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 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

“ДЛИННАЯ АРИФМЕТИКА”

Студент **Доколин Георгий Александрович**

Группа **ИУ7-32Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Оценка _____

Преподаватель _____

2024 г.

Оглавление

Описание условия задачи

Смоделировать операцию деления целого числа длиной до 40 десятичных цифр на действительное число в форме ±m.n E±K, где суммарная длина мантиссы (m+n) - до 40 значащих цифр, а величина порядка K - до 5 цифр. Результат выдать в форме ±0.m1 E ±K1, где m1 - до 40 значащих цифр, а K1 - до 5 цифр.

Описание технического задания

Входные данные:

1. Целое число, вида [+-]m, где m – целая часть, состоящая только из десятичных цифр, длина которой больше 0 символов и меньше или равна 40 символам. Так же перед числом можно ввести его знак, однако при отсутствии число будет считаться положительным. Наличие иных символов в при вводе не допускается.
2. Вещественное число, вида [+-]m.n[eE][+-]k, где m – целая часть мантиссы, n – дробная часть мантиссы, k - порядок числа. Ввод вещественного числа разделен на две части. Первая часть считывает только знак, целую часть мантисы и дробнуяю часть мантисы. А вторая часть считывает порядок. Сначала пользователь может ввести знак, однако по умолчанию число считается положительным. Далее пользователь может ввести целую часть мантисы, однако в случае начала ввода с точки, целая часть мантисы будет равна 0. После ввода точки, пользователю необходимо ввести дробную часть мантисы. Суммарно длина целой и дробной части не должна превышать 40 символов. Иные символы при вводе не допускаются. После ввода мантисы пользователя попросят ввести порядок, который представляет из себя целое число, находящееся в диапозоне от -99999 до 99999.

Выходные данные:

1. Вещественное число, представленное в виде строки [+-]0.mE[+-]k, где m - мантисса числа, k – порядок числа. Длина мантиссы не превосходит 40 символов, но и не меньше 1 символа. Порядок находится в диапозоне от -99999 до 99999. Так же в начале выводится знак результата.

Реализуемая программой задача:

Деление целого числа на вещественное.

Способ обращения к программе:

Обращение к программе происходит через командную строку, при помощи ввода команды: “./app.exe”

Возможные аварийный ситуации и ошибки пользователя:

1. ERROR_LEN_NUM – ошибка длины вводимого целого числа. Значит вводимое число более 40 симолов.
2. ERROR_SYMBOL_IN_NUM – ошибка символа при вводе. Это значит введен некорректный символ.
3. ERROR_DOT_IN_NUM – ошибка количества точек в числе.
4. ERROR_ORDER – ошибка при вводе или выводе порядка числа.
5. ERROR_LEN_ORDER – ошибка длины порядка.
6. ERROR_LEN_FULL_NUM – ошибка общей длины числа
7. ERROR_RESULT_LEN - ошибка итоговой длины числа.

Описание структур данных

Целое число записывается в массив типа INT с выделенным размером в 40 элементов.

Знак целого числа хранится в отдельной переменной типа INT.

Вещественное число хранится в отдельной структуре:

```
typedef struct
{
   int sign;
   int mantissa_integer[LEN_MANTISSA];
   int mantissa_float[LEN_MANTISSA];
   int order;
} valid_numbers;
```

int sign – целочисленная переменная для сохранения знака.

int mantissa_integer – массив целых чисел, который хранит в себе целую часть вещественного числа.

int mantissa_float – массив целых чисел, который хранит в себе дробную часть вещественного числа.

int order – целочисленная переменная, которая хранит в себе порядок числа.

Константа LEN_MANTISSA равна 40, данная константа отвечает за длинну чисел.

Описание алгоритма

1. **Ввод данных и их обработка**
    1. **Вввод:**

        Сначала пользователь вводит целое число до 40 символов, которое записывается в массив символов. Далее пользователю предлагается ввести мантиссу, которая также вводится в массив символов. Последним на ввод идет порядок, который записывается в целочисленную переменную.

