Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Дискретный анализ»

Студент: Р. Р. Гаптулхаков
Преподаватель: Н. С. Капралов
Группа: М8О-208Б

Дата: 13.03.21
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 ($\wedge$).
- 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.

1 Описание

Иногда, встроенных типов данных бывает не достаточно для произведения различных расчётов. Поэтому появляется необходимость в реализации представления максимально возможного числа в компьютере. В лабораторной работе 6 мы пытаемся решить проблему работы с большими числами. Для представления числа берётся массив типа T, где T может быть встроенным типом данным, а может быть и своей реализации. Для примера возьмём массив типа $Integer$, который занимает 4 байта в компьютере. Максимальное значение типа int : 2147483647. Это число характеризует максимальное основание нашего длинного числа. Пусть массив - это последовательность ячеек, каждая из которых занимает по 4 байта и в каждой которой может быть число не более 2147483647. Мы разделяем наше длинное число на данные ячейки удовлетворяя условиям написанным выше. Стоит отметить, что система счисления должна быть определена и постоянна на всё этапе работы программы. Далее для более удобного проведения вычислений мы разварачиваем наш массив. Это делается для того, чтобы не смещать весь массив при появление не существующего до вычисления более старшего разряда. Мы можем безболезненнго вставить ещё одну ячейку при развороте. Далее рассмотри различные операции с длинными числами.

Сложение. Пожалуй самая простая операция, за исключением операций сравнения. Итак, для вычисления операции сложения мы прибегнем к простому школьному алгоритму сложения в столбик. Прибавляем, переносим десятки получаем ответ и радуемся жизни. Сложность по времени: O(max(n,m)).

Вычитание. Ничем не отличается от операции сложения, также решаем проблему школьным алгоритмом вычитания. Вычитаем, если не достёт занимает десяток и всё супер. Главное не забыть, что мы работаем только с положительными числами и 0, а значит нужно проверять то, что первое число не меньше второго. Сложность такая же, что логично, O(n). Двигаемся дальше.

textbfУмножение. Всё так же, школьный алгоритм поможет нам в этом деле, не зря учился. Перемножаем, складываем, получаем ответ. Для реализации используется 2 цикла, один из которых вложеный отсюда получается сложность $O(n^2)$. Есть более быстрые алгоритмы, которые работают за $O(n * logn)$. Например, это алгоритм Карацубы и алготитм дескретного преобразования Фурье. Алгоритм Карацубы делит два числа пополам и путём преобразований получает не 4 произведения, а 3 за счёт этого получаем ускорение. Идея алгоритма БПФ(FFT), заключается в следующем: представляем наш массив, как коэффициенты полинома делим пополам наш полином на полиномы с чётными и нечётными степенями. Происходит чёрная магия и мы получаем полином произведения. Из него получаем ответ.

Деление. Пожалуй самая сложная часть реализации. Деление будет реализовано "уголоком". Сносим старшие $BUCKET$ разрядов и бинарным поиском определяем множитель частного. Сложность: $O(n * (n + log(BASE) * m))$ $O(n^2 + n * m)$, n,m -

длины чисел, BASE - основание СС. Деление на ноль выводит ошибку.

Возведение в степень. Чтобы не перемножать число n раз, воспользуемся формулой $b^n = (b^{n/2})^2$. Если n - нечётное нужно представить число в виде $b^n = b^n * b$. Возведение выполняется за $O(log(n) * m^2)$, n - степень, m - длина.

2 Исходный код

Проект состоит из 4 файлов: lab6.cpp, BigInt.hpp, BigInt.cpp, makefile.

- **lab6.cpp**:основной файл, в котором происходит взаимодействие программы с пользователем, функции для работы с файлами;

- **BigInt.hpp**: объявление структур, классов, их методов;

- **BigInt.cpp**: реализация методов класссов, структур;

Таблица методов и функций

lab6.cpp	
Функция	Описание
int main()	Главная функция, в которой происходит чтение данных.
BigInt.hpp	
Функция	Описание
class TBigInt()	Класс для хранения числа и реализация перегруженных операторов вычислений.
std::vector<int> bigNum	Структура дял хранения числа.
TBigInt(std::string str)	Конструктор копирования.
TBigInt operator+(const TBigInt)	Перегруженный оператор сложения.
TBigInt Power(TBigInt r)	Возведение в степень.
bool operator>(const TBigInt) const	Сравнение.
BigInt.cpp	
Функция	Описание
void TBigInt::DeleteZeros()	Удаляет незначащие нули.
std::string Expand(std::string str)	Разворачивает число в массиве.

