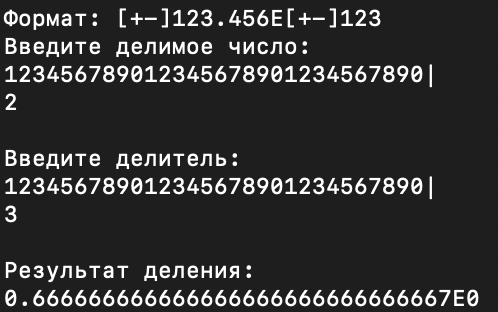
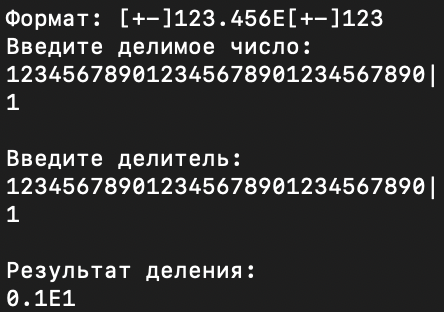
int is\_bigger(num\* num\_cpy, num\* number2, int div\_start, int div\_end)

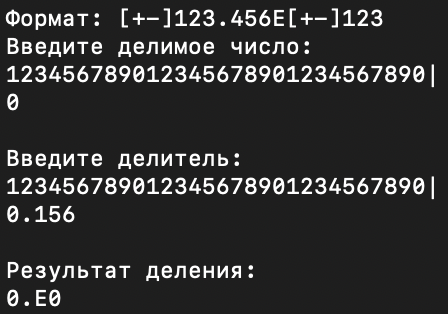
С помощью вспомогательной ф-ии before() происходит проверка на возможность продолжения деления (Можем ли занять у старших разрядов?).

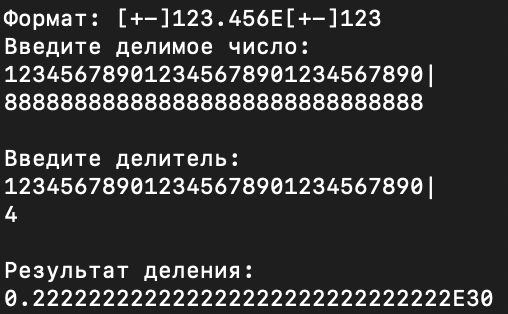
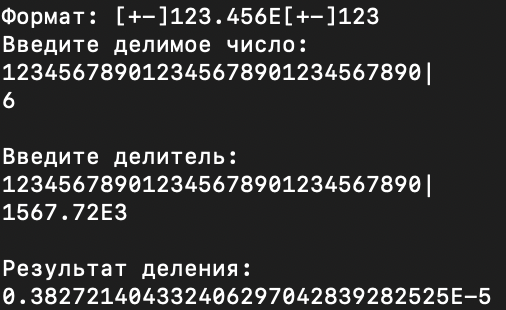
int before(num\* num\_cpy, int i)

**Тесты**

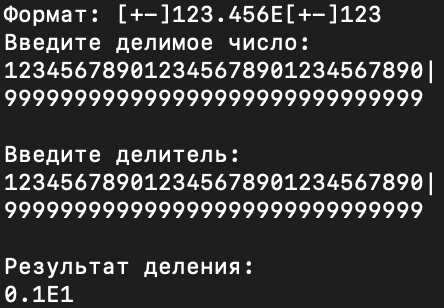
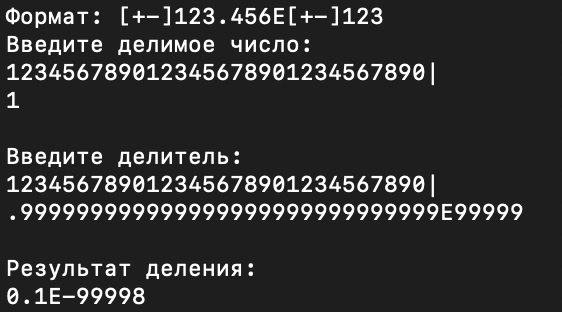
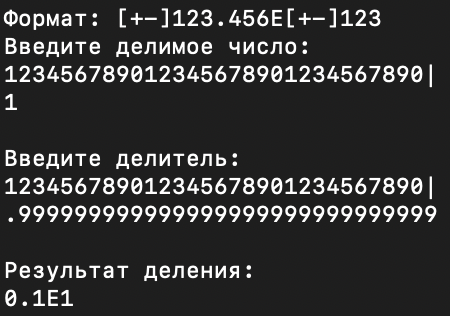
**Тепличные тесты:**

