|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 Артемьев Илья Олегович

Группа ИУ7 – 33Б

*2020 г.*

# **Описание условия задачи**

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.

Десятичное число всегда представляется с точкой и знаком экспоненты “E”. Возможны следующие варианты его представления: +.1E+0, +0.1E+0

Если при делении чисел длина мантиссы стала больше 30 знаков, то

необходимо произвести округление (если 31-й разряд больше или равен 5,

то к 30-му разряду добавляется единица, если меньше 5, то 31-й разряд

отбрасывается).

При разработке интерфейса программы следует предусмотреть:

* указание операции, производимой программой,
* указание формата и диапазона вводимых данных,
* указание формата выводимых данных,
* наличие пояснений при выводе результата.

# **Описание технического задания**

Входные данные:

Целое число: строка, содержащая целое число в виде +\- m. Знак перед числом обязательно. Длина модуля числа m - до 30 цифр.

Действительное число: строка, содержащая вещественное число в виде +\-m.nE+\-K. Знак перед числом и перед порядком обязательно вводить. Символ экспоненты E обязательно вводить. Суммарная длина m+n - до 31 цифры, включая точку; длина порядка — до 5 цифр.

Выходные данные:

Длинное число в виде +\-0.m1E+\-K1. Длинна мантиссы m1 - до 30 цифр; длинна порядка K1 — до 5 цифр.

Действие программы:

Деление целого числа на вещественное.

Обращение к программе:

Запускается через терминал: ./main.exe

Аварийные ситуации:

1. Некорректный ввод: строка с целым числом содержит символы, которые не цифра и не (+\-), если это не нулевой элемент строки.

На выходе сообщение: «Проверьте число на предмет записи лишних символов»

1. Некорректный ввод: строка с вещественным числом содержит символ не цифру и не символ из набора („+“, „-“, „ . “, „E“).

На выходе сообщение: «Проверьте число на предмет записи лишних символов»

1. Некорректный ввод: переполнение порядка при вводе вещественного числа. (порядок превышает по модулю 99999)

На выходе сообщение: «Проверьте количество символов в порядке числа»

1. Некорректный ввод: превышение длины при вводе целого числа (больше 30 цифр).

На выходе: «Превышено допустимое количество символов в целом числе»

1. Некорректный ввод: превышение длины при вводе вещественного числа (больше 31 цифры, включая точку)

На выходе сообщение: «Проверьте количество символов в мантиссе числа»

1. Некорректный ввод: целое или вещественное число без знака (+\-).

На выходе сообщение: «Проверьте запись следующий символов: + - . E»

1. Деление на нуль: при вводе вещественного числа введен нуль.

На выходе сообщение: «Деление на нуль запрещено»

1. Некорректный ввод: введена пустая строка (т. е. просто введен знак „\n’ ).

На выходе сообщение: «Проверьте запись следующий символов: + - . E»

1. Некорректный вывод: переполнение порядка.  
   На выходе сообщение: «Произошло переполнение порядка»
2. Некорректный вывод: машинный нуль.  
   На выходе сообщение: «Машинный нуль»

# **Описание структуры данных**

После ввода числа, оно хранится в массиве символов длинной 42 (с учетом всех служебных знаков: точка “.” , знак экспоненты “E”, знак числа и знак порядка “+\-“).

Далее число обрабатывается и записывается в структуру description.

Структура description:

**typedef struct**

**{**

**char signs[SIGNS\_LEN];**

**char mantissa[MANTISSA\_LEN\_STRUCTURE];**

**int order;**

**} description;**

Поля структуры:

**signs[SIGNS\_LEN]** – хранит два знака: 0 – ой индекс – знак действительного числа, 1 – ый индекс – знак порядка; **SIGNS\_LEN = 3**

**mantissa[MANTISSA\_LEN\_STRUCTURE]** – хранит мантиссу вещественного числа; **MANTISSA\_LEN\_STRUCTURE = 35**

**order** – хранит порядок вещественного числа

# **Описание алгоритма**

1. Программа считывает две строки, одна содержит целое число, другая — вещественное.
2. Проводится проверка на все возможные ошибки с помощью специальной функции.
3. Строки обрабатываются и записываются в структуру description.
4. Вещественное число приводится к виду целого числа, учитывая порядок.
5. Если все данные верны, то происходит деление первого (целого) числа на второе (вещественное) по методу «деление в столбик» с контролем округления.
6. После деления результат выводится в нормализованном виде в соответствии со спецификацией, указанной в ТЗ (+\-0.m1E+\-K1).

