Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6 по курсу «Дискретный анализ»

Студент: Е. С. Пищик
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №6

Задача: Необходимо разработать программную библиотеку на языке C или C++, реализующую простейшие арифметические действия и проверку условий над целыми неотрицательными числами. На основании этой библиотеки нужно составить программу, выполняющую вычисления над парами десятичных чисел и выводящую результат на стандартный файл вывода.

Список арифметических операций:
Сложение.
Вычитание.
Умножение.
Возведение в степень.
Деление.

В случае возникновения переполнения в результате вычислений, попытки вычесть из меньшего числа большее, деления на ноль или возведении нуля в нулевую степень, программа должна вывести на экран строку Error.
Список условий:
Больше.
Меньше.
Равно.

В случае выполнения условия программа должна вывести на экран строку true, в противном случае — false.

Количество десятичных разрядов целых чисел не превышает 100000. Основание выбранной системы счисления для внутреннего представления «длинных» чисел должно быть не меньше 10000.

Формат входных данных: Входной файл состоит из последовательности заданий, каждое задание состоит из трех строк:

Первый операнд операции.

Второй операнд операции.

Символ арифметической операции или проверки условия.

Числа, поступающие на вход программе, могут иметь «ведущие» нули.

Формат результата: Для каждого задания из выходного файла нужно распечатать результат на отдельной строке в выходном файле:

Числовой результат для арифметических операций.

Строку Error в случае возникновения ошибки при выполнении арифметической операции. Строку true или false при выполнении проверки условия.

В выходных данных вывод чисел должен быть нормализован, то есть не содержать в себе «ведущих» нулей.

1 Описание

Требуется реализовать класс для хранения «длинных» чисел, реализовать операции сложения, вычитания, умножения, деления и возведения в степень. Сложение выполняется очень просто, складываем разряды, в текущий разряд записываем остаток, а целую часть от деления запоминаем для следующего разряда. Вычитание выполняется аналогичным образом, с запоминанием, требуется ли нам «занять» число. Умножение выполняется аналогичным образом, только для каждой пары разрядов и последующих сдвигов для больших разрядов. Деление в степень выполняется начиная с больших разрядов, берется минимальное число, которое делится нацело, при помощи бинарного поиска и вычитается как в обычном делении стобком, и так до конца разрядов. Возведение в степень выполняется, используя уже написанное умножение, рекурсивно, если степень четная, то умножаем полученный результат из рекурсии сам на себя и делим степень пополам, если степень нечетная, то умножаем на исходное число и вычитаем из степени 1.Сравнения происходят очень просто, сначала сравниваем длины, если равны, то сравниваем по разрядам.

2 Исходный код

bigint.cpp	
TBigInt::TBigInt(std::string const& str)	Конструктор с 1 аргументом - строкой, который производит инициализацию.
void TBigInt::Init(std::string const& str)	Функция для инициализации строкой.
void TBigInt::DeleteLeadingZeros()	Функция удаления ведущих нулей.
std::ostream& operator<<(std::ostream& out, TBigInt const& rhs)	Оператор вывода для TBigInt.
std::istream& operator>>(std::istream& in, TBigInt& rhs)	Оператор ввода для TBigInt.
TBigInt const operator+(TBigInt const& lhs, TBigInt const& rhs)	Оператор сложения 2 TBigInt.
TBigInt const operator-(TBigInt const& lhs, TBigInt const& rhs)	Оператор вычитания 2 TBigInt.
TBigInt const operator*(TBigInt const& lhs, TBigInt const& rhs)	Оператор умножения 2 TBigInt.
TBigInt const TBigInt::MultShort(TBigInt const& rhs) const	Функция умножения «длинного» на «короткое».
TBigInt const operator^(TBigInt const& lhs, TBigInt const& power)	Оператор возведения числа в степень.
void TBigInt::ShiftRight()	Функция сдвига вправо для числа.
TBigInt const operator/(TBigInt const& lhs, TBigInt const& rhs)	Оператор деления 2 TBigInt.
bool operator<(TBigInt const& lhs, TBigInt const& rhs)	Оператор сравнения «меньше» 2 TBigInt.
bool operator==(TBigInt const& lhs, TBigInt const& rhs)	Оператор сравнения «равно» 2 TBigInt.
bool operator!=(TBigInt const& lhs, TBigInt const& rhs)	Оператор сравнения «не равно» 2 TBigInt.
bool operator<=(TBigInt const& lhs, TBigInt const& rhs)	Оператор сравнения «меньше или равно» 2 TBigInt.
bool operator>(TBigInt const& lhs, TBigInt const& rhs)	Оператор сравнения «больше» 2 TBigInt.
bool operator>=(TBigInt const& lhs, TBigInt const& rhs)	Оператор сравнения «больше или равно» 2 TBigInt.

