Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: Информатика и системы управления
КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии
ДИСЦИПЛИНА: Типы и структуры данных
ТЕМА: Длинная арифметика
ВАРИАНТ: 9

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Студент: _____ *Княжев А. В.*
(подпись, дата)

Преподаватель: _____ *Силантьева А. В.*
(подпись, дата)

2021 г.

Содержание

1 Условие задачи

Создать программу для деления целого числа длиной до тридцати значащих цифр на вещественное число вида $\pm M.Ne \pm K$, где $(M + K)$ не более тридцати значащих цифр, а $-99999 \leq K \leq +99999$.

2 Техническое задание

Смоделировать операцию деления целого числа длиной до 30 десятичных цифр на действительное число в форме $\pm M.N \pm K$, где суммарная длина мантиссы $(M + N)$ — до 30 значащих цифр, а $-99999 \leq K \leq +99999$. Результат выдать в форме $\pm 0.M1 \pm K1$, где $M1$ — до 30 значащих цифр, а $-99999 \leq K1 \leq +99999$.

После ввода чисел программа должна выдать результат в указанном формате, либо вывести сообщение об ошибке при возникновении каки-либо аварийных ситуаций.

2.1 Входные данные

* длинное целое число со знаком или без, длиной до 30 значащих цифр;
* длинное вещественное число вида $\pm M.Ne \pm K$, где $(M+K)$ (целая и дробная части соотвественно) — не более тридцати значащих цифр, а $-99999 \leq K \leq +99999$. Порядок может отсутствовать, так же может отсутствовать целая или дробная части, но не обе вместе.

2.2 Выходные данные

* длинное вещественное число вида $\pm 0.Ne \pm K$, где N (дробная часть) — не более тридцати значащих цифр, а $-99999 \leq K \leq +99999$.

2.3 Действие программы

Программа осуществляет деление длинного целого числа на длинное вещественное и представляет результат в виде нормализованного вещественного числа.

2.4 Обращение к программе

Программа может быть запущена из командных оболочек `sh/bash/zsh/fish`. Программа не принимает никаких аргументов.

2.5 Аварийные ситуации

2.5.1 Ошибки ввода

1. в целой части мантиссы введено что-либо кроме цифр;
2. вместо знака мантиссы введено что-то кроме + и −;
3. в дробной части мантиссы введено что-то кроме цифр;
4. не введено ни целой, ни дробной части;

5. в мантиссе введено более двух точек;
6. некорректный знак экспоненты;
7. в порядке вместо знака что-то кроме + и −;
8. в поряде введено что-то кроме цифр;
9. количество цифр мантиссы больше тридцати;
10. порядок не входит в промежуток $[-99999; 99999]$;
11. в целом числе вместо знака что-то кроме + и −;
12. в целом числе есть что-то кроме цифр.

2.5.2 Ошибки арифметики

1. деление на нуль;
2. переполнение порядка.

3 Структуры данных

В данной работе используется запись со статическими полями — с помощью нее хранятся большие целые части и большие числа с плавающей точкой.

* запись для хранения больших целых чисел (содержит два поля — знак и число в виде массива);
* запись для хранения больших чисел с плавающей точкой (содержит два поля — мантиссу со знаком и порядок со знаком);
* запись для хранения ошибки (содержит два три — описание ошибки, код ошибки и название функции, в которой появилась ошибка);
* строка из 257 символов для ввода-вывода чисел;
* однобайтовое целое число.