# **Набор тестов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Что проверяется | Целое число | Вещественное число | Вывод |
| 1 | Деление на нуль | +12 | +0.E+0 | Деление на нуль запрещено |
| 2 | Деление нуля на число | +0 | -232.12E+3 | +0.0E+0 |
| 3 | Деление нуля на нуль | +0 | +0.E+0 | Деление на нуль запрещено |
| 4 | Округление | +2 | +3.E+0 | +0.666..6667E+0 |
| 5 | Деление чисел разных знаков | +12345 | -123.45E+0 | -0.1E+3 |
| 6 | Деление целых чисел | +123456 | +111.E+0 | +0.111221621621621621621621621622E+4 |
| 7 | Граничные значения (целое число) | +999999 (30 девяток) | +1.E+0 | +0.999999999999999999999999999999E+30 |
| 8 | Граничные значения (вещественное число) | +123 | +9999...999.E+0 | +0.123E-27 |
| 9 | Граничные значения (порядок) | +123 | +123.E+99999 | +0.1E-99998 |
| 10 | Превышение длины мантиссы (вещественное число) | +1 | +0.99999 (31 девятка).E+0 | Проверьте количество символов в мантиссе числа |
| 11 | Превышение длины целого числа | +999999 (31 девятка) | - | Превышено допустимое количество символов в целом числе |
| 12 | Превышение длины порядка (вещественное число) | +1 | +1.E+999999 | Проверьте количество символов в порядке числа |
| 13 | Некорректный ввод (буква вместо цифры) | +a | - | Проверьте число на предмет записи лишних символов |
| 14 | Некорректный ввод (без знака) | 1 | - | Проверьте запись следующий символов: + - . E |
| 15 | Некорректный ввод (дробное вместо целого) | +123.34 | - | Проверьте запись следующий символов: + - . E |
| 16 | Некорректный ввод (дробное вместо целого) | +123E+45 | - | Проверьте запись следующий символов: + - . E |
| 17 | Некорректный ввод (несколько точек) | +123 | +1.2.3.E+0 | Проверьте запись следующий символов: + - . E |
| 18 | Некорректный ввод (буква вместо числа) | +1 | +f | Проверьте запись следующий символов: + - . E |
| 19 | Некорректный ввод (без знака) | +123 | 1.E+0 | Проверьте запись следующий символов: + - . E |
| 20 | Некорректный ввод (неправильный порядок) | +1 | +1.E+1.4 | Проверьте запись следующий символов: + - . E |
| 21 | Некорректный ввод (введен порядок, нет числа) | +1 | +.E+1 | Проверьте количество символов в мантиссе числа |
| 22 | Некорректный ввод (пустой ввод) | „\n“ | - | Проверьте запись следующий символов: + - . E |
| 23 | Некорректный ввод (пустой ввод) | +1 | ‘\n’ | Проверьте запись следующий символов: + - . E |
| 24 | Некорректный ввод (посторонний знак вместо «Е») | +1 | +1.Q+0 | Проверьте запись следующий символов: + - . E |
| 25 | Некорректный ввод (в порядке встречается не цифра) | +1 | +1.E+13r | Проверьте число на предмет записи лишних символов |
| 26 | Некорректный ввод (число введено буквами) | +ten | - | Проверьте число на предмет записи лишних символов |
| 27 | Ввод вещественного числа (начинаем с точки) | +1 | -.1E+0 | -0.1E+2 |
| 28 | Переполнение порядка (порядок меньше 99999) | +1 | +1000...00E+99999 | Машинный нуль |
| 29 | Переполнение порядка (порядок больше 99999) | +100..000 | +1.E-99999 | Произошло переполнение порядка |
| 30 | Округление | +999…99(30девяток) | +2.E+0 | +0.5E+30 |

