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. С. Лагуткина
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, 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

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 в степень производится многократным умножением числа самого на себя, если степень четная, чтобы сократить время работы программы, каждый раз число умножается на себе дважды. Для сравнения двух чисел сначала сравниваются их длины, если они совпали, то сравниваются разряды.

2 Исходный код

Этапы написания кода:

1. Реализация класса «длинных» чисел
2. Реализация действий над числами
3. Осуществление ввода

Сначала считываются числа считываются как строки, затем знак операции, если операция может быть выполнена, происходит ее выполнение, иначе выводится Error.

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


namespace NSuperAlg {

    class TSuperAlg {
    public:
        static const int BASE = 10000;
        static const int RADIX = 4;

        TSuperAlg() = default;
        TSuperAlg(std::string const &str) {
            this->Initialize(str);
        }
        TSuperAlg(int n){
            if (n < BASE)
                data.push_back(n);
            else {
                for (; n; n /= BASE)
                    data.push_back(n % BASE);
            }
        }
        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);
        TSuperAlg operator-(const TSuperAlg &rhs);
        TSuperAlg operator*(const TSuperAlg &rhs) const;
        TSuperAlg operator/(const TSuperAlg &rhs);
```

```
37          TSuperAlg Power(int r);
38          bool operator<(const TSuperAlg &rhs) const;
39          bool operator==(const TSuperAlg &rhs) const;
40
41      private:
42          void DeleteLeadingZeros();
43          std::vector<int32_t> data;
44      };
45
46      using bigInt_t = TSuperAlg;
47
48
49      void TSuperAlg::Initialize(const std::string & str) {
50          int size = static_cast<int>(str.size());
51          for (int i = size - 1; i >= 0; i = i - TSuperAlg::RADIX) {
52              if (i < TSuperAlg::RADIX) {
53                  data.push_back(static_cast<int32_t>(atoll(str.substr(0, i + 1).c_str()))
                        );
54              }
55              else {
56                  data.push_back(static_cast<int32_t>(atoll(str.substr(i - TSuperAlg::
                        RADIX + 1, TSuperAlg::RADIX).c_str())));
57              }
58          }
59          DeleteLeadingZeros();
60      }
61
62      TSuperAlg TSuperAlg::operator+(const TSuperAlg &rhs){
63          TSuperAlg res;
64          int32_t carry = 0;
65          size_t n = std::max(rhs.data.size(), data.size());
66          res.data.resize(n);
67          for (size_t i = 0; i < n; ++i) {
68              int32_t sum = carry;
69              if (i < rhs.data.size()) {
70                  sum += rhs.data[i];
71              }
72              if (i < data.size()) {
73                  sum += data[i];
74              }
75              carry = sum / TSuperAlg::BASE;
76              res.data[i] = sum % TSuperAlg::BASE;
77          }
78          if (carry != 0) {
79              res.data.push_back(1);
80          }
81          res.DeleteLeadingZeros();
82          return res;
83      }
```

```
84
85      TSuperAlg TSuperAlg::operator-(const TSuperAlg &rhs){
86          TSuperAlg res;
87          int32_t carry = 0;
88          size_t n = std::max(rhs.data.size(), data.size());
89          res.data.resize(n + 1, 0);
90          for (size_t i = 0; i < n; ++i) {
91              int32_t diff = data[i] - carry;
92              if (i < rhs.data.size()) {
93                  diff -= rhs.data[i];
94              }
95
96              carry = 0;
97              if (diff < 0) {
98                  carry = 1;
99                  diff += TSuperAlg::BASE;
100             }
101             res.data[i] = diff % TSuperAlg::BASE;
102         }
103         res.DeleteLeadingZeros();
104         return res;
105     }
106
107     TSuperAlg TSuperAlg::operator*(const TSuperAlg & rhs) const {
108         size_t n = data.size() * rhs.data.size();
109         TSuperAlg res;
110         res.data.resize(n + 1);
111
112         int k = 0;
113         int r = 0;
114         for (size_t i = 0; i < data.size(); ++i){
115             for (size_t j = 0; j < rhs.data.size(); ++j) {
116                 k = rhs.data[j] * data[i] + res.data[i + j];
117                 r = k / TSuperAlg::BASE;
118                 res.data[i + j + 1] = res.data[i + j + 1] + r;
119                 res.data[i + j] = k % TSuperAlg::BASE;
120             }
121         }
122         res.DeleteLeadingZeros();
123         return res;
124     }
125
126     TSuperAlg TSuperAlg::operator/(const TSuperAlg &rhs) {
127         TSuperAlg res, cv = TSuperAlg(0);
128         res.data.resize(data.size());
129
130         for (int i = (int)data.size() - 1; i >= 0; --i) {
131             cv.data.insert(cv.data.begin(), data[i]);
132             if (!cv.data.back())
```

