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. П. Будникова
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:
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.

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

Как сказано в [1]: «Длинная арифметика — выполняемые с помощью вычислительной машины арифметические операции (сложение, вычитание, умножение, деление, возведение в степень, элементарные функции) над числами, разрядность которых превышает длину машинного слова данной вычислительной машины. Эти операции реализуются не аппаратно, а программно, с использованием базовых аппаратных средств работы с числами меньших порядков.»

Сложение:

При сложении выделяются разряды числа и происходит поразрядное сложение («сложение в столбик») Если разряд переполняется, т. е. становится больше, чем основание системы счисления, то остаток от деления на основание суммируется со следующим, более старшим разрядом.

Вычитание:

Перед вычитание числа сравниваются, если уменьшаемое меньше, чем вычитаемое, то бросается исключение. На экран выводится Error. Если уменьшаемое больше, чем вычитаемое, то вычитание производится аналогично сложению.

Умножение:

При умножении производится сравнение чисел и дальше происходит поразрядное умножение («умножение в столбик» большего числа на меньшее.

Деление:

При делении числа сравниваются, если делимое меньше, чем делитель, то бросается исключение. Если происходит деление на ноль, также бросается исключение. Деление происходит на основе «деления в слолбик» При делении, бинарным поиском ищется такой максимальный множитель, чтобы при умножении на делитель, полученное произведение не превышало числа, полученного с помощью старших разрядов. Далее происходит вычитание и прибавление следующего разряда делимого.

Возведение в степень:

Если происходит попытка возведения нуля в нулевую степень, то бросается исключение. Если степень четная, то производится деление степени на 2 и умножение числа самого на себя, в результат ничего не записывается. Если же степень не четная, то также производится деление степени на 2 и в результат записывается умножение переменной результата на число, полученное при умножении ранее. Так как степень была разделена на 2, то число снова умножается на самого на себя. Завершение алгоритма происходит тогда, когда степень становится равной нулю.

Больше:

Сначала происходит сравнение длины чисел, если длина первого больше, то выводится true, если длина второго больше, то выводится false. Если числа одинаковой длины, то происходит поразрядное сравнение чисел, начиная с большего разряда.

Меньше:

Аналогично реализации «Больше»

Равно:
Сначала происходит сравнение длин чисел. Если они не равны, то выводится false. Если они равны, то производится поразрядное сравнение чисел.

Все исключения обрабатываются в main()

2 Исходный код

TLongArithmetic	Класс длинных чисел
Методы класса	
void Init(const std::string &str)	Преобразование строки в длинное число
bool operator < (const TLongArithmetic &rhs) const	сравнение чисел
bool operator > (const TLongArithmetic &rhs) const	сравнение чисел
bool operator == (const TLongArithmetic &rhs) const	сравнение чисел
TLongArithmetic operator ^(TLongArithmetic &rhs) const	Операция возведения в степень
TLongArithmetic operator + (const TLongArithmetic &rhs) const	Операция возведения сложения
TLongArithmetic operator - (const TLongArithmetic &rhs) const	Операция возведения вычитания
TLongArithmetic operator * (const TLongArithmetic &rhs) const	Операция возведения умножения
TLongArithmetic operator / (const TLongArithmetic &rhs) const	Операция возведения деления

```
using long_type = NLongArithmetic::TLongArithmetic;

int main() {
    long_type a, b, c;
    char action;
    while (std::cin >> a >> b >> action){
        c.Clear();
        switch (action) {
        case '+':
            c = a + b;
            std::cout << c << std::endl;
            break;
        case '-':
            try {
                c = a - b;
            } catch (const char * er) {
                std::cout << er << std::endl;
            }
            if (c.IsOk()) {
                std::cout << c << std::endl;
            }
            break;
        case '*':
            try {
                c = a * b;
            } catch (const char * er) {
                std::cout << er << std::endl;
            }
            if (c.IsOk()) {
                std::cout << c << std::endl;
            }
            break;
        case '^':
            try {
                c = a ^ b;
            } catch (const char * er) {
                std::cout << er << std::endl;
            }
            if (c.IsOk()) {
                std::cout << c << std::endl;
            }
            break;
        case '/':
            try {
                c = a / b;
            } catch (const char * er) {
                std::cout << er << std::endl;
            }
            if (c.IsOk()) {
```

```
50                 std::cout << c << std::endl;
51             }
52             break;
53         case '<':
54             if (a < b) {
55                 std::cout << "true" << std::endl;
56             } else {
57                 std::cout << "false" << std::endl;
58             }
59 
60             break;
61         case '>':
62             if (a > b) {
63                 std::cout << "true" << std::endl;
64             } else {
65                 std::cout << "false" << std::endl;
66             }
67             break;
68         case '=':
69             if (a == b) {
70                 std::cout << "true" << std::endl;
71             } else {
72                 std::cout << "false" << std::endl;
73             }
74             break;
75         default:
76             break;
77         }
78     }
79 
80 }
```

3 Консоль

```
Lera:Long_arithmetic_vsc valeriabudnikova$ make
./lab6 <test.txt >1.txt
Lera:Long_arithmetic_vsc valeriabudnikova$ cat 1.txt
14317716111385
90491408684812784630569
true
true
836561414558319702015930583133700765402231597245355454337573933453019302892O
59841167335419301693584171528748387953578559999
34545
false
256728343729212822931384520830499681671219235655338042871379597588458529800475480 20
42905185967604542201172616020983220 12
11800563070220827605539581658975581244562870981782145789069 5
false
29614189279089257695112364956125068662103854201296400282458667024123668651385709 9
25456251732528791578075417275783234510
Error
24755064948215838239363685075753673022233796553080981497698901013545777 2
74874234972421346195510528988434695714646567784819888707473 3
224821467216596
53228536397868315
true
17831677152270041158173505166271681079011459448784145595635145424004379 6
38254263202786594312519327100261109790600675859716638876503375523496763582865519 52
```

4 Тест производительности

Сравним время работы сложения, вычитания, умножения и деления с функциями из библиотеки *gmp.h*. Тесты проводятся на числах, длинна которых от 1, до 100 символов. Количество операций - 10000.
Сложение:

```
Lera:banchmark valeriabudnikova$ make run
./lab6 <test.txt >1.txt
cat 1.txt
time_my_lab: 0.010135 s  time_gmp: 0.000557 s
```

Вычитание:

```
./lab6 <test.txt >1.txt
cat 1.txt
time_my_lab: 0.013744 s  time_gmp: 0.002047 s
```

Умножение:

```
Lera:banchmark valeriabudnikova$ make run
./lab6 <test.txt >1.txt
cat 1.txt
time_my_lab: 0.027144 s  time_gmp: 0.000416 s
```

Деление:

```
Lera:banchmark valeriabudnikova$ make run
./lab6 <test.txt >1.txt
cat 1.txt
time_my_lab: 1.92452 s  time_gmp: 0.000436 s
```

Моя реализация проиграла функциям из gmp, причем в вычитании и сложении время отличается на $\backsim$ 10 мс, а в умножении и делении функции из gcd были быстрее в несколько раз.

5 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я научилась реализовывать арифметические операции над длинными числами, такие как сложение, вычитание, умножение, деление и возведение в степень, а также сравнивать длинные числа. Наибольшей сложностью в данной лабораторной работе была реализация алгоритма деления и возведения в степень.

Список литературы

[1] *Длинная арифметика— Википедия.*
URL: `https://ru.wikipedia.org/wiki/Длинная_арифметика` (дата обращения: 20.03.2021).