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: К. А. Калугин
Преподаватель: А. А. Кухтичев
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №6

Задача:

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.

1 Описание

Основная идея реализации калькулятора для длинных чисел состоит в выполнении операций «в столбик», то есть поразрядной обработки каждого из чисел.

2 Исходный код

Во входных строчках располагаются числа и операции между ними. Программа считывает числа и необходимую операцию, после чего вызывает соответствующую функцию. Числа хранятся в виде векторов, причем разряд равен 10000. Индексация массивов инвертирована, для удобства выполнения операций. Все операции выполняются «в столбик». Интерес представляет только возведение в степень, которое оптимизирует количество умножений числа путем представления степени в виде $2^n + k$.

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <math.h>
5  #include <algorithm>
6  #include <ctime>
7  using namespace std;
8  vector <int> dividers;
9
10 //
11 bool read (vector <int> & number) {
12     string buffer;
13     cin >> buffer;
14     if ((buffer [0] > '9') || (buffer [0] < '0')) {
15         return true;
16     }
17     while (buffer.length() > 0) {
18         int n = 0;
19         int ib = 4;
20         if (buffer.length () < 4) {
21             ib = buffer.length ();
22         }
23         for (int i = 0; i < ib; i++) {
24             n += (buffer [buffer.length () - 1] - '0') * (pow(10, i));
25             buffer.pop_back ();
26         }
27         number.push_back (n);
28     }
29     for (int i = number.size () - 1; i > 0; i --) {
30         if (number [i] == 0) {
31             number.pop_back ();
32         }
33         else {
34             return false;
35         }
36     }
37     return false;
38 }
39
```

```
40 //
41 void write (vector <int> & number) {
42     int k = 0;
43     for (int i = number.size () - 1; i >= 0; i --) {
44         if (i + 1 == number.size () - k) {
45             if ((number [i] != 0) || (i == 0)) {
46                 cout << number [i];
47             }
48             else {
49                 k ++;
50             }
51         }
52         else {
53             if ((0 <= number [i]) && (number [i] <= 9)) {
54                 cout << "000" << number [i];
55             }
56             else if ((10 <= number [i]) && (number [i] <= 99)) {
57                 cout << "00" << number [i];
58             }
59             else if ((100 <= number [i]) && (number [i] <= 999)) {
60                 cout << "0" << number [i];
61             }
62             else {
63                 cout << number [i];
64             }
65         }
66 
67     }
68     cout << endl;
69 }
70 
71 //    1:"1>2" 2:"1<2" 3:"1=2"
72 int comparsion (vector <int> & first, vector <int> & second) {
73     int fs = first.size ();
74     int ss = second.size ();
75     for (int i = first.size () - 1; i >= 0; i --) {
76         if (first [i] != 0) {
77             break;
78         }
79         else {
80             fs --;
81         }
82     }
83 
84     for (int i = second.size () - 1; i >= 0; i --) {
85         if (second [i] != 0) {
86             break;
87         }
88         else {
```

```
89              ss --;
90          }
91      }
92
93      if (fs > ss) {
94          return 1;
95      }
96      else if (fs < ss) {
97          return 2;
98      }
99      else {
100         for (int i = fs - 1; i >= 0; i --) {
101             if (first [i] > second [i]) {
102                 return 1;
103             }
104             else if (first [i] < second [i]) {
105                 return 2;
106             }
107         }
108         return 3;
109     }
110 }
111
112 //
113 void lplus (vector <int> & first, vector <int> & second) {
114     for (int i = 0; i < max(first.size (), second.size ()); i ++) {
115         if (first.size () == i) {
116             first.push_back (0);
117         }
118         if (second.size () - 1 < i){
119             first [i] += 0;
120         }
121         else {
122             first [i] += second [i];
123         }
124         if (first [i] > 9999) {
125             first [i] -= 10000;
126             if (first.size () == i + 1) {
127                 first.push_back (0);
128             }
129             first [i + 1] ++;
130         }
131     }
132     write (first);
133 }
134
135 //
136 void lminus (vector <int> & first, vector <int> & second) {
137
```

