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: В. С. Епанешников
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 зменьшего 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. Реализацияделения заключается в том, чтобы угадать число, на которое умножается делитель ивычесть из исходного числа произведение делителя на угаданное число. Возведение встепень производится многократным умножением числа самого на себя, если степеньчетная, чтобы сократить время работы программы, каждый раз число умножаетсяна себе дважды. Для сравнения двух чисел сначала сравниваются их длины, если они совпали, то сравниваются разряды.

2 Исходный код

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>

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 = uint64_t;
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 {
```

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

```
92
93    res.DeleteLeadingZeros();
94    return res;
95  }
96
97  TSuperAlg TSuperAlg::operator*(const TSuperAlg &rhs) const {
98    size_t n = _data.size() * rhs._data.size();
99    TSuperAlg res;
100   res._data.resize(n + 1);
101
102   int k = 0;
103   int r = 0;
104   for (size_t i = 0; i < _data.size(); ++i) {
105     for (size_t j = 0; j < rhs._data.size(); ++j) {
106       k = rhs._data[j] * _data[i] + res._data[i + j];
107       r = k / TSuperAlg::BASE;
108       res._data[i + j + 1] = res._data[i + j + 1] + r;
109       res._data[i + j] = k % TSuperAlg::BASE;
110     }
111   }
112   res.DeleteLeadingZeros();
113   return res;
114 }
115
116 TSuperAlg TSuperAlg::operator/(const TSuperAlg &rhs) const {
117   TSuperAlg res("0"), cv("0");
118   res._data.resize(_data.size());
119
120   for (int i = (int)_data.size() - 1; i >= 0; --i) {
121     cv._data.insert(cv._data.begin(), _data[i]);
122     if (!cv._data.back()) {
123       cv._data.pop_back();
124     }
125     int x = 0, l = 0, r = BASE;
126     while (l <= r) {
127       int m = (l + r) / 2;
128       TSuperAlg cur = rhs * TSuperAlg(std::to_string(m));
129       if ((cur < cv) || (cur == cv)) {
130         x = m;
131         l = m + 1;
132       }
133       else {
134         r = m - 1;
135       }
136     }
137     res._data[i] = x;
138     cv = cv - rhs * TSuperAlg(std::to_string(x));
139   }
140   res.DeleteLeadingZeros();
```

```
141   return res;
142 }
143
144 TSuperAlg TSuperAlg::Pow(int p) {
145   TSuperAlg res("1");
146   while (p > 0) {
147     if (p % 2 == 1) {
148       res = res * *this;
149     }
150     *this = *this * *this;
151     p /= 2;
152   }
153   return res;
154 }
155
156
157 bool TSuperAlg::operator<(const TSuperAlg &rhs) const {
158   if (_data.size() != rhs._data.size()) {
159     return _data.size() < rhs._data.size();
160   }
161
162   for (int i = _data.size() - 1; i >= 0; --i) {
163     if (_data[i] != rhs._data[i]) {
164       return _data[i] < rhs._data[i];
165     }
166   }
167   return false;
168 }
169
170 bool TSuperAlg::operator==(const TSuperAlg &rhs) const {
171   if (_data.size() != rhs._data.size()) {
172     return false;
173   }
174
175   for (int i = _data.size() - 1; i >= 0; --i) {
176     if (_data[i] != rhs._data[i]) {
177       return false;
178     }
179   }
180   return true;
181 }
182
183 void TSuperAlg::DeleteLeadingZeros() {
184   while (!_data.empty() && _data.back() == 0) _data.pop_back();
185 }
186
187 std::istream& operator>>(std::istream &in, TSuperAlg &rhs) {
188   std::string str;
189   in >> str;
```

```
190   rhs.Initialize(str);
191   return in;
192 }
193 
194 std::ostream& operator<<(std::ostream &out, const TSuperAlg& rhs) {
195   if (rhs._data.empty()) {
196     out << "0";
197     return out;
198   }
199 
200   out << rhs._data.back();
201   for (int i = rhs._data.size() - 2; i >= 0; --i) {
202     out << std::setfill('0') << std::setw(TSuperAlg::RADIX) << rhs._data[i];
203   }
204   return out;
205 }
206 
207 
208 int main() {
209   std::ios_base::sync_with_stdio(false);
210   std::cin.tie(nullptr);
211 
212   std::string str1, str2;
213   char action;
214   while (std::cin >> str1 >> str2 >> action) {
215     int100500_t num1(str1), num2(str2);
216     if (action == '+') {
217       int100500_t res = num1 + num2;
218       std::cout << res << std::endl;
219     }
220     else if (action == '-') {
221       if (num1 < num2) {
222         std::cout << "Error\n";
223         continue;
224       }
225       int100500_t res = num1 - num2;
226       std::cout << res << "\n";
227     }
228     else if (action == '*') {
229       int100500_t res = num1 * num2;
230       std::cout << res << "\n";
231     }
232     else if (action == '/') {
233       if (str2 == "0") {
234         std::cout << "Error\n";
235         continue;
236       }
237       int100500_t res = num1 / num2;
238       std::cout << res << "\n";
```

```
239     }
240     else if (action == '^') {
241       if (str1 == "0") {
242         if (str2 == "0") {
243           std::cout << "Error\n";
244           continue;
245         }
246         else {
247           std::cout << "0\n";
248           continue;
249         }
250       }
251       if (str1 == "1") {
252         std::cout << "1\n";
253         continue;
254       }
255       int100500_t res = num1.Pow(std::stoi(str2));
256       std::cout << res << "\n";
257     }
258     else if (action == '<') {
259       std::cout << ((num1 < num2) ? "true\n" : "false\n");
260     }
261     else if (action == '>') {
262       std::cout << ((num2 < num1) ? "true\n" : "false\n");
263     }
264     else if (action == '=') {
265       std::cout << ((num1 == num2) ? "true\n" : "false\n");
266     }
267   }
268   return 0;
269 }
```

3 Консоль

```
MacBook:solution vladislove$ make
g++ -std=c++17 -pedantic -g -Wall -Wextra -Wno-unused-variable main.o super_alg.o
-o solution
MacBook:solution vladislove$ ./solution
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.

```
MacBook:solution vladislove$ ./da6Big
SuperAlg sum time: 7.91448 sec
Gmp sum time: 1.82999 sec
SuperAlg sub time: 8.65126 sec
Gmp sub time 2.53862 sec
SuperAlg mult time: 15.2799 sec
Gmp mult time: 1.77125 sec
```

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

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я реализовал алгоритмы работы с «длинными» числами, посмотрел 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*
URL: `https://e-maxx.ru/algo/big_intege` (дата обращения 25.04.2021).