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. А. Айрапетова
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, впротивном 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 навход 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 содержатьв себе «ведущих» нулей.

1 Описание

Требуется реализовать класс для хранения «длинных» чисел и операции над ними: сложение, вычитание, умножение, деление, возведение в степень, сравнение.

Сложение и вычитания выполняются поразрядно. То есть при сложении, если появляется переполнение разряда, то оно переносится на следующий разряд.

При вычитании нужно «занять» число.

Умножение выполняется аналогично, только при выполнении сложения после умножения на 1 разряд числа слагаемое сдвигается.

Реализация деления заключается в том, чтобы угадать число, на которое умножается делитель и вычесть из исходного числа произведение делителя на угаданное число.

Возведение встепень производится многократным умножением числа самого на себя, если степень четная, чтобы сократить время работы программы, каждый раз число умножается на себя дважды.

Для сравнения двух чисел сначала сравниваются их длины, если они совпали, то сравниваются разряды.

2 Исходный код

Считываются два числа, а затем операция. Если операция не может быть выполнена, выводится Error. Реализация каждой операции описана ниже:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>

class TSuperAlg {
    public:
        static const int BASE = 10000;
        static const int RADIX = 4;
        TSuperAlg() = default;
        TSuperAlg(const std::string &s) {
            Initialize(s);
        }
        void Initialize(const std::string &str);
        friend std::istream& operator>>(std::istream &in, TSuperAlg &rhs);
        friend std::ostream& operator<<(std::ostream &out, const TSuperAlg& rhs);
        TSuperAlg operator-(const TSuperAlg &rhs) const;
        TSuperAlg operator+(const TSuperAlg &rhs) const;
        TSuperAlg operator/(const TSuperAlg &rhs) const;
        TSuperAlg operator*(const TSuperAlg &rhs) const;
        TSuperAlg Pow(int p);
        bool operator<(const TSuperAlg &rhs) const;
        bool operator>(const TSuperAlg &rhs) const;
        bool operator==(const TSuperAlg &rhs) const;
    private:
        void DeleteLeadingZeros();
        std::vector<int32_t> _data;

};
using int100500_t = TSuperAlg;
void TSuperAlg::Initialize(const std::string &str) {
    int size = static_cast<int>(str.size());
    for (int i = size - 1; i >= 0; i = i - TSuperAlg::RADIX) {
        if (i < TSuperAlg::RADIX) {
            _data.push_back(static_cast<int32_t>(atoll(str.substr(0, i + 1).c_str())));
        } else {
            _data.push_back(static_cast<int32_t>(atoll(str.substr(i - TSuperAlg::RADIX
                + 1, TSuperAlg::RADIX).c_str())));
        }
    }
    DeleteLeadingZeros();
}

TSuperAlg TSuperAlg::operator+(const TSuperAlg &rhs) const {
```

```
44      TSuperAlg res;
45      int32_t carry = 0;
46      size_t n = std::max(rhs._data.size(), _data.size());
47      res._data.resize(n);
48      for (size_t i = 0; i < n; ++i) {
49          int32_t sum = carry;
50          if (i < rhs._data.size()) {
51              sum += rhs._data[i];
52          }
53          if (i < _data.size()) {
54              sum += _data[i];
55          }
56          carry = sum / TSuperAlg::BASE;
57          res._data[i] = sum % TSuperAlg::BASE;
58      }
59      if (carry != 0) {
60          res._data.push_back(1);
61      }
62      res.DeleteLeadingZeros();
63      return res;
64  }
65  TSuperAlg TSuperAlg::operator-(const TSuperAlg &rhs) const {
66      TSuperAlg res;
67      int32_t carry = 0;
68      size_t n = std::max(rhs._data.size(), _data.size());
69      res._data.resize(n + 1, 0);
70      for (size_t i = 0; i < n; ++i) {
71          int32_t diff = _data[i] - carry;
72          if (i < rhs._data.size()) {
73              diff -= rhs._data[i];
74          }
75          carry = 0;
76          if (diff < 0) {
77              carry = 1;
78              diff += TSuperAlg::BASE;
79          }
80          res._data[i] = diff % TSuperAlg::BASE;
81      }
82      res.DeleteLeadingZeros();
83      return res;
84  }
85  TSuperAlg TSuperAlg::operator*(const TSuperAlg &rhs) const {
86      size_t n = _data.size() * rhs._data.size();
87      TSuperAlg res;
88      res._data.resize(n + 1);
89      int k = 0;
90      int r = 0;
91      for (size_t i = 0; i < _data.size(); ++i) {
92          for (size_t j = 0; j < rhs._data.size(); ++j) {
```