```
138     if (comparsion(first, second) == 2) {
139         cout << "Error" << endl;
140         return;
141     }
142     else {
143         for (int i = 0; i < first.size (); i ++) {
144             if (second.size () - 1 < i) {
145                 first [i] -= 0;
146             }
147             else {
148                 first [i] -= second [i];
149             }
150             if (first [i] < 0) {
151                 first [i] += 10000;
152                 first [i + 1] -= 1;
153             }
154         }
155         write (first);
156     }
157 }
158 
159 //
160 void lmult (vector <int> & first, vector <int> & second) {
161     vector <int> result;
162 
163     for (int i = 0; i < first.size() + second.size (); i ++) {
164         result.push_back (0);
165     }
166 
167     for (int i = 0; i < first.size(); i ++) {
168         for (int j = 0; j < second.size (); j ++) {
169             result [i + j] += first [i] * second [j];
170             if (result [i + j] > 9999) {
171                 result [i + j + 1] += result [i + j] / 10000;
172                 result [i + j] -= (result [i + j] / 10000) * 10000;
173             }
174         }
175     }
176 
177     write (result);
178 }
179 
180 vector <int> binDiv (vector <int> & first, vector <int> & second, int & divider) {
181     int l = 0;
182     int r = 10000;
183     vector <int> result;
184     for (int i = 0; i < second.size () + 1; i ++) {
185         result.push_back (0);
186     }
```

```
187
188     int comp;
189
190     while (1) {
191         divider = dividers [(l + r) / 2];
192
193         result [second.size ()] = 0;
194         for (int i = 0; i < second.size(); i ++) {
195             result [i] = second [i] * divider;
196             if (i) {
197                 if (result [i - 1] > 9999) {
198                     result [i] += result [i - 1] / 10000;
199                     result [i - 1] -= (result [i - 1] / 10000) * 10000;
200                 }
201             }
202
203         }
204
205         if (result [second.size() - 1] > 9999) {
206             result [second.size()] += result [second.size() - 1] / 10000;
207             result [second.size() - 1] -= (result [second.size() - 1] / 10000) * 10000;
208         }
209
210         comp = comparsion (first, result);
211
212         if (comp == 1) {
213             if (l != (l + r) / 2) {
214                 l = (l + r) / 2;
215             }
216             else {
217                 if (result [result.size() - 1] == 0) {
218                     int i = result.size() - 1;
219                     while (result [i] == 0) {
220                         result.pop_back ();
221                         i --;
222                     }
223                 }
224
225                 return result;
226             }
227         }
228         else if (comp == 2) {
229             r = (l + r) / 2;
230         }
231         else if (comp == 3) {
232             if (result [result.size() - 1] == 0) {
233                 int i = result.size() - 1;
234                 while (result [i] == 0) {
235                     result.pop_back ();
```

```
                    i --;
                }
            }
            return result;
        }
    }
}

//
void ldiv (vector <int> & first, vector <int> & second) {
    vector <int> result;
    vector <int> buffer;
    vector <int> workplace;
    int d;
    int comp;
    bool flag = false;

    for (int i = second.size() - 1; i >= 0; i --) {
        workplace.push_back (first[first.size () - 1 - i ]);
    }

    for (int i = first.size () - second.size () - 1; i >= -1; i --) {
        while (1) {
            comp = comparsion (workplace, second);
            if (comp == 1) {
                break;
            }
            else if (comp == 2) {
                result.push_back (0);
                if (i >= 0) {
                    workplace.insert (workplace.begin(), first [i]);
                    i --;
                }
                else {
                    flag = true;
                    break;
                }
            }
            else if (comp == 3) {
                break;
            }
        }