    2. **Обработка:**

        Каждый из 2 массивов проверяется на наличие сторонних симолов, а также на корректность ввода сразу после его введение и только в случае успешного ввода пользователю разрешается ввод следующего числа. Порядок вещественного числа также проверяется на корректность, тоесть идет проверка на граничные значения и сторонние символы.

2. **Нормализация чисел**
    1. **Обработка целого числа для будущего использования**
        1. Добавление нулей, чтобы выровнять его с вещественным числом.
        2. Запись количества добавленнных нулей в переменную для будущего подсчета порядка.
    2. **Обработка вещественного числа**
        1. Запись целой части в новый массив.
        2. Запись дробной части в новый массив.
3. **Деление**
    1. **Подготовка к деленииию**
        1. Проверка, что делимое по длинне не меньше делителя.
        2. Проверка, что сумма цифр делимого не меньше 0.
        3. Проверка на деление целого числа на 0.
        4. Проверка делимого числа на 0. В этом случае процесс деления не выполняется, а прогрмма переходит сразу к выводу.

2. **Основной алгоритм деления**
    1. Из делимого вычетается делитель до того момента, что делимое не станет меньше или равное 0.
    2. Параллельно с вычетанием идет подсчет количества этих вычетаний и именно оно и равно результату деления на данном этапе
3. **Проверка результата деления**
    1. Если результат деления меньше или равен нулю, то необходимо добавить еще одну цифру в конец делимого.
    2. В случае если изначальное число закончилось, добавляется 0 и сохраняется место для постановки точки.
    3. После выполнения прошлых двух пунктов, процесс вычитания повторяется до тех пор, пока результат от деления не станет больше 0.
4. **Запись результата**
    1. В случае получения результата от деления больше 0, этот результат записывается в массив.
    2. Весь алгоритм деления повторяется до тех пор пока длинна результата не станет больше 41 символа или же сумма оставшихся цифр не станет равна 0.
5. **Округление результата**
    1. Происходит обрезка нулей в конце числа
    2. Если длина числа более 40 символов, то происходит округление по математическим правилам.
    3. Обрезка нулей в конце числа перед печатью результата
6. **Вывод результата**
    1. Из функции деления возвращается сам массив, а так же указатель на место для запятой.
    2. Функция вывода результата определяет знак результата.
    3. Далее происходит вывод знака, результата, а также вывод разницы места для запятой и порядка, введенного в начале программы.

7. **Обработка ошибок**

   На этом этапе все ошибки возникающие в программе обрабатываеются и выдаются подробные сообщения.

Основные функции

// Функция ввода порядка числа

// Функция принимает адрес на структуру

int input_order(valid_numbers *number)

// Функция ввода массивов, которая так же проверяет длинну на 0

// Функция принимает массив симоволов, в который записывает результат ввода, длинну считывания, а также параметр для выбора ввода

int input_symbols(char num[LEN_MANTISSA + 3], size_t len_num, int code)

// Функция проверки символов внутри массива хранящего значения для мантисы

// Функция принимает массив символов, с которого считываются данные, массив символов, куда считывается целая часть вещественного числа, массив

// куда считывается дробная часть вещественного числа, адрес на переменную, хранящую знак числа

// адрес на переменную, хранящую длинну дробной части, адрес на перменную, хранящую длинну целой части вещественного числа

int check_elem_float(char saver_full_number[LEN_MANTISSA + 3], char mantissa_saver_integer[LEN_MANTISSA + 2], char mantissa_saver_float[LEN_MANTISSA + 2], int *flag, size_t *len_float, size_t *len_int)

// Функция записи данных в структуру

// Функция принимает адрес на структуру, длину целой части, длину дробной части

int input_float_arr(valid_numbers *number, size_t *len_num_int, size_t *len_num_float)

// Функция проверки целого в массиве символов

// Функция принимает массив символов, в котором хранится все целое число, массив целых чисел, куда записывается результат обработки,