3.1 Представление в коде

3.1.1 Основные константы

```
#define BIG_FLOAT_SIZE 30
#define BIG_FLOAT_MAX_EXPONENT 99999
#define BIG_FLOAT_MIN_EXPONENT -99999
#define BIG_INT_BASE 10
#define BIG_INT_SIZE BIG_FLOAT_SIZE * 2 + 1
#define SHORT_STRING_SIZE 257
```

3.1.2 Большое целое число

```
/**
 * content - содержимое числа (модуль числа);
 * sign - знак числа (-1 если число отрицательное, 1
 * если положительное, иначе 0).
 */
typedef struct {
    digit content[BIG_INT_SIZE];
    digit sign;
} bigint_t;
```

3.1.3 Большое число с плавающей точкой

```
/**
 * significand - мантисса числа;
 * exponent - порядок.
 */
typedef struct {
    bigint_t significand;
    int exponent;
} bigfloat_t;
```

3.1.4 Однобайтовое целое число

```
typedef char digit;
```

3.1.5 Строка

```
typedef char short_string_t[SHORT_STRING_SIZE];
```

3.1.6 Ошибка

```
/**
 * text - текст ошибки;
 * code - код ошибки;
 * func - функция, в которой случилась ошибка.
 */
typedef struct
{
    const char *text;
    int code;
    const char *func;
} error_t;
```

3.2 Модули обработки структур данных

3.2.1 Модуль для обработки большого целого числа

```
/**
 * Создание большого целого числа.
 */
bigint_t bi_create();
```

```
/**
 * Конвертация большого целого из long long.
 * n - число, которое будем конвертировать.
 */
bigint_t bi_from_ll(long long n);
```

```
/**
 * Изменение знака числа
 * a - число, знак которого нужно изменить.
 */
bigint_t bi_neg(bigint_t a);
```

```
/**
 * Сложение двух больших целых. Результат помещается в
 * первый аргумент.
 * a - первое слагаемое;
 * b - второе слагаемое.
 */
error_t bi_sum(bigint_t *a, bigint_t b);
```

```
/**
 * Вычитание двух больших целых. Результат помещается в
 * первый аргумент.
 * a - уменьшаемое;
 * b - вычитаемое.
 */
error_t bi_diff(bigint_t *a, bigint_t b);
```

```
/**
 * Целочисленное деление больших целых. Результат помещается в
 * первый аргумент.
 * a - делимое;
 * b - делитель.
 */
error_t bi_div(bigint_t *a, bigint_t b);
```

```
/**
 * Сравнение двух больших целых.
 * a - первое число для сравнения;
 * b - второе число.
 */
int bi_cmp(bigint_t a, bigint_t b);
```

```
/**
 * Сдвиги большого целого: влево, вправо, вправо с округлением.
 * Результат помещается в первый аргумент.
 * a - сдвигаемое число;
 * n - сдвиг.
 */
void bi_lshift(bigint_t *a, int n);
void bi_rshift(bigint_t *a, int n);
void bi_rshift_rounded(bigint_t *a, int n);
```

```
/**
 * Преобразование символа в цифру.
 * c - символ.
 */
digit char_to_digit(char c);
/**
 * Преобразование цифры в число.
 * d - цифра.
 */
char digit_to_char(digit d);
```

```
/**
 * Преобразование строки в большое целое.
 * str - преобразуемая строка;
 * a - резульат.
 */
error_t bi_from_str(char *str, bigint_t *a);
```

3.2.2 Модуль для обработки большого числа с плавающей точкой

```
/**
 * Преобразование строки в большое число с
 * плавающей точкой.
 * str - исходная строка;
 * a - результат преобразования.
 */
error_t bf_from_str(char *str, bigfloat_t *a);
```

```
/**
 * Преобразование большого числа с плавающей
 * точкой в строку.
 * str - результат преобразования;
 * a - преобразуемое число.
 */
error_t bf_to_str(char *str, bigfloat_t a);
```

```
/**
 * Преобразование большого целое в большое число
 * с плавающей точкой.
 * n - преобразуемое число;
 * a - результат преобразования.
 */
error_t bf_from_bigint(bigint_t n, bigfloat_t *a);
```

```
/**
 * Деление больших чисел с плавающей точкой.
 * a - делимое;
 * b - делитель.
 */
error_t bf_div(bigfloat_t *a, bigfloat_t b);
```