        if (flag) {
            break;
        }

        buffer = binDiv (workplace, second, d);
        result.push_back (d);
```

```
285         for (int j = 0; j < workplace.size (); j ++) {
286             if (buffer.size () == j) {
287                 workplace [j] -= 0;
288             }
289             else {
290                 workplace [j] -= buffer [j];
291             }
292             if (workplace [j] < 0) {
293                 workplace [j] += 10000;
294                 workplace [j + 1] -= 1;
295             }
296         }
297         while ((workplace.size () > 0) && (workplace [workplace.size () - 1] == 0)) {
298             workplace.pop_back ();
299         }
300         if (i >= 0) {
301             workplace.insert (workplace.begin(), first [i]);
302         }
303         else {
304             break;
305         }
306         buffer.clear ();
307     }
308     reverse (result.begin (), result.end ());
309     write (result);
310 }
311 
312 //
313 void sdiv (vector <int> & first, int second) {
314     vector <int> result;
315     int current = 0;
316     int overBase = 0;
317     for (int i = 0; i < int (first.size ()); i ++) {
318         result.push_back (0);
319     }
320     for (int i = first.size () - 1; i >= 0; i --) {
321         current = overBase * 10000 + first [i];
322         result [i] = current / second ;
323         overBase = result [i];
324         overBase *= second;
325         overBase = current - overBase;
326     }
327     int i = result.size () - 1;
328     while ((i > 0) && (result [i] == 0)) {
329         result.pop_back ();
330         i --;
331     }
332     write (result);
333 }
```

```
334
335 vector <int> multiplication (vector <int> & first, vector <int> & second) {
336     vector <int> result;
337
338     for (int i = 0; i < first.size() + second.size (); i ++) {
339         result.push_back (0);
340     }
341
342     for (int i = 0; i < first.size(); i ++) {
343         int next = 0;
344         for (int j = 0; j < second.size (); j ++) {
345             int curr = result [i + j] + next + first [i] * second [j];
346             if (curr >= 10000) {
347                 next = curr / 10000;
348                 result [i + j] = curr - next * 10000;
349             }
350             else {
351                 result [i + j] = curr;
352                 next = 0;
353             }
354         }
355         result [i + second.size ()] += next;
356     }
357     int i = result.size () - 1;
358     while ((i > 0) && (result [i] == 0)) {
359         result.pop_back ();
360         i --;
361     }
362
363     return result;
364 }
365
366 vector <int> division (vector <int> & first, int second) {
367     vector <int> result;
368     int current = 0;
369     int overBase = 0;
370     for (int i = 0; i < int (first.size ()); i ++) {
371         result.push_back (0);
372     }
373     for (int i = first.size () - 1; i >= 0; i --) {
374         current = overBase * 10000 + first [i];
375         result [i] = current / second ;
376         overBase = result [i];
377         overBase *= second;
378         overBase = current - overBase;
379     }
380     int i = result.size () - 1;
381     while ((i > 0) && (result [i] == 0)) {
382         result.pop_back ();
```

```
383         i --;
384     }
385     return result;
386 }
387 
388 //
389 void ldegr (vector <int> & first, vector <int> & second) {
390     vector <int> result;
391     if ((second.size () == 1) && (second [0] == 0) && (first.size () == 1) && (first
            [0] == 0)) {
392         cout << "Error" << endl;
393         return;
394     }
395 
396     if ((second.size () == 1) && (second [0] == 0)) {
397         cout << "1" << endl;
398         return;
399     }
400 
401     if ((first.size () == 1) && ((first [0] == 0) || (first [0] == 1))) {
402         result.push_back (first [0]);
403         write (result);
404         return;
405     }
406 
407     vector <int> part = {1};
408     result = first;
409     while ((second.size () != 1) || (second [0] != 0)) {
410         if (second [0] % 2 == 1) {
411             part = multiplication (part, result);
412         }
413         result = multiplication (result, result);
414         second = division (second, 2);
415     }
416     write (part);
417     return;
418 }
419 
420 int main () {
421 
422     vector <int> first;
423     vector <int> second;
424 
425     for (int i = 0; i <= 10000; i ++) {
426         dividers.push_back (i);
427     }
428 
429     while (cin) {
430         if (read (first)) {
```