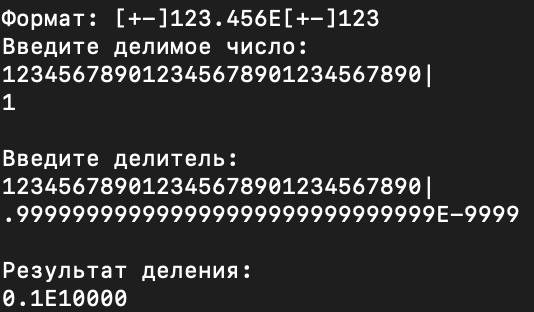
****

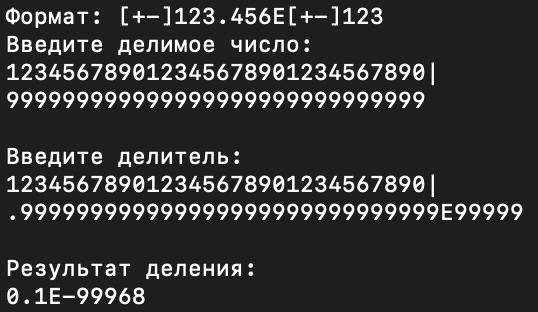
****

****

**Экстремальные тесты:**

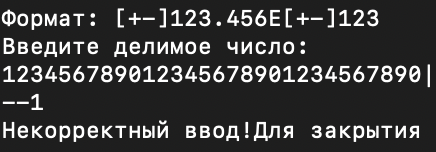
****

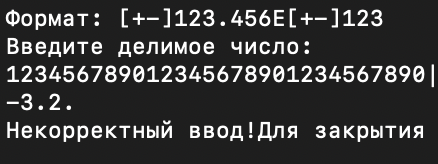
****

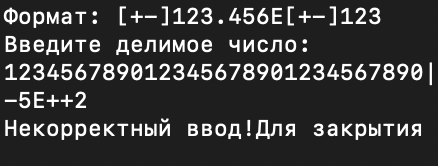
****

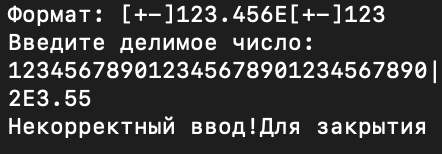
**Запредельные тесты:**

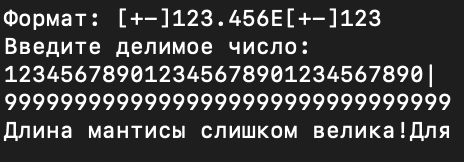
**Неправильный ввод**

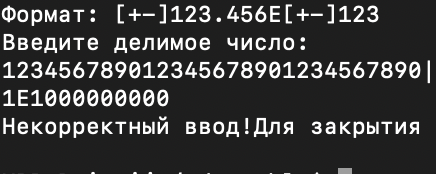
****

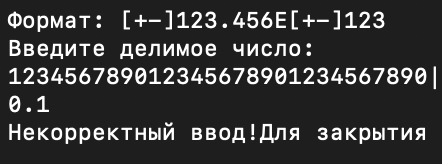
****

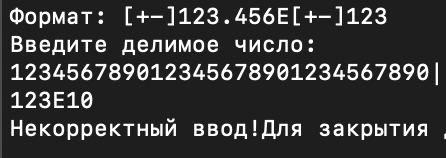
****

****

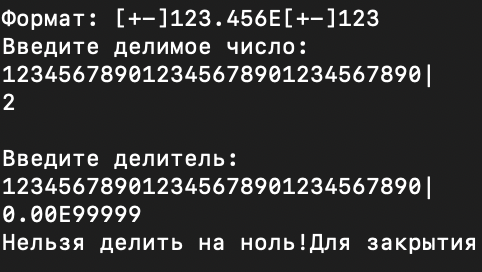
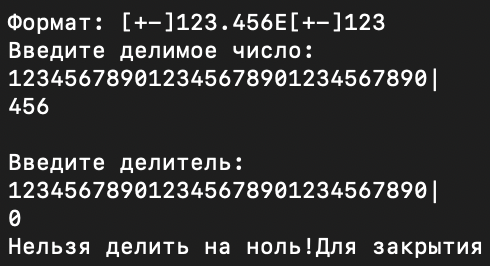
****

****

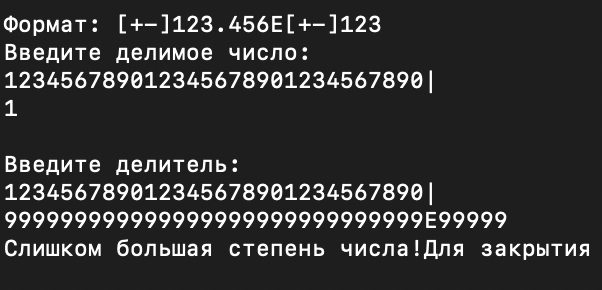
****

****

**Деление на 0:**

****

**Превышение допустимого размера степени:**

****

**Оценка эффективности**

Концепцией реализованного алгоритма является разделение числа на две части: мантиссу и порядок. После чего происходит деление двух чисел, с помощью алгоритма “деления в столбик”, вычитание порядков чисел.

**Вывод**

В процессе выполнения Л/Р я ознакомился с обработкой чисел, размер которых не умещается в стандартные размеры типов языка C. Опираясь на статические СД, я имитировал работу с числами, разработал алгоритм для решения поставленной цели. Для реализации я выбрал одну СД - Запись, в которой хранится знак, мантисса и порядок числа.

**Ответы на вопросы.**

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

