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: П. А. Мохляков
Преподаватель: Н. С. Капралов
Группа: М8О-308Б

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:

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 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 (+, -, *, $\wedge$, /, >, <, =).

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

Длинная арифметика позволяет работать с числами превышающими максимальные размеры стандартных типов. Для ее реализации релиции нужно реализации структуру в которой будут храниться числа и ряд логических и арифметических операторов.

Сами составлющие числа храняться в векторе целых чисел.

Алгоритмы операторов сложения, вычитания и умножения аналогичны привычным нам алгоритмам сложения, вычитания и умножения в столбик.

С делением все немного сложнее, дело в том, что мы должны угадывать каждую цифру ответа. Рассмотрим этот алгоритм. Пусть делимое U состоит из m+n разрядов, а делитель V состоит из n разрядов.

1. Нормализируем делимое и делитель, умнажая на число $d = \frac{BASE}{V_{n-1}+1}$
2. Инициализация цикла деления $i = m$ (для 3-6 шагов алгорима)
3. Присвоить $q = \frac{U_{i+n}*b+U_{i+n-1}}{V_{n-1}}$, $r = (U_{i+n}b + U_{i+n-1}) \bmod V_{n-1}$. Проверить выполнения неравенств $q = BASE$ или $q * V_{n-2} > b * r + U_{i+n-2}$, если удовлетворяется, то уменьшить q на 1, увеличить r на V_{n-1}, повторяем проверку при $r < b$. Таким образом мы исключаем большинство случаев, когда q больше истинного на единицу, и все случаи, когда q больше истинного на два.
4. Умножаем делитель на q и вычитаем из старших разрядов делимого.
5. Если в результате прошлого шага получили отрицательное число, то уменьшаем q на единицу, а к старшим разрядам делимого добавляем делитель.
6. Записываем q как разряд ответа, и повторяем шаги цикла до завершения деления.
7. Денормализуем ответ, разделив на число d.

Для уксорения возведения в степень мы будет не умножать основание n раз, а если это возможно, то разбивать эту задачу на две подзадачи. Если степень кратно двум, то мы делим ее на 2 и находим степень этого числа, а потом возводим ответ в квадрат. Если степень не кратна двум, то возводим число в степень $n - 1$, а потом умнажаем его на основание. Таким образом мы вмесло n умножений получаем примерно $\sqrt{n}$.

Операторы стравнения работают на поразрядном сравнении чисел.

2 Исходный код

SuffTree.cpp

```
1     #pragma once
2
3  #include <vector>
4  #include <algorithm>
5  #include <string>
6  #include <iostream>
7  #include <iomanip>
8
9  const long long BASE = 1000000000;
10 const long long POW = 9;
11
12 namespace NBigNum{
13 class TBigNum{
14 private:
15     long long Size,MaxSize;
16     std::vector<long long> Nums;
17     TBigNum ShortDiv(TBigNum other);
18     TBigNum LongDiv(TBigNum other);
19 public:
20
21     TBigNum();
22     TBigNum(long long size);
23     TBigNum(const TBigNum &other);
24     TBigNum(const char* &str);
25     ~TBigNum();
26
27     long long GetSize();
28     long long GetMaxSize();
29     int GetNum(long long pos);
30     void SetSize(long long size);
31     void SetMaxSize(long long maxsize);
32     void SetNum(long long pos,int num);
33     void FromStr(std::string str);
34     void Revers();
35     TBigNum ShiftMinus(TBigNum& other,int shift = 0);
36
37     TBigNum& operator=(const TBigNum other);
38     TBigNum& operator=(int number);
39
40     TBigNum operator+(TBigNum& other);
41     TBigNum operator-(TBigNum& other);
42     TBigNum operator-(int number);
43     TBigNum operator*(TBigNum& other);
44     TBigNum operator*(int number);
45     TBigNum operator/(TBigNum other);
46     TBigNum operator/(int number);
```

```
47      TBigNum operator^(TBigNum other);
48
49      bool operator<(const TBigNum& other);
50      bool operator>(const TBigNum& other);
51      bool operator==(const TBigNum& other);
52      bool operator<(int number);
53      bool operator>(int number);
54      bool operator==(int number);
55
56      friend std::ostream& operator<< (std::ostream &out, const TBigNum &num){
57          long long size = num.Size;
58          if(size > 0){
59              out << num.Nums[size-1];
60          }
61          for(long long i = size - 1; i > 0 ; --i){
62              out <<std::setw(POW) << std::setfill('0')<< num.Nums[i-1];
63          }
64          return out;
65      }
66
67      friend std::istream& operator>> (std::istream& in,TBigNum& num){
68          std::string str;
69          in >> str;
70          num.FromStr(str);
71          return in;
72      }
73
74
75  };
76
77
78  }//namespace BigNum
79
80  inline NBigNum::TBigNum operator "" _bn(const char* str, size_t size){
81      NBigNum::TBigNum tmp;
82      tmp.FromStr(str);
83      return tmp;
84  }
```

SuffTree.hpp	
long long GetSize()	Получение количества разрядов числа
long long GetMaxSize()	Получение максимального количества разрядов числа
int GetNum(long long pos)	Получение отдельного разряда числа
void SetSize(long long size)	Изменение значения размера числа
void SetMaxSize(long long maxsize)	Изменение значения максимального размера числа
void SetNum(long long pos,int num)	Задает значение отдельного разряда
void FromStr(std::string str)	Преобразует строку в длинное число
TBigNum ShiftMinus(TBigNum& other,int shift = 0)	Вычитает одно число из другого со сдвигом разрядов

3 Консоль

```
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB6 cat test
38943432983521435346436
354353254328383
+
90409438473849324729384738 43
2343543
-
972323
2173937
>
2
3
-
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB6 g++ main.cpp bignum.cpp -o
main
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB6 cat test | ./main
3894343333787468967 4819
90409438473849324729361303 00
false
Error
```

4 Тест производительности

10000 операций

```
Python time: 0.02045750617980957
C++ time: 0.0828969
```

100000 операций

```
Python time: 0.21204495429992676
C++ time: 0.788728
```

Мы получили сравнимое время выполнения с языком Python, где длинная арифменика втроена и отимизированна изначально.

5 Выводы

Выполнив данную лабораторную работу по курсу «Дискретный анализ», я научился обходить ограничения диапозона целых чисел в C++. Для этого необходимо использовать длинную арифметику, суть которой состоит в том, что мы работаем не с числом а массивом чисел. Длинные числа могут поддерживать все те операции, что и встроинные, но выполняться они будут дольше.

Список литературы

[1] Кнут Д. Э. *Искусство программирования. Том 2. Получисленные алгоритмы* — Издательский дом «Вильямс», 2001.