3.2.3 Модуль для обработки строк

```
/**
 * Чтение строки из файла.
 * f - файловый дескриптор;
 * str - строка, в которую вводим.
 */
error_t f_read_line(FILE *f, short_string_t str);
```

```
/**
 * Проверка строки на пустоту.
 * str - исходная строка.
 */
bool is_empty(char *str);
```

```
/**
 * Нормализация строки, удаление пробелов, возведение в один регистр.
 * str - исходная строка.
 */
char *normalize_str(char *str);
```

3.2.4 Модуль для обработки ошибок

```
/**
 * Создание новой ошибки.
 * text - текст ошибки;
 * code - код ошибки;
 * func - функция, в которой случилась ошибка.
 */
error_t new_error(const char *text, int code, const char *func);
```

```
/**
 * Создание ошибки-маркера успеха.
 * func - функция, в которой был создан "успех".
 */
error_t new_success(const char *func);
```

```
/**
 * Проверка, отображает ли ошибка ошибочную ситуацию.
 * err - исходная ошибка.
 */
bool is_failure(error_t err);
```

4 Описания алгоритмов

В случае неуспешности каких-либо операций, происходит вывод сообщения об ошибке и завершение работы программы с ненулевым кодом возврата.

1. считываются входные данные;
2. делимое делится на делитель;
3. результат преобразуется в строку;
4. результирующая строка выводится.

4.1 Считывание входных данных

1. считывание целого числа в виде строки (`f_read_line`);
2. проверка строки на переполнение;
3. считывание вещественного числа в виде строки;
4. проверка строки на переполнение;
5. перевод строки с целым числом в большое целое (`bi_from_str`);
6. перевод большого целого в большое число с плавающей точкой (`bf_from_bigint`);
7. перевод строки с вещественным в большое число с плавающей точкой (`bf_from_str`).

4.2 Сложение (вычитание) больших целых чисел (`bi_add, bi_diff`)

Используется метод сложения «в столбик». Большие целые числа хранятся в виде знака (целое число), и массива цифр, причем массив хранится в перевернутом виде.

1. если операнд отрицательный, происходит инверсия знаков всех его цифр, то есть если есть число -42, то оно записывается при сложении, как `[-2 -4]`;
2. происходит проход числа «справа налево» (если смотреть в прямой записи), как только встречаются ненулевые разряды, проверяется знак итоговой суммы;
3. при сложении цифр проверяется размер получившейся суммы, а также его знак, он должен соответствовать итоговому знаку суммы, определенному ранее;
4. если цифра не соответствует (в данному случа, по модулю больше 10 или имеет знак, отличный от итогового), то оно приводится к нормальному виду с помощью прибавления/вычитания 10 и переноса единицы со знаком «+» или «−» на следующий разряд;

5. происходит обратное приведение цифр к нормальному виду, то есть если они были отрицательные, они становятся положительными.

4.3 Деление больших целых (`bi_div`)

1. сравнивается количество цифр делимого и делителя;
2. если количество цифр делителя больше, то возвращается нуль;
3. происходит определение порядка n такого, чтобы длина делимого и делителя умноженного на 10^n совпадали;
4. пока возможно, вычитается домноженный делитель из делимого. Подсчитывается количество таких вычитаний;
5. это количество вычитаний ставится на n-ую позицию результата;
6. домноженный делитель делится на 10;
7. значение n уменьшается на единицу;
8. все это происходит, пока $n \geq 0$.

4.4 Деление больших чисел с плавающей точкой (`bf_div`)

1. к мантиссе дописывается 30 нулей (умножение мантиссы на 10^{30});
2. происходит целочисленное деление домноженной мантиссы делимого на мантиссу делителя;
3. порядок делителя вычитается из порядка делимого;
4. происходит нормализация числа.

4.5 Нормализация большого числа с плавающей точкой (`__to_normal`)

1. происходит поиск первого вхождения ненулевой цифры мантиссы (в прямом порядке записи первой слева);
2. происходит сдвиг вправо с округлением таким образом, чтобы эта цифра стояла на 29 позиции;
3. порядок изменяется в соответствие со сдвигом.

