# 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: А. О. Дубинин
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:


**Москва, 2019**

# 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 операций:** Найти в заранее известном тексте поступающие на вход образцы.

- Сложение (+)
- Вычитание (-)
- Умножение (*)
- Возведение в степень (^)
- Деление (/)
- Больше (>)
- Меньше (<)
- Равно (=)

**Входные данные:** Входный файл состоит из последовательностей заданий, каждое задание состоит их трех строк:

- Первый операнд операции.
- Второй операнд операции.
- Символ арифметической операции или проверки условия (+, -, *, ^, /, >, <, =).

Числа, поступающие на вход программе, могут иметь «ведущие нули».

**Выходные данные:** Для каждого задания из входного файла нужно распечатать результат на отдельной строке в выходном файле:

- Числовой результат для арифметических операций.
- Строку Error в случае возникновения ошибки при выполнении арифметичесокй 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

В библиотеке реализованы следущие элементы:
Конструкторы для чисел в виде текстовой строки, для создание внутреннего в вектора с размером типа size_t, и для создание вектора с использование size_t и заполнением его числом типа int. Конструктур с числом в виде строки, дробит эту строку на цифры в степени BASE, записивая эти цифры в формат int и добавляя в вектор. Сложность $O(n)$, где n - кол-во символов в строке(числе).

Логические операции (больше, меньше, равно) сперва сверяют количество разрядов в числах, если они различны, то сразу выдается ответ, в ином случае числа сравниваются поочерёдно по разряду, начиная со старшего. Как только будет встречено неравенство, функция вернёт значение, если же неравенства не встречено во всех разрядах, то эти числа равны. Сложность $O(min(n; m))$ где n, m – количество разрядов в числах.

Сложение и вычитания реализованы подобно тому, как считают люди – столбиком. Если на каком-то шаге разряд превысил базу (сложение) или его нехватило (вычитание), то учитывается remainder, который переносит значение на разряд выше. При этом предусмотрена выдача ошибки если вычитаемое больше уменьшемого с помощью оператора $>=$. Сложность $O(max(n; m))$

Умножение реализовано похожим образом, столбиком : алгоритм проходит по всем цифрам множителя, и прибавляет получившееся произведение в результат с соответсвующим сдвигом. Сложность $O(n * m)$

Деление реализовано следующим путём: от делимого берётся часть числа – от старшего рязряда и поочерёдно следующие разряды. Методом двоичного поиска находится соответствующее частное, после, от взятой части отнимается получившееся частное и процесс повторяется пока не будет учтёно всё делимое. Предусмотрена проверка деления на ноль. Сложность $O(n * log(m))$

Возведение в степень сделано разложением степени бинарно: если степень чётная, то ответ равен произведению числа в два раза меньшей степени на эту же степень, т.е., фактически, возведение в квадрат в два раза меньшей степени, если нечётная, то произведению числа в два раза меньшей степени на самого себя (на следующем шаге степень уже будет чётной). Сложность $O(n * log(d))$, где d – степень

# 2 Исходный код

| TLongInt | |
|---|---|
| friend std::ostream& operator « (std::ostream& stream, TLongInt &number); | Вывод числа. |
| TLongInt(std::string &strNum); | Считать число со строки во внутренний формат. |
| void operator+(TLongInt &rhs); | Оператор сложения. |
| void operator-(TLongInt &rhs); | Оператор вычитания. |
| void operator*(TLongInt &rhs); | Оператор умножения. |
| void operator/(TLongInt &rhs); | Оператор деления. |
| void operator^(int num); | Оператор возведения в степень. |

# 3 Производительность

Реализованная библиотека сравнивается с библиотекой GSM. Сравнения были произведенны для *,/,+,-. Для этого был создан тестовый генератор выбирающий рандомно операции.

Тест на 100 операций:

```
<Gsm:
<Time of working 0 ms.
---
>TLongInt:
>Time of working 3 ms.
```

Тест на 1000 операций:

```
<Gsm:
<Time of working 6 ms.
---
>TLongInt:
>Time of working 43 ms.
```

Тест на 10000 операций:

```
<Gsm:
<Time of working 61 ms.
---
>TLongInt:
>Time of working 414 ms.
```

Из тестов видно, что GMP работает значительно быстрее на тестах с умножением,делением и возведением в степень. Это объясняется выбранными мной алгоритмамиреализации: существует более быстрое умножение, например алгоритм Карацубы

# 4 Выводы

Выполнив лабораторную работу по курсу «Дискретный анализ», я изучил методы создание длинной арифметики. Открыл для себя метод быстрого возведения в степень и деление с помощью поиска делители бинарным поиском. Остальные же задачи довольно тривиальны, используя школьные знания подсчета в столбик, легко можно запрограммировать операции +,- и *. Программирование длинной арифметики хоть и не сложно, но будет полезно людям на начальном этапе программирование, для отлаживанния ошибок, вызванных большим объемом кода арифметики.

Длинная арифметика встроена в таких языках программирования как Ruby, Python и Java. Также для C++ есть библиотека gmp, которая зарекомендавала себя как очень быстрое решение для длинной арифметики. Сомневаюсь, что я буду в дальнейшем использовать написанную библитеку – для повседневных целей гораздо удобнее использовать Python, но приобретённый опыт, думаю, найдёт своё применение.

# Список литературы

[1] *Emaxx - Длинная арифметика*
URL: `https://e-maxx.ru/algo/big_integer`

[2] *Итмо – Арифметика чисел в b-ичной системе счисления*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Арифметика_чисел_в_b-ичной_системе_счисления_(Длинная_арифметика)`