// Адресс на перменную, хрянящую длину массива, адрес на переменную хранящую знак числа

int write_elem_integer(char saver_full_number[LEN_MANTISSA], int integer_arr[LEN_MANTISSA], size_t *len_int, int *flag)

// Функция записи символов в массив, хранящий целое число

// Функция принимает массив целых чисел, адрес на длину этого массива, а также адрес на переменную для хранения знака

int input_integer_arr(int integer_arr[], size_t *len_num_int, int *sign)

// Функция для округления результата перед выводом

// Функция принимает массив целых чисел, а так же адрес на переменную хранящую длину

void round_arr(int arr_result[LEN_MANTISSA], size_t *len_res_arr)

// Функция вывода результата

// Функция принимает адрес на структуру, также массив целых чисел, длину массива, знак целогого числа, а также переменную для подсчета порядка

int print_result(valid_numbers *number, int arr_result[LEN_MANTISSA], size_t *len_res_arr, int sign_int, int cnt_zero)

// Функция для выравнивания целого числа перед делением

// Функция принимает массив целых чисел для записи обработанного числа, массив целых чисел, откуда берутся данные, длину целой части вещественного числа,

// длину дробной части вещественного числа, длину целого числа, а так же адрес на переменную для хранения "указателя" на цифру внутри числа

void stabilize_nums(int buffer[LEN_MANTISSA], int integer_arr[], size_t len_num_int, size_t len_num_float, size_t len_integer_arr, size_t *cursor)

// Функция для объединения вещественного числа в один массив

// Функция принимает массив, в который будет записн обработанный массив, адрес на структура, длина целой части, длина дробной части

void write_to_all_num(int all_float_num[], valid_numbers *number, size_t len_num_int, size_t len_num_float)

// Функция для вычетания делителя из делимого

// Функциця принимает массив челых чисел из которого происходит вычетание, массив целых чисел, которых хранит в себе вычетаемое число

// массив целых чисел, который хранит в себе результат вычетания, длину числа из которого вычетают,

// адрес на переменную храняющую длинну массива с результатом, переменная хранящаю разницу между длинами чисел,

// адрес на переменную хранящую результат деления, адрес на переменную хранящую длину итогового массива

void substracting(int num[LEN_MANTISSA * 2], int all_float_num[LEN_MANTISSA], int history_arr[LEN_MANTISSA], size_t len_num, size_t *len_hist, size_t checker, int *need_sub, size_t *res_len)

// Основная функция деления, которая последовательно вызвает остальные функции в этом файле, а так же возвращает результат

// Функция принимает адрес на структуру, массив целых чисел, хранящий результат, адрес на длину итогового массива,

// массив целых чисел, хранящий целое число, длинну целой части дробного числа, длину дробной части дробного числа,

// длинну целого числа, адрес на переменную, хранящую место постановки точки

int separator(valid_numbers *number, int arr_result[LEN_MANTISSA * 2], size_t *len_res_arr, int integer_arr[], size_t len_num_int, size_t len_num_float, size_t len_integer_arr, int *cnt_z)

Тесты

Отрицательные:

Тест	Целое число	Вещественное число	Порядок	Ожидаемый результат	Класс эквивалентности
1	+-123	2	0	Ошибка символа в числе	Ввод нескольких знаков перед числом
2	++34	2	0		
3	--44	2	0		
4	66	--2	0		
5	56	++2	0		
6	89	+-2	0		
7	4444444444444444444444444444444444444444	2	0	Ошибка длины целого числа	Ввод больше 40 символов
8	1567	444444444444444444444444.44444444444444	0	Ошибка длины вещественного числа	
9	1567	.4444444444444444444444444444444444444444	0	Ошибка длины вещественного числа	
10	123	2.543	10000	Ошибка при вводе порядка	Ошибка при вводе порядка
11	123	2.543	-10000	Ошибка при вводе порядка	
12	123	2.543	4a	Ошибка при вводе порядка	
13	123a	2	0	Ошибка символа в числе	Ошибка символов при вводе
14	12a3	2	0	Ошибка символа в числе	
15	12.3	2	0	Ошибка символа в числе	
16	123	2.5.8	0	Ошибка, введено более 1 точки	
17	123	2a7	0	Ошибка символа вещественного числа	
18	123	0	0	Ошибка деления на 0	Деление на 0
19	123	0.00000	0	Ошибка деления на 0	
20	1	100	99999	Ошибка порядка в результате	Переполнение порядка
21	1	0.1	-99999	Ошибка порядка в результате	
22	123456789	0.000005	-99999	Ошибка порядка в	