4.6 Сдвиг вправо с округлением (`bi_rshift_rounded`)

1. если сдвигаем на отрицательное количество позиций, то вызывается сдвиг влево без округления;
2. происходит проверка ближайшего элемента, который обрежется при округлении;
3. происходит сдвиг числа вправо;
4. если обрезанный элемент больше пяти, то к результату прибавляется единица.

4.7 Перевод строки в большое число с плавающей точкой (`bf_from_str`)

1. строка разбивается на мантиссу и порядок. Порядка может не быть, а вот мантисса быть обязана;
2. мантисса переводится в большое целое с соответствующим положению точки изменению порядка:
   (a) если в начале мантиссы «−», то сразу меняем ее знак;
   (b) проходим по строке, пока не встретим точку или конце строки;
   (c) если встречаем цифру, то записываем ее в соответсвующий элемент массива, содержащего цифры мантиссы. Запись начинается в обратном порядке с 29 по 0 элементы (так как в мантиссе не более 30 знаков);
   (d) если встречаем что-то другое, выходим с ошибкой;
   (e) прибавляем к порядку количество значащих цифр до точки;
   (f) проходим по строке после точки до конца;
   (g) если видим цифру, то дописываем ее к мантиссе;
   (h) если еще не встретили ни одной значащей цифры, то вычитаем из порядка единицу при каждой встретившийся цифре;
   (i) если встречаем не цифру, выходим с ошибкой.
3. проверяем мантиссу на переполнение;
4. порядок переводится в целое число:
   (a) проходим по строке;
   (b) если встречаем знак «−», то сохраняем его;
   (c) проходим по всем цифрам, если встречаем не цифру, то выходим в ошибкой;
   (d) если встречаем цифру, то умножаем порядок на 10 и прибавляем цифру;

(e) если был минус в начале, то домножаем порядок на -1.

5. проверяем порядок на переполнение;
6. складываем изменение порядка от мантиссы и считанный порядок.

4.8 Перевод большого числа с плавающей точкой в строку (`bf_to_str`)

1. записывается знак мантиссы и `0.`;
2. записывается сама мантисса (значащие разряды);
3. записывается `E`;
4. записывается порядок со знаком.

4.9 Перевод строки в большое целое число (`bf_from_bigint`)

1. проверяем строку на переполнение;
2. определяется знак числа (первый символ);
3. происходит проход строки от конца к началу;
4. если встречаем не цифру, выходим с ошибкой;
5. ставим цифру на соотвествующую позицию.

4.10 Перевод большого целого в большое число с плавающей точкой

1. присвоение мантиссе значения большого целого;
2. изменение порядка числа, увеличение его на 30 разрядов;
3. нормализация числа.

5 Тестирование

Для проверко корректности работы программы было проведено функциональное тестирование. Таблица с тестовыми данными для «позитивных» и «негативных» случаев приведена ниже.

5.1 «Негативные» тесты

Название	**Входные данные**	**Выходные данные**
введена пустая строка		Строка пуста (f_read_line).
некорректное целое число	1a \n 1	Некорректное число (bi_from_str).
в целом числе введен только знак	- \n 1	Целое число пустое (bi_from_str).
в мантиссе второго числа больше двух точек	1 \n 1.2.3	Некорретная дробная часть (__parse_significant).
введена строка из пробелов		Строка пуста (f_read_line).
введен неправильный знак в целом числе	*34 \n 28	Некорректное число (bi_from_str).
введен неправильный знак в мантиссе вещественного	34 \n *28.22	Некорретная целая часть (__parse_significant).
введен неправильный знак в порядке	10 \n 1e*10	Некорретная степень (__parse_exponent).
введены не цифры в целой части мантиссы	10 \n -1a.10e5	Некорретная целая часть (__parse_significant).
введены не цифры в дробной части мантиссы	10 \n -10.1ae5	Некорретная дробная часть (__parse_significant).
введены не цифры в порядке	10 \n 10.4e17y	Некорретная степень (__parse_exponent).
деление на нуль	10 \n 0	Деление на нуль (bi_div).
переполнение при делении	10000000000000 \n 1e-99999	Переполнение экспоненты (bf_div).