```
            break;
        }
        if (read (second)) {
            break;
        }
        char sign = 'q';
        cin >> sign;
        if (sign == '+') {
            lplus (first, second);
        }
        else if (sign == '-') {
            lminus (first, second);
        }
        else if (sign == '*') {
            lmult (first, second);
        }
        else if (sign == '/') {
            if ((second [0] == 0) && (second.size () == 1)) {
                cout << "Error" << endl;
            }
            else {
                int comp = comparsion (first, second);
                if (comp == 1) {
                    if (second.size () == 1) {
                        sdiv (first, second [0]);
                    }
                    else {
                        ldiv(first, second);
                    }
                }
                else if (comp == 2) {
                    cout << "0" << endl;
                }
                else if (comp == 3) {
                    cout << "1" << endl;
                }
            }

        }
        else if (sign == '^') {
            ldegr (first, second);
        }
        else if (sign == '>') {
            if (comparsion (first, second) == 1) {
                cout << "true" << endl;
            }
            else {
                cout << "false" << endl;
            }
```

```
        }
        else if (sign == '<') {
            if (comparsion (first, second) == 2) {
                cout << "true" << endl;
            }
            else {
                cout << "false" << endl;
            }
        }
        else if (sign == '=') {
            if (comparsion (first, second) == 3) {
                cout << "true" << endl;
            }
            else {
                cout << "false" << endl;
            }
        }
        else {
            break;
        }
        first.clear ();
        second.clear ();
    }
    return 0;
}
}
```

3 Консоль

```
PS C:\VSC\DA>g++ .\Lab6.cpp
PS C:\VSC\DA>.\a.exe
38943432983521435346436
354353254328383
+
389434333378746896748 19
904094384738493247293847 3843
2343543
-
90409438473849324729361303 00
972323
2173937
>
false
2
3
-
Error
```

4 Тест производительности

Сравним время работы моей программы и библиотеки GNU MP. Для основных операций протестируем на числах длины 10^3, количество которых будет 10^3 и 10^4.

```
Умножение
PS C:\VSC\DA>./s* <tests/1.in
My program: 537.978ms
PS C:\VSC\DA>./a* <tests/1.in
GMP: 59.782ms
PS C:\VSC\DA>./s* <tests/1.in
My program: 5732.134ms
PS C:\VSC\DA>./a* <tests/1.in
GMP: 539.434ms
Деление
PS C:\VSC\DA>./s* <tests/1.in
My program: 95.534ms
PS C:\VSC\DA>./a* <tests/1.in
GMP: 65.568ms
PS C:\VSC\DA>./s* <tests/1.in
My program: 845.069ms
PS C:\VSC\DA>./a* <tests/1.in
GMP: 617.876ms
Сумма
PS C:\VSC\DA>./s* <tests/1.in
My program: 47.629ms
PS C:\VSC\DA>./a* <tests/1.in
GMP: 62.614ms
PS C:\VSC\DA>./s* <tests/1.in
My program: 397.976ms
PS C:\VSC\DA>./a* <tests/1.in
GMP: 532.170ms
```

Из-за квадратичной сложности умножения, моя программа работает намного медленнее. Также отстаёт в делении. Однако в сложении выигрывает, из-за использование 10000-ой системы счисления

5 Выводы

При выполнении данной работы я научился работать с очень большими числами при написании программ, а также оценил удобство использования отдельных функций для различных целей. Кроме того, я узнал интересный алгоритм оптимизации возведения числа в большую степень.

Список литературы

[1] URL: https://habr.com/ru/post/172285/ (дата обращения: 23.06.2021).