				результате	

Положительные:

Тест	Целое число	Вещественное число	Порядок	Ожидаемый результат	Класс эквивалентности
1	125	5	0	+0.25 E2	Проверка подсчета знака
2	-125	5	0	-0.25 E2	
3	-125	-5	0	+0.25 E2	
4	49	7	0	+0.7 E1	Деление целого на целое
5	56	9	0	+0.62222222222222222222222222222222222222 E1	
6	49	7	0	+0.70 E1	
7	123	2.543	0	+0.48368692959496657491152182461659457 32 E2	Деление целого на дробное
8	345678	0.678	0	+0.50984955752212389385397345132743362 8 E6	
9	100	2.5	0	+0.4 E2	
10	345678	.678	0	+0.50984955752212389385397345132743362 8 E6	Проверка деления на вещственное без ввода целой части
11	100	.25	0	+0.4 E3	
12	00007	0.1	0	+0.70 E2	Проверка работы с незначащими нулями
13	000009	3	0	+0.3 E1	
14	7	0000.1	0	+0.70 E2	
15	9	00003		+0.3 E1	
16	99999999999999999999999999999999999	2	0	+0.5 E40	Деление больших чисел
17	99999999999999999999999999999999999	1111111111111111111111111111111111111	0	+0.9 E1	
18	99999999999999999999999999999999999	123456789123456789123456789123456789 1234	0	+0.81 E1	
19	0	7	0	+0.0 E0	Проверка деления 0 на число
20	0000	0.1	0	+0.0 E0	

21	0	2.543	0	+0.0 E0	
22	1	10	99999	+0. E99999	
23	1	100	99999	+0.E99999	
24	1	0.1	-99997	+0.E-99999	
25	1	0.1	-99996	+0.E-99998	

Вывод

Для работы с длинными числами, выходящими за границы компьютерного представления, необходимо и нужно реализовывать собственные типы данных для хранения длинных чисел, собственные функции для реализации операций над этими числами, проверок и вывода данных. Эта работа показала как может быть реализовано деление больших чисел.

Ответы на контрольные вопросы

- Каков возможный диапазон чисел, представляемых в ПК?

Диапазон чисел определяется разрядностью процессора и типом данных, используемых для хранения числа. Например, для вещественных чисел двойной точности в 64-разрядной архитектуре диапазон составляет от 3.6×10−4951 до 1.1×104932. Для беззнакового целого типа unsigned long long (8 байт) диапазон — от 0 до 18 446 744 073 709 551 615.

- Какова возможная точность представления чисел, чем она опреде-

ляется?

Точность числа зависит от объема памяти, выделенной для хранения его мантиссы. Максимальная стандартная точность достигается с использованием типа данных двойной точности (double), где мантисса занимает 52 бита. Это позволяет числу содержать до 16 значащих цифр.

- Какие стандартные операции возможны над числами?

Над числами можно выполнять операции сложения, вычитания, умножения, деления, целочисленного деления, взятия остатка, сравнение.

- Какой тип данных может выбрать программист, если обрабатывае-

мые числа превышают возможный диапазон представления чисел в ПК?

Программист может использовать структурный тип данных, содержащий поля для знака мантиссы, самой мантиссы и порядка.

- Как можно осуществить операции над числами, выходящими за

рамки машинного представления?

Можно реализовать собственные функции, которые будут выполнять не-

обходимые операции над числами, предварительно проверив их при помощи своих же функций.