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: А. В. Барсов
Преподаватель: Н. С. Капралов
Группа: М8О-205Б-19

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

Предполагаем, что не любое введённое число представляется возможным сохранить в базовом целочисленном типе, так как оно слишком большое и происходит переполнение. Тогда будем его хранить в виде массива, сохраняя каждый разряд отдельно в виде целого неотрицательного числа. Как писал Кнут [1]:«Числа повышенной точности можно рассматривать как числа, записанные в системе счисления по основанию w, где w - размер слова». Размер слова w рекомендуется выбрать таким образом, чтобы длина массива была как можно меньше, однако необходимо, чтобы результат любой элементарной операции над двумя элементами размера слова умещался в это же слово, то есть не происходило переполнение базовых типов данных.

Алгоритмы сложения, вычитания, умножения и деления аналогичны механическому выполнению знакомых операций сложения в столбик, деления в столбик и т.д.

Время работы алгоритмов сложения, вычитания, сравнений двух чисел - $O(n)$, где n - длина массива, содержащего число по основанию w. Иначе говоря, сложность $= O(l/\log_{10} w)$, где l - длина введённого числа в 10-ичной системе счисления.

Алгоритм умножения «в столбик» работает за $O(n * m)$.

Алгоритм деления реализован через подбор наилучшего частного, так что его сложность равна $O(mn - n^2)$.

Алгоритм возведения степень m^n реализован с помощью бинарного возведения в степень [2], что позволяет проводить операцию за $O(\log n)$ умножений.

2 Консоль

```
[artem@IdeaPad solution]$ python generator.py 6 >input.txt
[artem@IdeaPad solution]$ cat input.txt
283679655592757793559923825054813532782931531510736436702128727727926755202
353721550083150221752986952153455162128891718483934365168625
*
350387753138543424664631341390967687006537753942603876425904552341229668 45
611956879452615287703003220000416751674527327454963815572368
<
659981148116348657672577946440343239831306772338730611935644865283214530606
129243663733833903865261314542255402129527899329348549085525
-
739792094897282374699338042210227184480651799754177279485443877072689833227
113380810915484603583307819784615150010380440982611718269240
/
651051619354531986956647478483510717661088328782767636276847569091797376751
167336719252642052455287896177249525772667754290996254524497
-
570473906012361106420010393174733236493050686345090688188109096659632425529
167080039750349585719131787606756961605373781585681447195 60
<
[artem@IdeaPad solution]$ make
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm -c main.cpp
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm -c BigInteger.cpp
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm main.o BigInteger.o
-o solution
[artem@IdeaPad solution]$ ./solution <input.txt
100343607503324481781945049067030072362014519160530172575121162655941167843468746257 14
false
659981148116348528428914212606439374569992230083328482407745535934665445081
6524843921329263
651051619354531819619928225841458262373192151533241863609093278095542852254
false
```

3 Тест производительности

Тест производительности представляет из себя сравнение результатов и времени работы разработанной библиотеки для длинной арифметики с уже существующей реализацией на Python.

Тест состоит из 1000 наборов пар чисел и операции для вычисления.

```
[artem@IdeaPad solution]$ ./wrapper.sh
Generating tests... OK.
Testing python BigInteger...
real 0m1.327s
user 0m1.309s
sys 0m0.017s
OK.
Testing my C++ BigInteger...
real 0m8.690s
user 0m8.668s
sys 0m0.010s
OK.
Comparing output... OK.
Start benchmark? [Y/n]

Number of tests: 1000
Oper.   Python  My C++
+       2.59    0.34
-       2.58    0.33
*       3.09    1.78
/       2.45    2.58
^       49.99   332.76
<       0.86    0.28
>       0.86    0.32
=       0.77    0.27
```

Как видно, реализованная библиотека показала лучшую производительность почти во всех случаях.

4 Выводы

Выполнив данную лабораторную работу по курсу «Дискретный анализ», я научился эффективно проводить вычисления над числами произвольной длины, оптимизировать вычисления, находить и устранять утечки памяти при помощи специальных инструментов. Я понял, как работает неочевидная операция деления чисел повышенной точности.

Список литературы

[1] Дональд Кнут *Искусство программирования.* Перевод и издатели: В. Штонда, Г. Петриковец, А. Орлович.

[2] *Справочник по алгоритмам и структурам данных.*
URL: `https://e-maxx.ru/` .