```
            k = rhs._data[j] * _data[i] + res._data[i + j];
            r = k / TSuperAlg::BASE;
            res._data[i + j + 1] = res._data[i + j + 1] + r;
            res._data[i + j] = k % TSuperAlg::BASE;
        }
    }
    res.DeleteLeadingZeros();
    return res;
}
 TSuperAlg TSuperAlg::operator/(const TSuperAlg &rhs) const {
    TSuperAlg res("0"), cv("0");
    res._data.resize(_data.size());
    for (int i = (int)_data.size() - 1; i >= 0; --i) {
        cv._data.insert(cv._data.begin(), _data[i]);
        if (!cv._data.back()) {
            cv._data.pop_back();
        }
        int x = 0, l = 0, r = BASE;
        while (l <= r) {
            int m = (l + r) / 2;
            TSuperAlg cur = rhs * TSuperAlg(std::to_string(m));
            if ((cur < cv) || (cur == cv)) {
                x = m;
                l = m + 1;
            } else {
                r = m - 1;
            }
        }
        res._data[i] = x;
        cv = cv - rhs * TSuperAlg(std::to_string(x));
    }
    res.DeleteLeadingZeros();
    return res;
}
TSuperAlg TSuperAlg::Pow(int p) {
    TSuperAlg res("1");
    while (p > 0) {
        if (p % 2 == 1) {
            res = res * *this;
        }
        *this = *this * *this;
        p /= 2;
    }
    return res;
}
bool TSuperAlg::operator<(const TSuperAlg &rhs) const {
    if (_data.size() != rhs._data.size()) {
        return _data.size() < rhs._data.size();
    }
```

```
142     for (int i = _data.size() - 1; i >= 0; --i) {
143         if (_data[i] != rhs._data[i]) {
144             return _data[i] < rhs._data[i];
145         }
146     }
147     return false;
148 }
149 bool TSuperAlg::operator==(const TSuperAlg &rhs) const {
150     if (_data.size() != rhs._data.size()) {
151         return false;
152     }
153     for (int i = _data.size() - 1; i >= 0; --i) {
154         if (_data[i] != rhs._data[i]) {
155             return false;
156         }
157     }
158     return true;
159 }
160 void TSuperAlg::DeleteLeadingZeros() {
161     while (!_data.empty() && _data.back() == 0) _data.pop_back();
162 }
163 
164 std::istream& operator>>(std::istream &in, TSuperAlg &rhs) {
165     std::string str;
166     in >> str;
167     rhs.Initialize(str);
168     return in;
169 }
170 std::ostream& operator<<(std::ostream &out, const TSuperAlg& rhs) {
171     if (rhs._data.empty()) {
172         out << "0";
173         return out;
174     }
175     out << rhs._data.back();
176     for (int i = rhs._data.size() - 2; i >= 0; --i) {
177         out << std::setfill('0') << std::setw(TSuperAlg::RADIX) << rhs._data[i];
178     }
179     return out;
180 }
181 
182 int main() {
183     std::ios_base::sync_with_stdio(false);
184     std::cin.tie(nullptr);
185     std::string str1, str2;
186     char action;
187     while (std::cin >> str1 >> str2 >> action) {
188         int100500_t num1(str1), num2(str2);
189         if (action == '+') {
190             int100500_t res = num1 + num2;
```

```
            std::cout << res << std::endl;
        } else if (action == '-') {
            if (num1 < num2) {
                std::cout << "Error\n";
                continue;
            }
            int100500_t res = num1 - num2;
            std::cout << res << "\n";
        } else if (action == '*') {
            int100500_t res = num1 * num2;
            std::cout << res << "\n";
        } else if (action == '/') {
            if (str2 == "0") {
                std::cout << "Error\n";
                continue;
            }
            int100500_t res = num1 / num2;
            std::cout << res << "\n";
        } else if (action == '^') {
            if (str1 == "0") {
                if (str2 == "0") {
                    std::cout << "Error\n";
                    continue;
                } else {
                    std::cout << "0\n";
                    continue;
                }
            }
            if (str1 == "1") {
                std::cout << "1\n";
                continue;
            }
            int100500_t res = num1.Pow(std::stoi(str2));
            std::cout << res << "\n";
        } else if (action == '<') {
            std::cout << ((num1 < num2) ? "true\n" : "false\n");
        } else if (action == '>') {
            std::cout << ((num2 < num1) ? "true\n" : "false\n");
        } else if (action == '=') {
            std::cout << ((num1 == num2) ? "true\n" : "false\n");
        }
    }
    return 0;
}
```

3 Консоль

```
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ g++ -pedantic -Wall -std=c++11 -Werror lab6.cpp
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ cat test1.txt
125354643466434464654113564 43
0435434345435432112313 5
+
2
3
<
12
36
-
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ ./a.out <test1.txt
125354687009869008197324795 78
true
Error
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$
```

4 Тест производительности

Тест проводился по 1000 операций для сложения, вычитания, умножения.

```
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ g++ -pedantic -Wall -std=c++11 -Werror benchmark.cpp
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ ./a.out <test2.txt >out.txt
all 2 ms
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ ./a.out <test3.txt >out.txt
all 1 ms
jane@Evgenia:/mnt/c/Files/ДА/ЛР6$ ./a.out <test4.txt >out.txt
all 2 ms
```

5 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я поработала с длинными числами, посмотрела их внутреннее представление и реализовала для них базовые арифметические операции.

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] Реализация длинной арифметики на C++ — habr.
URL: `https://habr.com/ru/post/172285/` (дата обращения: 04.06.2021).