Возможный диапазон чисел напрямую зависит от объёма памяти, выделяемый под хранение конкретного типа, наличия знака, а также представления числа (вещественное, целое).Например, если целая беззнаковая переменная типа integer занимает 4 байта (32 бит), то диапазон её значений: [0, 2^32-1] = [0, 4294967295]. Если же она может принимать также и отрицательные значения, то диапазон: [-2^31, 2^31 -1]

**2 Какова возможная точность представления чисел?**

Вещественные числа хранятся в представлении с плавающей точкой в виде:

X = M \* E^p, где М – мантисса, представленная правильной дробью в интервале [0.1..1). Точность представления вещественного числа зависит от максимально возможной длины мантиссы, которая в свою очередь зависит от области выделяемой памяти и наличия знака. Если длина мантиссы выходит за границы разрядной сетки, то происходит округление.Максимально под представление мантиссы отводится 52 знака , а под представление порядка 11 знаков.

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

Стандартные операции над числами – сложение, вычитание, умножение, деление, остаток от деления, целочисленное деление

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

Для представления чисел, превышающих возможный диапазон, может быть использован массив int-ов как имитация цифр – один из видов представления больших чисел как целых, так и вещественных.

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

Операции над большими числами можно выполнять путём последовательного выполнения операций над всеми цифрами, применяя соответствующие алгоритмы.