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. М. Воронов
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б-19

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 ().
5. Деление (/).

В случае возникновения переполнения в результате вычислений, попытки вычесть изменьшего 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, 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.

1 Описание

Так как длинные числа не влезают в существующие типы, можно использовать идею представления их в массиве, как сказано в [1].

В каждой ячейке массива мы будем хранить, например, по 4 цифры числа, то есть как бы работать в 10000 системе исчисления. Для реализации операций над такими числами достаточно вспомнить школьную математику. Сложение, вычитание и деление реализованы по принципу «в столбик» , умножение по принципу произведения многочленов, а возведение в степень - деления на 2 (бинарное возведение в степень). Для удобства реализации многих операций число хранится задом-наперёд.

2 Исходный код

Программа состоит из двух файлов: main.cpp и calc.hpp. В первом реализован интерфейс взаимодействия с пользователем и вызова соответствующих операторов. Во втором реализован класс TLongNumbers, который имеет поля SIZEBLOCK(количество цифр в ячейке массива), BASE(система счисления) и вектор BigNumber(само число). Тут же перегружены все нужные нам операторы.

```
1 class TLongNumbers {
2    private:
3        static const int SIZEBLOCK = 4;
4        static const int BASE = 10000;
5        vector<int64_t> BigNumber;
```

calc.hpp	
void DeleteZeros()	Функция удаления ведущих нулей
TLongNumbers ()	Конструктор по умолчанию
TLongNumbers (const TLongNumbers &a)	Конструктор копирования
bool Null()	Функция, которая проверяет, равно ли нулю длинное число
friend istream & operator » (istream & in, TLongNumbers & num)	Перегрузка оператора ввода
friend ostream & operator « (ostream & out, const TLongNumbers & num)	Перегрузка оператора вывода
friend TLongNumbers operator + (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора сложения
friend TLongNumbers operator - (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора вычитания
friend bool operator > (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора больше
friend bool operator < (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора меньше
friend bool operator == (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора сравнения

friend TLongNumbers operator * (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора умножения
TLongNumbers & operator = (const TLongNumbers &a)	Перегрузка оператора присваивания
friend TLongNumbers operator ∧ (const TLongNumbers &a, TLongNumbers b)	Перегрузка оператора возведения в степень
friend TLongNumbers operator / (const TLongNumbers &a, const TLongNumbers &b)	Перегрузка оператора деления

main.cpp

```cpp
#include <iostream>
#include <string>
#include <vector>
#include "calc.hpp"

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    char op;
    TLongNumbers num1;
    TLongNumbers num2;
    while (cin >> num1 >> num2 >> op) {
        if (op == '+') {
            cout << num1 + num2 << "\n";
        } else if (op == '-') {
            if (num2 > num1) {
                cout << "Error\n";
            } else {
                cout << num1 - num2 << "\n";
            }
        } else if (op == '>') {
            if (num1 > num2) {
                cout << "true\n";
            } else {
                cout << "false\n";
            }
        } else if (op == '<') {
            if (num1 < num2) {
                cout << "true\n";
            } else {
                cout << "false\n";
```

```
35              }
36          } else if (op == '=') {
37              if (num1 == num2) {
38                  cout << "true\n";
39              } else {
40                  cout << "false\n";
41              }
42          } else if (op == '*') {
43              cout << num1 * num2 << "\n";
44          } else if (op == '/') {
45              if (num2.Null()) {
46                  cout << "Error\n";
47              } else {
48                  cout << num1 / num2 << "\n";
49              }
50          } else if (op == '^') {
51              if (num2.Null() && num1.Null()) {
52                  cout << "Error\n";
53              } else {
54                  cout << (num1 ^ num2) << "\n";
55              }
56          }
57      }
58      return 0;
59  }
```

3 Консоль

```
kirill@kirill-G3-3779:~/DA/lab6$ cat test1
454546546545645
123121
+
897897987987
1
/
555555
1000000
*
36363989855
1
>
888
4
^
kirill@kirill-G3-3779:~/DA/lab6$ ./s* <test1
454546546668766
897897987987
555555000000
true
621801639936
```

4 Тест производительности

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

Умножение

```
kirill@kirill-G3-3779:~/DA/lab6$ ./s* <tests/1.in
My program: 537.978ms
kirill@kirill-G3-3779:~/DA/lab6$ ./a* <tests/1.in
GMP: 59.782ms

kirill@kirill-G3-3779:~/DA/lab6$ ./s* <tests/1.in
My program: 5732.134ms
kirill@kirill-G3-3779:~/DA/lab6$ ./a* <tests/1.in
GMP: 539.434ms
```

Деление

```
kirill@kirill-G3-3779:~/DA/lab6$ ./s* <tests/1.in
My program: 95.534ms
kirill@kirill-G3-3779:~/DA/lab6$ ./a* <tests/1.in
GMP: 65.568ms

kirill@kirill-G3-3779:~/DA/lab6$ ./s* <tests/1.in
My program: 845.069ms
kirill@kirill-G3-3779:~/DA/lab6$ ./a* <tests/1.in
GMP: 617.876ms
```

Сумма

```
kirill@kirill-G3-3779:~/DA/lab6$ ./s* <tests/1.in
My program: 47.629ms
kirill@kirill-G3-3779:~/DA/lab6$ ./a* <tests/1.in
GMP: 62.614ms

kirill@kirill-G3-3779:~/DA/lab6$ ./s* <tests/1.in
My program: 397.976ms
kirill@kirill-G3-3779:~/DA/lab6$ ./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, я думаю, это из-за того, что в моей программе используется 10000 система счисления.

5 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я научился работать с очень большими числами в компьютере, реализовал все основные операции с ними, а также вспомнил школьную математику. Работа с большими числами имеет важную роль в развитии человечества, особенно в нынешнее время. Наши возможности растут, требуя всё больше трудоёмких вычислений. Именно здесь нам и поможет "длинная"арифметика.

Список литературы

[1] *С.М. Окулов. "Длинная"арифметика.*
URL: `http://comp-science.narod.ru/DL-AR/okulov.htm` (дата обращения: 07.03.2021)

[2] *Бинарное возведение в степень.*
URL: `https://e-maxx.ru/algo/binary_pow` (дата обращения: 09.03.2021).