Название	**Входные данные**	**Выходные данные**
слишком большое целое	100000000000000000000000000000 \n 1	Слишком большое целое число. (bf_from_bigint).
слишком большая мантисса	1 \n 999.999999999999999999999999e-10	Мантисса слишком большая (__parse_significant).
слишком большой порядок	1 \n 1e100000	Переполнение экспоненты (bf_from_str).
слишком маленький порядок	1 \n 1e-100000	Переполнение экспоненты (bf_from_str).

5.2 «Позитивные» тесты

Название	**Входные данные**	**Выходные данные**
деление числа само на себя	12 \n 12	+0.1E+1
обычный тест	1 \n 2	+0.5E+0
длинный результат	1 \n 3	+0.33333333333333333333333333333333E+0
обычный тест	4444444444444444444444444444 \n 2.2	+0.2020202020202020202020202020202E+30
целое очень большое	999999999999999999999999999999 \n 1	+0.999999999999999999999999999999E+30
деление большого числа самого на себя	999999999999999999999999999999 \n 999999999999999999999999999999	+0.1E+1
крайний порядок перед переполнением	1 \n 1e-99998	+0.1E+99999
округление разрядов	999999999999999999999999999999 \n 4	+0.25E+30
деление на единицу	2929291 \n 1000e-3	+0.2929291E+7
деление нуля на число	0 \n 20e9999	+0.0E+0
порядок близок к граничному	1 \n 100e-99999	+0.1E+99998

6 Контрольные вопросы

6.1 Каков возможный диапазон чисел, представляемых в ПК?

Каждый тип данных характеризуется определенным диапазоном значений чисел, который, в свою очередь, зависит от размера области памяти, выделяемой под хранение переменной этого типа, от наличия знака в числе и от типа представления числа (целое или вещественное).

Для 64-битного типа данных, например беззнакового, диапазон значений будет $[0, 2^{64} - 1]$, то есть $[0, 18446744073709551615]$.

6.2 Какова возможная точность представления чисел, чем она определяется?

Для целых чисел количество разрядов определяет точность представления чисел. Для вещественных точность определяется длиной мантиссы.

В современных 64-битных компьютерах максимальный размер чисел (как вещественных, так и целых) обычно ограничивается 64 разрядами. То есть целое можно представить с точностью в 64 двоичных разряда. В то же время, из 64 разрядов под мантиссу числа с плавающей точкой выделяется 52 разряда, под представление порядка — 11. 52 двоичных разряда соответствует около 15 десятичных разрядов, поэтому, если нам нужна точность более 15 знаков, мы прибегаем к необходимости реализовывать свои типы для хранения и обработки таких чисел.

6.3 Какие стандартные операции возможны над числами?

Сложение, вычитание, умножение, деление, инкремент, декремент, изменение знака, сдвиги.

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

Можно использовать массив цифр или символов, также число можно разбивать по кусочкам и хранить массив, состоящий из этих кусочков.

Также хорошим методом для хранения таких чисел является запись, содержащая информацию о цифрах числа и его знаке, в случае чисел с плавающей запятой — мантиссе и порядке.

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

Так как соотвествующих типов под такие числа нет, и их создаем мы, то соответствующие операции тоже реализовывать должны мы. Тогда, например, для сложения, вычитания и умножения можно использовать стандартные алгоритмы «в столбик», для деления «уголок».

7 Вывод

В данной работе я ознакомился с числами с плавающей точкой большой точности. Не всегда нам подходят числовые типы данных, предоставляемые языком (например из-за отсутствия требуемой точности), так что в данном случае, логику работы числел с плавающей запятой пришлось реализовывывать самому.

В качестве метода сложения/вычитания чисел был использовать метод наподобие «столбика», для деления метод, похожий на школьный «уголок». Данные методы довольно просты и удобны для реализации и восприятия.

Список литературы

[1] Методические рекомендации по лабораторной работе №1 (*http://wwwcdl.bmstu.ru/*)