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: М. А. Волков
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б
Дата: 11 марта 2021 г.
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №2

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:

1. Сложение (+).
2. Вычитание (-).
3. Умножение (*).
4. Возведение в степень ($\wedge$).
5. 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:

1. Больше ($>$).
2. Меньше ($<$).
3. 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 данный

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

Как сказано в [1]: «**Длинная арифметика** – выполняемые с помощью вычислительной машины арифметические операции (сложение, вычитание, умножение, деление, возведение в степень, элементарные функции) над числами, разрядность которых превышает длину машинного слова данной вычислительной машины. Эти операции реализуются не аппаратно, а программно, с использованием базовых аппаратных средств работы с числами маньших порядков.»

Для реализации данной арифметики, мы используем строки, которые в дальнейшем переносим в массив, разбивая на какое-то количество цифр. Количество цифр подбирается экспериментальным путем так, чтобы при умножении чисел не было переполнения числа.

Умножение, сложение, вычитание и деление программировалось путем применения алгоритма вычисления операции «столбиком».

Вычисление степени числа программировалось путем факторизации числа. Имеется в виду, что показатель степени бился на биты и умножалось тогда число, когда бит равнялся 1.

2 Исходный код

Ниже приведен полный код программы.

Ввод и вывод работают так, как написано в задании.

main.c	
TLongAlg(const std::string & str)	Констуктор, который разбивает число
friend TLongAlg operator+(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор +, вычисляющий сложение
friend TLongAlg operator-(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор -, вычисляющий вычитание
friend TLongAlg operator*(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор *, вычисляющий умножение
friend TLongAlg operator/(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор /, вычисляющий целочисленное деление
friend TLongAlg operator^(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор ^, вычисляющий возведение в степень
friend TLongAlg operator>(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор >
friend TLongAlg operator<(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор <
friend TLongAlg operator==(const TLongAlg &lhs, const TLongAlg &rhs)	Перегруженный оператор =

operator /

```
1  friend TLongAlg operator/(const TLongAlg &lhs, const TLongAlg &rhs) {
2      TLongAlg result;
3      TLongAlg peaceOfDigit;
4
5      if (rhs == TLongAlg("0")) {
6        throw std::invalid_argument("Error");
7      }
8
9      if (lhs < rhs) {
10       return TLongAlg("0");
11     }
12
13     for (int i = lhs._data.size() - 1; i >= 0; --i) {
14       int64_t l = -1;
15       int64_t r = BASE;
16       peaceOfDigit._data.insert(peaceOfDigit._data.begin(), lhs._data[i]);
17       peaceOfDigit.ClearZeroes();
18
19       while (l + 1 < r) {
```

```
20        int64_t m = (l + r) / 2;
21        auto tmp = rhs * TLongAlg(m);
22        (!(peaceOfDigit < rhs * TLongAlg(m))) ? l = m : r = m;
23      }
24
25      result._data.push_back(l);
26      auto tmp = rhs * TLongAlg(l);
27      peaceOfDigit = peaceOfDigit - TLongAlg(l) * rhs;
28    }
29
30
31    std::reverse(result._data.begin(), result._data.end());
32    result.ClearZeroes();
33    return result;
34  }
```

3 Консоль

```
3894343298352143534643 6
354353254328383
+
904094384738493247293847 3843
2343543
-
972323
2173937
>
2
3
-

3894343333787468967481 9
904094384738493247293613 0300
false
Error
```

4 Тест производительности

Сравнение производительности будет производиться со стандартной библиотекой gmp c++. Сравнение будет происходить только с умножением и делением. Так как сложение и вычитание ассимптотически работают что у меня, что в библиотеке за линию.

кол-во тестов и цифр	время моего кода	время gmp
Умножение		
10^4 и 10^2	111.991 ms	51.043 ms
10^3 и 10^3	384.672 ms	50.694 ms
10^2 и 10^4	3309.01 ms	54.447 ms
Деление		
10^5 и 10	188.716 ms	85.443 ms
10^4 и 10^2	75.153 ms	51.759 ms
10^3 и 10^3	56.331 ms	45.642 ms

Как и следовало ожидать, мой способ решения задачи будет на много дольше работать, так как умножение происходит за сложность $O(n * m)$. В свою очередь деление за $O(logn * m)$

5 Выводы

Благодаря проделанной работе, я узнал как работает длинная арифметика, которая очень часто используется в ЯП python. Также сделал свой собственный калькулятор на основе длинной целочисленной арифметике.

Список литературы

[1] *Длинная арифметика – wikipedia.*

[2] *Длинная арифметика – E-maxx.*

[3] Список использованных источников оформлять нужно по ГОСТ Р 7.05-2008