3 Тест производительности

Для сравнения скорости работы с длинными числами воспользуемся Python3. Для замерки времени использовалась бибилиотека **chrono**, а в Python3 *time*.
Тесты создавались с помощью программы на языке Python:

```
import time

start = time.time()

while True:
    try:
        num1 = int(input())

    except:
        break
    num2 = int(input())
    op = input()

    if(op == '+'):
        print(num1+num2)

    elif(op == '-'):
        if num1 < num2:
            print("Error")
        else:
            print(num1 - num2)


    elif(op == '*'):
        print(num1 * num2)

    elif(op == '/'):
        if(num2 == 0):
            print("Error")
        else:
            print(num1 // num2)
    elif(op == '^'):
        if(num1 == num2 == 0):
            print("Error")
        else:
            print(num1**num2)
    else:
        print("Error")

end = time.time()

print("Time:" + str(end - start) + "s")
```

1 Консольный вывод

test01.txt Сложение 10000 вычислений чисел 10^{1000}
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ python3 generator.py < test01.txt > out01p.txt
$Time$: $0.6483066082000732s$
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ ./solution < test01.txt > out01c.txt
$Time$: $150 * 10^{-6} seconds$

test02.txt Вычитание 10000 вычислений чисел 10^{1000}
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ python3 generator.py < test02.txt > out02p.txt
$Time$: $0.6098551750183105s$
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ ./solution < test02.txt > out02c.txt
$Time$: $130 * 10^{-6} seconds$

test03.txt Умножение 10000 вычислений чисел 10^{1000}
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ python3 generator.py < test03.txt > out03p.txt
$Time$: $0.6098551750183105s$
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ ./solution < test03.txt > out03c.txt
$Time$: $688 * 10^{-6} seconds$

test03.txt Умножение 10000 вычислений чисел 10^{1000}
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ python3 generator.py < test03.txt > out03p.txt
$Time$: $0.6098551750183105s$
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ ./solution < test03.txt > out03c.txt
$Time$: $688 * 10^{-6} seconds$

test04.txt Деление 10000 вычислений чисел 10^{1000}
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ python3 generator.py < test04.txt > out04p.txt
$Time$: $0.588982343673706s$
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ ./solution < test04.txt > out04c.txt

$Time$: $2866 * 10^{-6} seconds$

test05.txt Возведение в степень 10^{100}10010000.
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ python3 generator.py < test05.txt > out05p.txt
$Time$: $11.316147327423096s$
$rusya@DESKTOP-93JGKCU$: $/mnt/c/Users/rusya/Desktop/lab6\_DA$ ./solution < test05.txt > out05c.txt
$Time$: $11908 * 10^{-6} seconds$

4 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я узнал способы представления длинных чисел в языке C++, реализовывать алгоритмы вычислений длинных чисел. Представление числа, как массив цифр, каждая из которых отвечает за свой разряд создаёт дополнительные трудности для произведения числовых операций.

Стоит отметить, что в языке Python3 можно работать с длинными числами. Хотя моя реализация оказалась быстрее, это не отменяет того факта, что библиотеку нужно реализовывать.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] Дональд Э. Кнут. *Искусство программирования. Том 3. Сортировка и поиск, 2-е издание.* — Правообладатель «Диалектика-Вильямс», 2018. Перевод с английского: И. В. Красиков, В. Т. Тертышный. — 834 с. (ISBN 978-5-8459-0082-1, 0-201-89685-0)

[3] *Длинная арифметика*
URL: `https://en.wikipedia.org/wiki/Длинная_арифметика` (дата обращения: 9.03.20210).

[4] *Длинная арифметика | WikiITMO*
URL: `https://clck.ru/VV5mA` (дата обращения: 9.03.2021).

[5] *Implication*
URL: `https://e-maxx.ru/algo/big_integer` (дата обращения: 9.03.2021).

[6] *Chrono in C++*
URL: `https://www.geeksforgeeks.org/chrono-in-c/` (дата обращения: 9.03.2021).