main.cpp	
int main()	Основная функция, собирающая все части программы в единое целое. Парсит пользовательский ввод и выдает результат для различных операций, с различными условиями для чисел.

```
class TBigInt
{
public:
    static size_t const BASE = 10000;
    static size_t const RADIX = 4;

    TBigInt() = default;
    TBigInt(std::string const& str);
    void Init(std::string const& str);
    TBigInt const MultShort(TBigInt const& rhs) const;
    void DeleteLeadingZeros();
    void ShiftRight();

    friend std::istream& operator>>(std::istream& in, TBigInt& rhs);
    friend std::ostream& operator<<(std::ostream& out, TBigInt const& rhs);
    friend TBigInt const operator+(TBigInt const& lhs, TBigInt const& rhs);
    friend TBigInt const operator-(TBigInt const& lhs, TBigInt const& rhs);
    friend TBigInt const operator*(TBigInt const& lhs, TBigInt const& rhs);
    friend TBigInt const operator^(TBigInt const& lhs, TBigInt const& power);
    friend TBigInt const operator/(TBigInt const& lhs, TBigInt const& rhs);
    friend bool operator<(TBigInt const& lhs, TBigInt const& rhs);
    friend bool operator<=(TBigInt const& lhs, TBigInt const& rhs);
    friend bool operator>(TBigInt const& lhs, TBigInt const& rhs);
    friend bool operator>=(TBigInt const& lhs, TBigInt const& rhs);
    friend bool operator==(TBigInt const& lhs, TBigInt const& rhs);
    friend bool operator!=(TBigInt const& lhs, TBigInt const& rhs);

private:
    std::vector<int32_t> data;
};
```

3 Консоль

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_sum.txt
050604130001
021410140107
+
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_sum.txt
72014270108
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_sub.txt
040500130012
021410140107
-
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_sub.txt
19089989905
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_mult.txt
050604130001
021410140107
*
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_mult.txt
1083441513314252050107
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_power.txt
14
02
^
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_power.txt
196
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_div.txt
050604130001
031014
/
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_div.txt
1631654
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_less.txt
001203
100103
<
```

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_less.txt
true
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_more.txt
9927
0234
>
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_more.txt
true
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ cat test_eq.txt
9927
00927
=
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./solution
<test_eq.txt
true
```

4 Тест производительности

Тест состоит из трех подчастей для сложения, умножения и вычитания. Каждая часть теста проходит с константами, заданными перед началом замера времени, количество выполнений константых операций для всех операций равно 100 млн.

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution_root/solution$ ./benchmark
Bigint sum time: 5.65296 seconds
Gmp sum time: 0.907357 seconds

Bigint sub time: 5.56938 seconds
Gmp sub time: 0.993383 seconds

Bigint mult time: 9.86926 seconds
Gmp mult time: 0.723498 seconds
```

Как можно увидеть, время работы примерно gmp достаточно сильно меньше на всех операциях, чем сложнее операция, тем больше разница, так мы реализовывали достаточно примитивные алгоритмы, не дающие такой скорости, как реализация в gmp.

5 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я познакомился с различными алгоритмами работы с «длинными» числами. Посмотрел на способы их внутреннего представления, научился писать основные арифметические операции для них, за приемлимое время выполнения данных операций.

Список литературы

[1] Длинная арифметика

[2] e-maxx