**Функции**

**int number\_characters(char \*const str, const char el, const int str\_len)**

Функция считает количество вхождений определенного символа в строку

**Аргументы**

Str - строка

El - элемент, который ищем

Str\_len - длина строки

**Возвращаемые значения**

Количество вхождений символа в строку

**int number\_check(char \*const str, const int start\_index, const int end\_index)**

Функция бежит по строке и проверяет ее на лишние символы

**Аргументы**

Str - строка

Start\_index - индекс элемента, с которого идет проверка

End\_index – индекс элемента, на котором заканчивается проверка

**Возвращаемые значения**

0 – не встречено лишних символов

7 – встречен лишний символ

**int index\_find(char \*const str, const char el, const int str\_len)**

Функция ищет индекс определенного элемента в строке

**Аргументы**

Str - строка

El - элемент, индекс которого ищем

Str\_len - длина строки

**Возвращаемые значения**

Индекс элемента

**void structure\_filling(char \*const floating\_point\_number)**

Функция заполняет структуру данными

**Аргументы**

Floating\_point\_number – считанное действительное число

**Возвращаемые значения**

-

**void create\_integer\_number(void)**

Функция делает из действительного числа целое, учитывая порядок (внутри функции вызывается структура)

**Аргументы**

-

**Возвращаемые значения**

-

**int number\_convert(char number)**

Функция делает из числа типа char число типа int

**Аргументы**

Number – число типа char

**Возвращаемые значения**

Число типа int

**void subtraction(char \*first\_number, char \*second\_number)**

Функция вычитает из первого числа второе

**Аргументы**

First\_number – число, из которого вычитают

Second\_number – число, которое вычитают

**Возвращаемые значения**

-

**int numbers\_compare(char \*first\_number, char \*second\_number)**

Функция сравнивает два числа

**Аргументы**

First\_number – первое число

Second\_number – второе число

**Возвращаемые значения**

0 – числа равны

1 – первое число больше второго

2 – первое число меньше второго

**int division\_two\_numbers(char \*const integer\_number)**

Функция делит два числа (внутри функции вызывается структура)

**Аргументы**

Integer\_number – челое число (делимое)

**Возвращаемые значения**

0 – функция сработала успешно

2 – второе число равно 0

**int normalized\_output(void)**

Функция нормализует число для вывода (внутри функции вызывается структура)

**Аргументы**

-

**Возвращаемые значения**

0 – функция сработала успешно

11 – произошло переполнение порядка

12 – машинный нуль

**void print\_invite(void)**

Функция печатает приглашение на ввод

**Аргументы**

-

**Возвращаемые значения**

-

**void print\_error(const int error\_code)**

Функция печатает информацию об ошибке

**Аргументы**

Error\_code – код ошибки

**Возвращаемые значения**

-

**int condition\_test(char \*const integer\_number, char \*const floating\_point\_number)**

Функция проверяет все возможные ошибки при вводе чисел

**Аргументы**

Integer\_number – целое число

Floating\_point\_number – действительное число

**Возвращаемые значения**

Код ошибки или успешное выполнение функции

# **Ответы на контрольные вопросы**

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

Возможный диапазон чисел зависит от их типа, размера выделенной для их хранения памяти, разрядности процессора. Для беззнакового целого числа выделяется 64 двоичных разряда, то есть его максимальное значение – 18 446 744 073 709 551 615 (long long unsigned int).

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

Точность представления вещественных чисел определяется количеством памяти, выделяемой для хранения мантиссы числа. Для мантиссы числа типа double выделяется 52 бита, с помощью этого мантисса числа может иметь значение до 4 503 599 627 370 496.

*3. Какие стандартные операции возможны над числами?*

Возможны операции сложения, вычитания, умножения, деление, взятие остатка, сравнение.

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

Программист может выбрать структуру, куда он сможет записать не только мантиссу, но и знак числа и порядка. Так же может использовать массив символов.

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

Для этого можно использовать некоторые языки — где уже есть поддержка длинных чисел — или библиотеки, или написать свой алгоритм, реализующий нужную операцию.

# **Вывод**

В процессе выполнения данной лабораторной работы, была реализована функция деления чисел, превышающих допустимый диапазон. Был получен опыт в работе с массивами, структурами и типами данных. При необходимости обрабатывать числовые данные выходящие за пределы разрядной сетки, следует использовать структуры для хранения и обработки.