```
133             cv.data.pop_back();
134         int x = 0, l = 0, r = BASE;
135         while (l <= r) {
136             int m = (l + r) / 2;
137             TSuperAlg cur = rhs * TSuperAlg(std::to_string(m));
138             if ((cur < cv)||(cur == cv)) {
139                 x = m;
140                 l = m + 1;
141             }
142             else {
143                 r = m - 1;
144             }
145         }
146         res.data[i] = x;
147         cv = cv - rhs * TSuperAlg(std::to_string(x));
148     }
149     res.DeleteLeadingZeros();
150     return res;
151 
152 }
153 
154 TSuperAlg TSuperAlg::Power(int p){
155     TSuperAlg res(1);
156     while (p > 0) {
157         if (p % 2 == 1)
158             res = res * (*this);
159         (*this) = (*this) * (*this);
160         p /= 2;
161     }
162     return res;
163 }
164 
165 bool TSuperAlg::operator<(const TSuperAlg &rhs) const {
166     if (data.size() != rhs.data.size()) {
167         return data.size() < rhs.data.size();
168     }
169 
170     for (int32_t i = data.size() - 1; i >= 0; --i) {
171         if (data[i] != rhs.data[i]) {
172             return data[i] < rhs.data[i];
173         }
174     }
175     return false;
176 }
177 
178 bool TSuperAlg::operator==(const TSuperAlg &rhs) const {
179     if (data.size() != rhs.data.size()) {
180         return false;
181     }
```

```
182
183         for (int32_t i = data.size() - 1; i >= 0; --i) {
184             if (data[i] != rhs.data[i]) {
185                 return false;
186             }
187         }
188         return true;
189     }
190
191     void TSuperAlg::DeleteLeadingZeros(){
192         while (!data.empty() && data.back() == 0) {
193             data.pop_back();
194         }
195     }
196
197     std::istream& operator>>(std::istream &in, TSuperAlg &rhs) {
198         std::string str;
199         in >> str;
200         rhs.Initialize(str);
201         return in;
202     }
203
204     std::ostream& operator<<(std::ostream &out, const TSuperAlg &rhs) {
205         if (rhs.data.empty()) {
206             out << "0";
207             return out;
208         }
209         out << rhs.data.back();
210         for (int i = rhs.data.size() - 2; i >= 0; --i) {
211             out << std::setfill('0') << std::setw(TSuperAlg::RADIX) << rhs.data[i];
212         }
213         return out;
214     }
215
216 } // namespace NSuperAlg
217
218 int main(){
219     std::string strNum1, strNum2;
220     NSuperAlg::bigInt_t zero("0");
221     NSuperAlg::bigInt_t one("1");
222     char action;
223     auto start = std::chrono::steady_clock::now();
224     while (std::cin >> strNum1 >> strNum2 >> action) {
225         NSuperAlg::bigInt_t num1(strNum1);
226         NSuperAlg::bigInt_t num2(strNum2);
227         if (action == '+') {
228             NSuperAlg::bigInt_t res = num1 + num2;
229             std::cout << res << "\n";
230         }
```

```
231         else if (action == '-') {
232             if (num1 < num2) {
233                 std::cout << "Error\n";
234                 continue;
235             }
236             NSuperAlg::bigInt_t res = num1 - num2;
237             std::cout << res << "\n";
238         }
239         else if (action == '*') {
240             NSuperAlg::bigInt_t res = num1 * num2;
241             std::cout << res << "\n";
242         }
243         else if (action == '/') {
244             if (num2 == zero) {
245                 std::cout << "Error\n";
246                 continue;
247             }
248             NSuperAlg::bigInt_t res = num1 / num2;
249             std::cout << res << "\n";
250         }
251         else if (action == '^') {
252             if (num1 == zero) {
253                 if (num2 == zero) {
254                     std::cout << "Error\n";
255                     continue;
256                 }
257                 else {
258                     std::cout << "0" << "\n";
259                 }
260             }
261             else if (num1 == one) {
262                 std::cout << "1" << "\n";
263             }
264             else {
265                 NSuperAlg::bigInt_t res = num1.Power(std::stoi(strNum2));
266                 std::cout << res << "\n";
267             }
268         }
269 
270         else if (action == '<') {
271             std::cout << ((num1 < num2) ? "true" : "false") << "\n";
272         }
273         else if (action == '>') {
274             std::cout << ((num2 < num1) ? "true" : "false") << "\n";
275         }
276         else if (action == '=') {
277             std::cout << ((num1 == num2) ? "true" : "false") << "\n";
278         }
279         else { std::cout << "Error\n"; }
```

```
280     }
281     auto finish = std::chrono::steady_clock::now();
282     auto dur1 = finish - start;
283 
284     return 0;
285 }
```

3 Консоль

```
maria@DESKTOP-6CRUDOR:~$ g++ -pedantic -Wall -std=c++14 -Werror -Wno-sign-compare
da6.cpp -o da6
maria@DESKTOP-6CRUDOR:~$ ./da6
125354643466434464654113564 43
0435434345435432112313 5
+
125354687009869008197324795 78
21553135151321221212351
123232321
-
2155313515132109798003 0
2
3
<
true
12
36
-
Error
```

4 Тест производительности

Для теста производительности использовалась библиотека GMP. Тест проводился по 100000000 операций для сложения, вычитания, умножения.

```
maria@DESKTOP-6CRUDOR:~$ ./da6
Big int sum time: 7.91448 sec
Gmp     sum time: 1.82999 sec

Big int sub time: 8.65126 sec
Gmp     sub time: 2.53862 sec

Big int mult time: 15.2799 sec
Gmp     mult time: 1.77125 sec
```

Как видно, данная реализация проигрывает в производительности GMP, так как не является наиболее эффективной.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я реализовала алгоритмы работы с «длинными» числами, посмотрела их внутреннее представление. Реализации операций в программе не являются единственными, так для умножения можно использовать алгоритм Карацубы, а не умножение в столбик.

Список литературы

[1] *MAXimal::algo::длинная арифметика*

`https://e-maxx.ru/algo/big_integer (дата обращения 10.03.2021)`